A LADOK3 Python API

Alexander Baltatzis, Daniel Bosk, Gerald Q. 'Chip' Maguire Jr.

KTH EECS
`{alba,dbosk,maguire}@kth.se`

27th October 2022

Abstract

We provide a Python wrapper for the LADOK3 REST API. We provide a useful object-oriented framework and direct API calls that return the unprocessed JSON data from LADOK.

Contents

Chapter 1

Introduction

LADOK (abbreviation of Lokalt Adb–baserat DOKumentationssystem, in Swedish) is the national documentation system for higher education in Sweden. This is the documented source code of `ladok3`, a LADOK3 API wrapper for Python.

The `ladok3` library provides a non-GUI application that, similar to access via a web browser, only provides the user with access to the LADOK data and functionality that they would actually have based on their specific user permissions in LADOK. It can be seem as a very streamlined web browser just for LADOK's web interface. While the library exploits caching to reduce the load on the LADOK server, this represents a subset of the information that would otherwise be obtained via LADOK's web GUI export functions.

You can install the `ladok3` package by running

```
pip3 install ladok3
```

in the terminal. You can find a quick reference by running

```
pydoc ladok3
```

We provide the main class `LadokSession` (Section 3.1). The `LadokSession` class acts like 'factories' and will return objects representing various LADOK data. These data objects' classes inherit the `LadokData` (Section 3.2.1) and `LadokRemoteData` (Section 3.2.3) classes. Data of the type `LadokData` is not expected to change, unlike `LadokRemoteData`, which is. Objects of type `LadokRemoteData` know how to update themselves, i.e., fetch and refresh their data from LADOK. When relevant they can also write data to LADOK, i.e., update entries such as results.

One design criteria is to improve performance. We do this by caching all factory methods of any `LadokSession`. The `LadokSession` itself is also designed to be cacheable: if the session to LADOK expires, it will automatically reauthenticate to establish a new session.

Part I

The library

Chapter 2

Overview

This is the documented source code of the LADOK3 API wrapper for Python, the package `ladok3`. We use the standard structure for the script. We start with the imports and then our class definitions follow in ⟨*classes* 5⟩ and some helper functions in ⟨*functions* 11⟩.

3a ⟨*ladok3.py* 3a⟩≡

```
"""A Python wrapper for the LADOK3 API"""
# -*- coding: utf-8 -*-
import cachetools
import datetime
import functools
import html
import json
import operator
import re
import requests
import urllib.parse

⟨classes 5⟩
⟨functions 11⟩
```

We also provide tests for all functionality in ⟨*test ladok3.py* 3b⟩. These illustrate how to the `ladok3` module and provide testing using `pytest`.

3b ⟨*test ladok3.py* 3b⟩≡

```
import json
import ladok3
import ladok3.kth
import os

⟨test functions 6a⟩
```

Chapter 3

Accessing LADOK

To access LADOK we provide a class `LadokSession`. An instance of this class provides the `session` attribute, which is an authenticated session to LADOK's REST API. It uses the `requests` module.

We can use the class as follows.

```python
import ladok3

ladok = ladok3.kth.LadokSession(
        "dbosk", "my secret password",
        test_environment=True) # for experiments
```

This chapter covers how the `LadokSession` class work. The remaining chapters cover what the `ladok` object can be used for. Chapter 6 covers how we can work with student data. Chapter 7 covers how we can work with course data.

3.1 The `LadokSession` class

There are two categories of functionality that we want from the `LadokSession` object. The first is the `session` attribute. This is an active and logged-in session that can be used to query LADOK (REST API over HTTPS). The second is the set of methods to get objects of interest, these methods use the `session` attribute to query data to create the objects. We cover the first (the `session` attribute) in this section and the second (the data methods) in Section 3.2.

We want to provide the attribute `session` which is an active and logged-in session for querying LADOK. We must thus handle authentication to LADOK. The authentication for using LADOK is performed using a university SSO system. The `login` method will set up a `session` object which is logged in. The `login` method will in turn use the `saml_login` method at the suitable place in the code to actually log in to LADOK by authenticating to the appropriate university. We can provide the interactions with LADOK, but not with the SSO system. So that one is left to implement by a university-specific child class. We provide `ladok3.kth.LadokSession` as an example.

We let the `session` attribute be a property, this way we can ensure that there exists an active session whenever it's used. This also means that we don't log in until we actually need to use the `session` attribute. We also automatically reauthenticate whenever the session has timed out.

5 ⟨*classes* 5⟩≡ (3a) 8 ▷

```
class LadokSession:
  """This is an interface for reading and writing data from and to LADOK."""
  def __init__(self, test_environment=False):
    """Log in and fetch base data"""
    self.__session = None
    self.__login_time = None
    ⟨LadokSession constructor body 6b⟩

  def login(self):
    """Log in to LADOK"""
    self.__login_time = datetime.datetime.now()
    try:
      ⟨log in to LADOK 7a⟩
    except Exception as err:
      self.__login_time = None
      raise err

  def saml_login(self, url):
    """Perform authentication to university SSO"""
    raise NotImplementedError()

  def logged_in(self):
    """Check if we have an active, logged-in session"""
    if self.__login_time:
      timeout = datetime.timedelta(minutes=15)
      return datetime.datetime.now() - self.__login_time < timeout
    return False

  def logout(self):
    """Close the session"""
    self.__login_time = None
    ⟨log out from LADOK 7d⟩

  @property
  def session(self):
    """A guaranteed to be active and logged in requests session to LADOK"""
    if not self.logged_in():
      self.login()
    self.__login_time = datetime.datetime.now()
    return self.__session

  @property
  def xsrf_token(self):
    cookies = self.session.cookies.get_dict()
    return next(cookies[cookie] for cookie in cookies if cookie == 'XSRF-TOKEN')
  @session.setter
  def session(self, new_value):
    self.__session = new_value

  ⟨LadokSession data methods 15⟩
```

Note that it's important to set `login_time` as soon as we start logging in (first thing in the `login` method), because the login function will actually use the `session` property.

The `saml_login` method must be implemented by a child class. See for example Chapter 9, which covers how to log in at KTH.

Now let's test this class. We will use the KTH version to log in.

6a ⟨*test functions* 6a⟩≡ (3b) 12a▷

```
ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments

def test_LadokSession():
  assert ladok.session is not None
```

3.1.1 Data about the session

The data that we must keep track of is which URL to use, i.e., which system to access: the production system or test system. To run against the test environment, change the base URL to

`https://www.test.ladok.se`[1]

6b ⟨*LadokSession constructor body* 6b⟩≡ (5) 6c▷

```
self.base_url = "https://www.start.ladok.se" if not test_environment \
  else "https://www.test.ladok.se"
self.base_gui_url = self.base_url + "/gui"
self.base_gui_proxy_url = self.base_gui_url + "/proxy"
```

We also set the headers that we will accept.

6c ⟨*LadokSession constructor body* 6b⟩+≡ (5) ◁6b 10b▷

```
self.headers = { 'Accept' : 'application/vnd.ladok-resultat+json, \
application/vnd.ladok-kataloginformation+json, \
application/vnd.ladok-studentinformation+json, \
application/vnd.ladok-studiedeltagande+json, \
application/vnd.ladok-utbildningsinformation+json, \
application/vnd.ladok-examen+json, application/vnd.ladok-extintegration+json, \
application/vnd.ladok-uppfoljning+json, application/vnd.ladok-extra+json, \
application/json, text/plain' }
```

[1] According to `https://ladok.se/drift-och-support/produktionsmiljo-for-nya-ladok`.

3.1.2 Signing in to LADOK

The aim of the `self.login()` method is to set up the `self.session` object which will be used for requests to LADOK. What we do here is to go through the same steps as a user would do with the web browser[2].

7a ⟨*log in to LADOK* 7a⟩≡ (5)

```
self.session = requests.session()
⟨get shibboleth url 7b⟩
response = self.saml_login(url)
⟨check SAML login response 7c⟩
```

We let `self.session` be our (HTTP) session to LADOK. The login functions set this up.

7b ⟨*get shibboleth url* 7b⟩≡ (7a)

```
response = self.session.get(url = self.base_gui_url+'/loggain')
response = self.session.get(url = self.base_gui_url+'/shiblogin')

shibstate = re.search('return=(.*?)(&|$)', response.url).group(1)
url = urllib.parse.unquote(shibstate)
```

Note that this URL is independent of the university chosen, which means we can simply authenticate to the university's system by passing the URL to the `saml_login` method for the university that we're interested in. See Chapter 9 for an example.

Check that login worked

When we have `relay_state` and `saml_response` from the SSO system, we can post them back to LADOK.

7c ⟨*check SAML login response* 7c⟩≡ (7a)

```
if 'Din användare finns inte i Ladok' in response.text:
  raise Exception('Signed in successfully, but not as a teacher.')
```

3.1.3 Signing out of LADOK

We might also want to close the session[3].

7d ⟨*log out from LADOK* 7d⟩≡ (5)

```
def logout(self):
  response = self.session.get(
    url=self.base_gui_proxy_url + '/logout',
    headers=self.headers)

  if response.status_code == 200:
    self.session.close()
    self.session = None
  else:
    raise Exception("Failed to log out of LADOK.")
```

[2]This neat trick is due to Alexander Baltatzis.

[3]This code is a slight modification of the code provided by Maguire.

3.2 The `LadokSession` data methods

The data methods are essentially factories for various objects mapping to data in LADOK. We will first look at the base classes and then provide methods that return instances of various subclasses.

3.2.1 LADOK data base class: `LadokData`

Some objects will only have read-only attributes, like grade scales, whereas others, like students, will have modifiable attributes, such as results, which must be synced.

We will provide properties for the various attributes. This way we can have read-only properties for attributes that cannot be changed, such as students' names, and read–write properties for attributes that can be changed, such as results.

We also want to serialize the data. We will use the JSON format for this. We provide this as a read only property `json`. If the output of `json` is given as the `kwargs` parameter to the `make_properties` method, it will restore all the attributes.

We provide the `LadokData` class with a constructor that takes keyword arguments (`**kwargs`). These are not used, but just so that we can add `super().__init__(**kwargs)` at the top of every constructor of the child classes.

Thus, we have the following base class.

8 ⟨*classes* 5⟩+≡ (3a) ◁5 9b▷

```
class LadokData:
  """Base class for LADOK data"""
  def __init__(self, /, **kwargs):
    pass

  def make_properties(self, kwargs):
    """Turn keywords into private attributes and read-only properties"""
    ⟨set up read-only properties 9c⟩

  def __eq__(self, other):
    if type(self) == type(other):
      return self.__dict__ == other.__dict__
    return False

  def __repr__(self):
    return str(self.json)

  @property
  def json(self):
    """JSON compatible dictionary representation of the object"""
    json_format = self.__dict__.copy()
    for key, value in json_format.items():
      if isinstance(value, LadokData):
        json_format[key] = value.json
    json_format["type"] = type(self).__name__
    return json_format
```

3.2.2 To and from JSON format

We want to be able to restore an object from that JSON formatted dictionary. We provide a function `restore`, which restores an object from a JSON-like dictionary. The `restore` function uses the constructor and, in turn, the `__setup_properties` method.

9a ⟨*function* 9a⟩≡
```
def restore(json_dict):
  if "type" not in json_dict:
    return LadokData(**json_dict)

  ⟨check type and return such object (never defined)⟩

  raise TypeError(f"cannot restor type {json_dict.type}, not a known type")
```

For completeness, we also provide `LadokDataEncoder`, a subclass of `JSONEncoder`.

9b ⟨*classes* 5⟩+≡ (3a) ◁8 10a▷
```
class LadokDataEncoder(json.JSONEncoder):
  def default(self, object):
    if isinstance(object, LadokData):
      return object.json
    return super().default(object)
```

For the `__setup_properties` method, we want it to set up private attributes and read-only properties for them. For each keyword in `kwargs`, we add the private attribute using `setattr(self, ...)` and the corresponding property to the class itself using `setattr(type(self), ...)`. (Note that we only add the property if it doesn't already exist.) We check the names, if the attributes already match the pattern for a private attribute name, we don't 'privatize' it again.

9c ⟨*set up read-only properties* 9c⟩≡ (8)
```
for attribute in kwargs:
  # private attributes are named on the form _class__attribute
  priv_attr_prefix = f"_{type(self).__name__}__"
  if priv_attr_prefix in attribute:
    priv_attr_name = attribute
    property_name = attribute.replace(priv_attr_prefix, "")
  else:
    priv_attr_name = priv_attr_prefix + attribute
    property_name = attribute

  setattr(self, priv_attr_name, kwargs[attribute])
  if not hasattr(type(self), property_name):
    setattr(type(self), property_name,
      property(operator.attrgetter(priv_attr_name)))
```

3.2.3 LADOK remote data base class: `LadokRemoteData`

The objects that can be modified can also be updated. They can be updated locally, which must be pushed to LADOK; they can be changed remotely, which must be pulled from LADOK. These objects must also keep a LADOK session internally (`self.ladok`).

10a ⟨*classes* 5⟩+≡ (3a) ◁9b 17a▷

```
class LadokRemoteData(LadokData):
  """Base class for remote LADOK data"""
  def __init__(self, /, **kwargs):
    super().__init__(**kwargs)
    if "_LadokRemoteData__ladok" in kwargs:
      self.make_properties(**kwargs)
    else:
      self.__ladok = kwargs.pop("ladok")

  @property
  def ladok(self):
    return self.__ladok

  def pull(self):
    """Pull updates for object from LADOK"""
    raise NotImplementedError("This object doesn't support pulling from LADOK")

  def push(self):
    """Push changes made to the object to LADOK"""
    raise NotImplementedError("This object doesn't support pushing to LADOK")
```

The LADOK session object provides some methods for searching for data, e.g., students. These methods will return instances of `LadokData` or `LadokRemoteData`.

3.2.4 Cached methods

Since the objects that can be updated can update themselves (`LadokRemoteData`), we can cache the factories to speed up interaction with LADOK. Since Python uses references for objects, a cached object will always be up-to-date if the program has made changes to it. This way we don't have to interact with the LADOK servers for every request, but only when necessary.

We will have a shared cache for all methods.

10b ⟨*LadokSession constructor body* 6b⟩+≡ (5) ◁6c

```
self.cache = {}
```

Then we can make the method caching by the following code.

```
@cachetools.cachedmethod(
  operator.attrgetter("cache"),
  key=functools.partial(cachetools.keys.hashkey, "method"))
def method(self, *args):
  pass
```

However, this construction cannot handle non-hashable objects (e.g., dictionaries and lists) as arguments. But the JSON representation of objects should be possible to make hashable.

Chapter 4

Helper functions

We have several helper functions that are useful throughout.

4.1 Filter on keys

Many data methods will filter objects based on attributes. We provide a function that takes a list of objects and a dictionary. Each key in the dictionary corresponds to an attribute for the objects. The value corresponding to the key, is checked for matching.

We will provide the following function, `filter_on_keys`, which does exactly that. This implementation runs through the (remaining) items once per keyword, meaning it's conjunctive — all key–values must match the attribute values of the objects.

We use the function `compare_values` to compare the values, we'll get back to that function below.

11 ⟨*functions* 11⟩≡ (3a) 12b ▷

```
def filter_on_keys(items, /, **kwargs):
  """
  Input:
  - items is a list of objects.
  - kwargs is a dictionary where keys match the attributes of the objects in
    items.

  Output:
  - Only objects where *all* key-value pairs match for the corresponding
    attribues.
  - If values are strings, the value from kwargs is interpreted as a regular
    expression.

  Example:
  student.first_name = "Student"
  student.last_name = "Studentdotter"

  filter_on_keys([student], firt_name="Student")
    gives [student]
  filter_on_keys([student], firt_name="Student", last_name="Studentsson")
    gives []
  """
  for key in kwargs:
    items = filter(
      lambda x: compare_values(operator.attrgetter(key)(x), kwargs[key]),
      items)
  return list(items)
```

We test this function with the following tests.

12a ⟨*test functions* 6a⟩+≡ (3b) ◁6a 13a▷

```
  def test_filter_on_keys():
    class Object:
      pass

    student1 = Object()
    student1.first_name = "Student"
    student1.last_name = "Studentsdotter"
    student2 = Object()
    student2.first_name = "Student"
    student2.last_name = "Studentsson"

    students = [student1, student2]
    assert ladok3.filter_on_keys(students, first_name="Student") == students
    assert ladok3.filter_on_keys(students,
      first_name="Student", last_name="Studentsdotter") == [student1]
    assert ladok3.filter_on_keys(students,
      last_name="Students(dotter|son)") == students
```

To achieve disjunctive behaviour, we must run `filter_on_keys` once per keyword. However, that causes problems if an item matches on more than one key. The following function solves that problem by not checking an item against more keys once it has matched one key.

12b ⟨*functions* 11⟩+≡ (3a) ◁11 13b▷

```
  def filter_on_any_key(items, /, **kwargs):
    """
    Input:
    - items is a list of objects.
    - kwargs is a dictionary where keys match the attributes of the objects in
      items.

    Output:
    - Only objects where *any* key-value pairs match for the corresponding
      attribues.
    - If values are strings, the value from kwargs is interpreted as a regular
      expression.

    Example:
    student.first_name = "Student"
    student.last_name = "Studentsdotter"
    filter_on_keys([student], firt_name="Student")
      gives [student]
    filter_on_keys([student], firt_name="Student", last_name="Studentsson")
      gives [student]
    """
    matching_items = []
    for item in items:
      for key in kwargs:
        if compare_values(operator.attrgetter(key)(item), kwargs[key]):
          matching_items.append(item)
          break

    return matching_items
```

We test this function with the following tests.

13a ⟨*test functions* 6a⟩+≡ (3b) ◁12a 14a▷

```
  def test_filter_on_any_key():
    class Object:
      pass

    student1 = Object()
    student1.first_name = "Student"
    student1.last_name = "Studentsdotter"
    student2 = Object()
    student2.first_name = "Student"
    student2.last_name = "Studentsson"

    students = [student1, student2]
    assert ladok3.filter_on_any_key(students,
      first_name="Student", last_name="Studentsdotter") == students
```

The function `compare_values` is how we compare the values. The reason we don't simply replace it with the `==` operator is that for strings, we actually want to use regular expressions for matching. The downside of this is that we might accidentally get substring matches when we expect exact matches. E.g., `F` will match both `F` and `Fx` unless we specify `^F$`.

13b ⟨*functions* 11⟩+≡ (3a) ◁12b 14b▷

```
  def compare_values(val1, val2):
    """
    Compares val1 and val2:
    - if val1 and val2 both are strings, then val2 is interpreted as a regular
      expression.
    - otherwise we use ==
    """
    if isinstance(val1, str) and isinstance(val2, str):
      return re.search(val2, val1)

    return val1 == val2
```

We test this function with the following tests.

14a ⟨*test functions* 6a⟩+≡ (3b) ◁13a

```
def test_compare_values():
  assert ladok3.compare_values("Studentsdotter", "^Student")
  assert ladok3.compare_values("Studentsdotter", "dotter")
  assert not ladok3.compare_values("Studentsdotter", "son")
  assert ladok3.compare_values(1, 1)
  assert not ladok3.compare_values(1, 2)
```

4.2 Extracting translations

In many cases, LADOK provides several translations. They do that in the form of this JSON structure:

```
{
  "Benamning": [
    {
      "Sprakkod": "sv",
      "Text": "Laborationer",
      "link": []
    },
    {
      "Sprakkod": "en",
      "Text": "Programming Assignments",
      "link": []
    }
  ]
}
```

We provide the function `get_translation` to get the `Text` based on a specified `Sprakkod`.

14b ⟨*functions* 11⟩+≡ (3a) ◁13b 21b▷

```
def get_translation(lang_code, list_of_translations):
  for translation in list_of_translations:
    if translation["Sprakkod"] == lang_code:
      return translation["Text"]
  raise KeyError(f"no translation for language {lang_code}")
```

Chapter 5

Grading scales

LADOK has various grading scales. Whenever we deal with results, we must use them and their identifiers. We provide classes that wraps LADOK's grading scales: `GradeScale` and `Grade`. We also provide a method, `get_grade_scales`, which returns a list of `GradeScale` objects.

5.1 The `get_grade_scales` method

We start with the method that returns the grading-scales objects. This method interacts with LADOK, so we want to cache its responses. We also want to be able to filter the responses, we do this by keyword arguments.

15 ⟨*LadokSession data methods* 15⟩≡ (5) 20▷

```
  @cachetools.cachedmethod(
    operator.attrgetter("cache"),
    key=functools.partial(cachetools.keys.hashkey, "grade_scale"))
  def get_grade_scales(self, /, **kwargs):
    """Return a list of (un)filtered grade scales"""
    if len(kwargs) == 0:
      return [GradeScale(**scale_data)
              for scale_data in self.grade_scales_JSON()]

    return filter_on_keys(self.get_grade_scales(), **kwargs)
```

The `grade_scales_JSON` method is part of the LADOK API and is documented in Section 11.1.

5.2 The `GradeScale` and `Grade` classes

We need a class for grade scales. This object should be read only and we never need to update it, so we can base it on `LadokData`. `kwargs` will have the following format:

```
{
  "Benamning": {
    "sv": "AF",
    "en": "AF"
  },
  "Beskrivning": {},
  "Betygsgrad": [
    {
      "GiltigSomSlutbetyg": true,
      "ID": 131661,
      "Kod": "A",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": true,
      "ID": 131667,
      "Kod": "B",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": true,
      "ID": 131673,
      "Kod": "C",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": true,
      "ID": 131679,
      "Kod": "D",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": true,
      "ID": 131691,
      "Kod": "E",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": false,
      "ID": 131696,
      "Kod": "FX",
      "link": []
    },
    {
      "GiltigSomSlutbetyg": false,
      "ID": 131697,
      "Kod": "F",
      "link": []
    }
  ],
  "Dolt": false,
  "Giltighetsperiod": {
    "Startdatum": "2007-07-01",
    "link": []
  },
  "ID": "131657",
  "Kod": "AF",
  "link": []
}
```

This makes the follow class design suitable.

17a ⟨*classes* 5⟩+≡ (3a) ◁10a 17b▷
```
  class GradeScale(LadokData):
    """A grade scale"""
    def __init__(self, /, **kwargs):
      super().__init__(**kwargs)

      if "_GradeScale__id" in kwargs:
        self.make_properties(**kwargs)
      else:
        self.__id = int(kwargs.pop("ID"))
        self.__code = kwargs.pop("Kod")
        self.__name = kwargs.pop("Benamning")["sv"]
        self.__grades = [Grade(**grade_data)
                          for grade_data in kwargs.pop("Betygsgrad")]

    @property
    def id(self):
      return self.__id

    @property
    def code(self):
      return self.__code

    @property
    def name(self):
      return self.__name

    def grades(self, /, **kwargs):
      """Returns grades filtered on keyword"""
      return filter_on_keys(self.__grades, **kwargs)

    def __contains__(self, grade):
      ⟨test if grade is in grading scale 18a⟩

    def __iter__(self):
      ⟨iterate over grades in grading scale 18b⟩
```

Then we need a class for the grades themselves. We construct a grade from the LADOK JSON data. We also implement easy comparison so that we can search for the string 'A' instead of the `Grade` object for the A grade.

17b ⟨*classes* 5⟩+≡ (3a) ◁17a 21a▷
```
  class Grade(LadokData):
```

```
  """An individual grade part of a grade scale"""
  def __init__(self, /, **json_data):
    """Constructor taking a dictionary (JSON-like) structure"""
    if "_Grade__id" in json_data:
      self.make_properties(**json_data)
    else:
      self.__id = int(json_data.pop("ID"))
      self.__code = json_data.pop("Kod")
      self.__accepted = json_data.pop("GiltigSomSlutbetyg")

  @property
  def id(self):
    return self.__id

  @property
  def code(self):
    return self.__code

  def __str__(self):
    return self.code

  @property
  def accepted(self):
    return self.__accepted

  def __eq__(self, other):
    if isinstance(other, Grade):
      return self.__dict__ == other.__dict__
    elif isinstance(other, str):
      return self.code == other
    else:
      raise NotImplementedError(f"can't test equality with {type(other)}")
```

Checking if a grade is in a grading scale We can now easily implement the check if a grade is in a grading scale.

18a ⟨*test if grade is in grading scale* 18a⟩≡ (17a)

```
  return grade in self.__grades
```

This works even if `grade` is a string, thanks to how `__eq__` is implemented in the `Grade` class.

Similarly, we can iterate over the grades in the grading scale.

18b ⟨*iterate over grades in grading scale* 18b⟩≡ (17a)

```
  return iter(self.__grades)
```

Chapter 6

Students

This chapter treats how we can work with student data. We can get student data from LADOK using the `LadokSession` class. We first create the `ladok` object, an instance of a LADOK session.

```python
import ladok3
import os

ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments
```

This is the same as in Chapter 3.

Next we want to access data of a student. We can refer to the student by personnummer or the unique LADOK ID.

```python
me = ladok.get_student("8506097891")
me2 = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")

print(f"{me.personnummer} {me.last_name}, {me.first_name}")
print(f"{me2.personnummer} {me2.last_name}, {me2.first_name}")
print(f"{me.ladok_id} == {me2.ladok_id}")
```

This gives us objects of the `Student` class. The objects `me` and `me2` both refer to the same person, so the outputs should be the same.

We can see which course instances the student is registered on (throughout history).

```python
for course in me.courses():
    print(f"{course.code} {course.name}")
```

We can select a particular course and we can get the results for that course in the same way.

```python
course = me.courses(code="DD2395")[0]

print(f"{course.code} results:")
for result in course.results():
    s = f"{course.code}"
    if result.component:
        s += f" {result.component}"
    s += f" {result.grade}"
    if result.attested:
        s += f" ({result.date})"
    print(s)
```

We can check if the result is attested or not. If it's not attested, we can change it.

```
student = ladok.get_student("1234561234")
prgi = student.courses(code="DD1315")[0]

print(f"{student.personnummer} {student.first_name} {student.last_name}")

for result in prgi.results():
    print(f"{result.component} {result.grade} ({result.date})", end="")
    if not result.attested:
        print("*")
    else:
        print()

print("Changing grades")

try:
    lab2 = prgi.results(component="LAB1")[0]
    lab2.set_grade("P", "2021-02-18")
    lab2.finalize()
except Exception as err:
    print(f"Couldn't change LAB1: {err}")
```

6.1 Getting students from LADOK

Perhaps the main data that we want to access is that of a student. We provide two ways to identify the student. The personal identity number (Swedish *personnummer*) and the unique LADOK identifier. These are so different in format that we can distinguish one from the other. However, we an delegate this job to the Student class.

20 ⟨*LadokSession data methods* 15⟩+≡ (5) ◁15 25▷

```
  @cachetools.cachedmethod(
    operator.attrgetter("cache"),
    key=functools.partial(cachetools.keys.hashkey, "get_student"))
  def get_student(self, id):
    """Get a student by unique ID, returns a Student object"""
    # note that self is the required LadokSession object
    return Student(ladok=self, id=id)
```

6.2 The `Student` class

We can create a `Student` object from an identifier and a `LadokSession` object. Then the `Student` object can fetch data from LADOK and update any changes.

We can determine if an identifier is a personnummer or LADOK ID. Based on either of those unique identifiers we can fetch the rest of the data using the LADOK session `self.ladok`.

21a ⟨*classes* 5⟩+≡ (3a) ◁17b 27▷

```
class Student(LadokRemoteData):
  """Class representing a student and related data"""
  def __init__(self, /, **kwargs):
    """Requires ladok (a LadokSession object),
    id (either a personnummer or LADOK ID)"""
    super().__init__(**kwargs)
    id = kwargs.pop("id")
    self.__personnummer = format_personnummer(id)
    if not self.__personnummer:
      self.__ladok_id = id
    else:
      self.__ladok_id = None

  def pull(self):
    """pull student data from LADOK"""
    ⟨pull all attributes from LADOK 23a⟩

  ⟨student attribute methods 22⟩
```

We use the function `format_personnummer` to format a personnummer according to LADOK's requirements. This function returns `None` if it's not a personnummer, in which case we assume that it is a LADOK ID in the code above.

21b ⟨*functions* 11⟩+≡ (3a) ◁14b

```
def format_personnummer(person_nr_raw):
  """Returns None or a LADOK-formated person nr"""
  pnrregex = re.compile(r"^(\d\d)?(\d\d)(\d\d\d\d)[+\-]?(\w\w\w\w)$")
  pnr = pnrregex.match(person_nr_raw)
  if pnr:
    now = datetime.datetime.now()
    if pnr.group(1) == None: # first digits 19 or 20 missing
      if now.year - 2000 >= int(pnr.group(2)) + 5: # must be > 5 years old
        return "20" + pnr.group(2) + pnr.group(3) + pnr.group(4)
      else:
        return "19" + pnr.group(2) + pnr.group(3) + pnr.group(4)
    else:
      return pnr.group(1) + pnr.group(2) + pnr.group(3) + pnr.group(4)
  else:
    return None
```

Note that we must match the start and end in the regex, otherwise we sometimes match parts of LADOK IDs as personnummer.

6.3 Students' personal attributes

The student's personal attributes are the following. We use the helper method `__get_personal_attributes` which pulls all personal attributes, since LADOK returns all of these in one call.

22 ⟨*student attribute methods* 22⟩≡ (21a) 23c ▷

```
def __get_personal_attributes(self):
  """Helper method that fetches personal attributes"""
  ⟨pull student attributes from LADOK 23b⟩

@property
def ladok_id(self):
  """Return the student's LADOK ID"""
  try:
    if self.__ladok_id:
      return self.__ladok_id
  except:
    pass
  self.__get_personal_attributes()
  return self.__ladok_id

@property
def personnummer(self):
  """Return the student's personnummer"""
  try:
    if self.__personnummer:
      return self.__personnummer
  except:
    pass
  self.__get_personal_attributes()
  return self.__personnummer

@property
def first_name(self):
  """Return the student's first name"""
  try:
    return self.__first_name
  except:
    self.__get_personal_attributes()
  return self.__first_name

@property
def last_name(self):
  """Return the student's last name"""
  try:
    return self.__last_name
  except:
    self.__get_personal_attributes()
  return self.__last_name
```

```
def __str__(self):
  return f"{self.personnummer} {self.first_name} {self.last_name}"

@property
def alive(self):
  """Return whether student is alive or not"""
  try:
    return self.__alive
  except:
    self.__get_personal_attributes()
  return self.__alive
```

Then we can call that method to update all attributes.

23a ⟨*pull all attributes from LADOK* 23a⟩≡ (21a) 24a▷

```
self.__get_personal_attributes()
```

When we pull the student attributes from LADOK, we can use either of the two IDs: `self.personnummer` and `self.ladok_id`. (However, we avoid using the properties above in this code, since we don't want to recursively trigger the fetch ad absurdum.) Depending on which attribute is set, we pull the data differently.

23b ⟨*pull student attributes from LADOK* 23b⟩≡ (22)

```
if self.__ladok_id:
  record = self.ladok.get_student_data_by_uid_JSON(self.__ladok_id)
elif self.__personnummer:
  record = self.ladok.get_student_data_JSON(self.__personnummer)
else:
  raise AttributeError("neither personnummer, nor LADOK ID set")

self.__ladok_id = record['Uid']
self.__personnummer = record['Personnummer'] # twelve digits only
self.__first_name = record['Fornamn']
self.__last_name = record['Efternamn']
self.__alive = not record['Avliden']
```

6.4 Students' study-related attributes

The study related attributes are courses and course results. Similarly to above, we use the `__get_study_attributes` helper method to populate all study-related attributes that LADOK returns in one request.

23c ⟨*student attribute methods* 22⟩+≡ (21a) ◁22

```
def __get_study_attributes(self):
  """Helper method to fetch study related attributes"""
  ⟨create a list of the student's course objects 24b⟩

def courses(self, /, **kwargs):
  """
  Returns a list of courses that the student is registered on.
  Filtered based on keywords, see ladok3.filter_on_keys for details.
```

```
    """
    try:
      courses = self.__courses
    except:
      self.__get_study_attributes()
      courses = self.__courses

    return filter_on_keys(courses, **kwargs)
```

Then we can call the last method when we want to pull all attributes too (using the `pull` method).

24a ⟨*pull all attributes from LADOK* 23a⟩+≡ (21a) ◁23a

```
  self.__get_study_attributes()
```

To create a list of the student's course registrations, we use the `registrations_JSON` API method.

24b ⟨*create a list of the student's course objects* 24b⟩≡ (23c)

```
  self.__courses = []

  for course in self.ladok.registrations_JSON(self.ladok_id):
    if not course['Nuvarande'] or \
      'Utbildningskod' not in course['Utbildningsinformation']:
      continue

    self.__courses.append(CourseRegistration(
      ladok=self.ladok,
      student=self,
      **course["Utbildningsinformation"]))
```

Chapter 7

Courses

We can use the course-related classes as follows.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# -*- mode: python; python-indent-offset: 4 -*-
import ladok3
import os

ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments

prgiX = ladok.search_course_rounds(code="DD1315")

for prgi in prgiX:
    print(f"{prgi.code} {prgi.start}--{prgi.end}")
print()

prgi = prgiX[0]
print(f"{prgi.code} {prgi.start}--{prgi.end}")
print(f"round: {prgi.round_id}")
print(f"round code: {prgi.round_code}")
print(f"instance: {prgi.instance_id}")
print(f"education: {prgi.education_id}")
print(f"{prgi.code} components:")
for component in prgi.components():
    print(f"{component.code}: {component.instance_id}")
```

7.1 Course rounds

We can search for courses like this. The result consists of a list of `CourseRound` objects.

25 ⟨*LadokSession data methods* 15⟩+≡ (5) ◁20
```
  @cachetools.cachedmethod(
    operator.attrgetter("cache"),
    key=functools.partial(cachetools.keys.hashkey, "search_courses"))
  def search_course_rounds(self, /, **kwargs):
    """Query LADOK about course rounds, possible keys:
    code, round_code, name
    """
    url = self.base_gui_proxy_url + "/resultat/kurstillfalle/filtrera?"

    if "code" in kwargs:
      url += f"kurskod={kwargs['code']}&"
    if "name" in kwargs:
      url += f"benamning={kwargs['name']}&"
    if "round_code" in kwargs:
      url += f"tillfalleskod={kwargs['round_code']}&"

    url += "page=1&limit=400&skipCount=false&sprakkod=sv"

    response = self.session.get(
      url=url,
      headers=self.headers)

    results = response.json()["Resultat"]

    return [CourseRound(ladok=self, **result) for result in results]
```

The response consists of this JSON data:

```
{
  "Resultat": [
    "0": {
      "Slutdatum": "2021-01-15",
      "Startdatum": "2020-08-24",
      "StudielokaliseringID": 135195,
      "StudietaktID": 6,
      "TillfallesKod": "51386",
      "Uid": "4e94fe55-1cef-11ea-a622-3565135944de",
      "UndervisningsformID": 1,
      "Utbildningsinstans": {
        "Avvecklad": false,
        "Benamning": [
          {
            "Sprakkod": "sv",
            "Text": "Programmeringsteknik och Matlab",
            "link": []
          },
          {
            "Sprakkod": "en",
            "Text": "Programming Techniques and Matlab",
            "link": []
          }
        ],
        "Enhet": "HP",
        "GallerUtbildningUtanAngivenOmfattning": false,
        "Omfattning": 7.5,
        "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
        "ResultatPaKursVidAttesteringAvModul": false,
        "Uid": "9f30cc02-d6b5-11e8-8fd5-cf9d2c5c41ba",
        "UtbildningUID": "38ab2393-73d8-11e8-afa7-8e408e694e54",
        "Utbildningskod": "DD1315",
        "Versionsnummer": 2,
        "link": []
      },
      "link": [ ... ]
    },
    "1": { ... },
    ...
  ],
  TotaltAntalPoster: 32
}
```

We have two classes that are relevant here, `CourseInstance` and `CourseRound`. The `CourseInstance` corresponds to the data in `Utbildningsinstans` above. And the `CourseRound` is the data 'wrapping around' that of the `CourseInstance`. Thus, we will make `CourseRound` a specialization of `CourseInstance`. This makes sense as it contains additional data, such as start and end date. This class takes care of those and then defers the rest to the `CourseInstance` class (by calling its constructor).

27 ⟨*classes* 5⟩+≡ (3a) ◁21a 33▷

```
class CourseInstance(LadokRemoteData):
  """Represents a course instance. Must be constructed from at least
  ladok (a LadokSession object),
  UtbildningsinstansUID (an instance_id from LADOK),
  optionally a data dictionary from LADOK"""
  ⟨CourseInstance methods 28b⟩

class CourseRound(CourseInstance):
  """Represents a course round"""
  def __init__(self, /, **kwargs):
    """Must be constructed from at least:
    Uid, TillfallesKod, Startdatum, Slutdatum"""
    ⟨prepare kwargs and call CourseInstance constructor 28a⟩

    self.__round_id = kwargs.pop("Uid")
    self.__round_code = kwargs.pop("TillfallesKod")

    self.__start = datetime.date.fromisoformat(kwargs.pop("Startdatum"))
    self.__end = datetime.date.fromisoformat(kwargs.pop("Slutdatum"))

  @property
  def round_id(self):
    return self.__round_id
  @property
  def round_code(self):
    return self.__round_code

  @property
  def start(self):
    return self.__start

  @property
  def end(self):
    return self.__end

  ⟨CourseRound data methods 34⟩
```

The `CourseInstance` constructor expects only the `Utbildningsinstans` part of the data. However, the `Uid` should be renamed `UtbildningsinstansUID`. We must also pass on the `ladok` object.

28a ⟨*prepare kwargs and call CourseInstance constructor* 28a⟩≡ (27)

```
  instance_data = kwargs.pop("Utbildningsinstans")
  instance_data["UtbildningsinstansUID"] = instance_data.pop("Uid")
  super().__init__(ladok=kwargs.pop("ladok"), **instance_data)
```

7.2 Course instances

We have the course instance as the base class. This is essentially an 'instance of a course syllabus'. We know from above that it inherits from `LadokRemoteData` and that it must be initialized with the keywords `ladok` (for its parent) and `UtbildningsinstansUID` (for itself) and optionally some data. This leaves the following methods.

28b ⟨*CourseInstance methods* 28b⟩≡ (27)

```
  def __init__(self, /, **kwargs):
    super().__init__(**kwargs)
    self.__instance_id = kwargs.pop("UtbildningsinstansUID")

    try:
      self.__assign_attr(kwargs)
    except:
      self.__pull_attributes()

  def __assign_attr(self, data):
    ⟨assign CourseInstance data to private attributes 32⟩

  def __pull_attributes(self):
    ⟨fetch CourseInstance data object from LADOK 29⟩
    self.__assign_attr(data)

  def pull(self):
    self.__pull_attributes()
```

```
@property
def instance_id(self):
  return self.__instance_id

@property
def education_id(self):
  return self.__education_id

@property
def code(self):
  return self.__code

@property
def name(self):
  return self.__name.copy()

@property
def version(self):
  return self.__version

@property
def grade_scale(self):
  return self.__grade_scale

@property
def credits(self):
  return self.__credits

@property
def unit(self):
  return self.__unit

def components(self, /, **kwargs):
  """Returns the list of components, filtered on keywords"""
  return filter_on_keys(self.__components, **kwargs)
```

Now we must fetch data from LADOK.

29 ⟨*fetch CourseInstance data object from LADOK* 29⟩≡ (28b)

```
response = self.ladok.course_instance_components_JSON(self.instance_id)
data = response["Utbildningsinstans"][0]
```

Then data will be populated with the following values:

```
{
  "Avvecklad": false,
  "Benamning": [
    {
      "Sprakkod": "sv",
      "Text": "Programmeringsteknik och Matlab",
      "link": []
    },
    {
      "Sprakkod": "en",
      "Text": "Programming Techniques and Matlab",
      "link": []
    }
  ],
  "BetygsskalaID": 131657,
  "Enhet": "HP",
  "GallerUtbildningUtanAngivenOmfattning": false,
  "KravPaHanvisningTillBeslutshandling": false,
  "KravPaProjekttitel": false,
  "Moduler": [
    {
      "Avvecklad": false,
      "Benamning": [
        {
          "Sprakkod": "sv",
          "Text": "Laborationer",
          "link": []
        },
        {
          "Sprakkod": "en",
          "Text": "Programming Assignments",
          "link": []
        }
      ],
      "BetygsskalaID": 131656,
      "Enhet": "HP",
      "GallerUtbildningUtanAngivenOmfattning": false,
      "KravPaHanvisningTillBeslutshandling": false,
      "KravPaProjekttitel": false,
      "Moduler": [],
      "Omfattning": 1.5,
      "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
      "ResultatPaKursVidAttesteringAvModul": false,
      "Uid": "387c7248-73d8-11e8-b4e0-063f9afb40e3",
      "UtbildningUID": "387c99a9-73d8-11e8-afa7-8e408e694e54",
      "Utbildningskod": "MAT1",
      "Versionsnummer": 1,
      "link": [ ... ]
    },
    {
      "Avvecklad": false,
      "Benamning": [
        {
          "Sprakkod": "sv",
          "Text": "Laborationer",
          "link": []
        },
        {
            "Sprakkod": "en",
            "Text": "Programming Assignments",
            "link": []
          }
        ],
        "BetygsskalaID": 131657,
        "Enhet": "HP",
        "GallerUtbildningUtanAngivenOmfattning": false,
        "KravPaHanvisningTillBeslutshandling": false,
        "KravPaProjekttitel": false,
        "Moduler": [],
        "Omfattning": 3,
        "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
        "ResultatPaKursVidAttesteringAvModul": false,
        "Uid": "389b1dda-73d8-11e8-b4e0-063f9afb40e3",
        "UtbildningUID": "389b1e62-73d8-11e8-afa7-8e408e694e54",
        "Utbildningskod": "LAB3",
        "Versionsnummer": 1,
        "link": [ ... ]
      },
      {
        "Avvecklad": false,
        "Benamning": [
          {
            "Sprakkod": "sv",
            "Text": "Laborationer",
            "link": []
          },
          {
            "Sprakkod": "en",
            "Text": "Programming Assignments",
            "link": []
          }
        ],
        "BetygsskalaID": 131656,
        "Enhet": "HP",
        "GallerUtbildningUtanAngivenOmfattning": false,
        "KravPaHanvisningTillBeslutshandling": false,
        "KravPaProjekttitel": false,
        "Moduler": [],
        "Omfattning": 1.5,
        "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
        "ResultatPaKursVidAttesteringAvModul": false,
        "Uid": "38a1867d-73d8-11e8-b4e0-063f9afb40e3",
        "UtbildningUID": "38a1d44d-73d8-11e8-afa7-8e408e694e54",
        "Utbildningskod": "LAB2",
        "Versionsnummer": 1,
        "link": [ ... ]
      },
      {
```

```
      "Avvecklad": false,
      "Benamning": [
        {
          "Sprakkod": "sv",
          "Text": "Laborationer",
          "link": []
        },
        {
          "Sprakkod": "en",
          "Text": "Programming Assignments",
          "link": []
        }
      ],
      "BetygsskalaID": 131656,
      "Enhet": "HP",
      "GallerUtbildningUtanAngivenOmfattning": false,
      "KravPaHanvisningTillBeslutshandling": false,
      "KravPaProjekttitel": false,
      "Moduler": [],
      "Omfattning": 1.5,
      "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
      "ResultatPaKursVidAttesteringAvModul": false,
      "Uid": "38a6416f-73d8-11e8-b4e0-063f9afb40e3",
      "UtbildningUID": "38a66848-73d8-11e8-afa7-8e408e694e54",
      "Utbildningskod": "LAB1",
      "Versionsnummer": 1,
      "link": [ ... ]
    }
  ],
  "Omfattning": 7.5,
  "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
  "ResultatPaKursVidAttesteringAvModul": false,
  "Uid": "9f30cc02-d6b5-11e8-8fd5-cf9d2c5c41ba",
  "UtbildningUID": "38ab2393-73d8-11e8-afa7-8e408e694e54",
  "Utbildningskod": "DD1315",
  "Versionsnummer": 2,
  "link": [ ... ]
}
```

Now that we have the `data` object, we can assign its values to the private attributes.

32 ⟨*assign CourseInstance data to private attributes* 32⟩≡ (28b)

```
self.__education_id = data.pop("UtbildningUID")

self.__code = data.pop("Utbildningskod")
self.__name = {}
names = data.pop("Benamning")
for name in names:
  self.__name[name["Sprakkod"]] = name["Text"]
self.__version = data.pop("Versionsnummer")
self.__credits = data.pop("Omfattning")
self.__unit = data.pop("Enhet")

self.__grade_scale = self.ladok.get_grade_scales(
  id=data.pop("BetygsskalaID"))

self.__components = [CourseComponent(
    ladok=self.ladok,
    **component) for component in data["Moduler"]]
```

7.3 Course components

The `CourseComponent` class will make the attributes available. We specify the most interesting ones and let the `LadokData` constructor turn the rest into properties as well, so that they are available for the curious.

33 ⟨*classes* 5⟩+≡ (3a) ◁27 36▷

```
class CourseComponent(LadokData):
  """Represents a course component of a course registration"""
  def __init__(self, /, **kwargs):
    super().__init__(**kwargs)

    if "UtbildningsinstansUID" in kwargs:
      self.__instance_id = kwargs.pop("UtbildningsinstansUID")
    else:
      self.__instance_id = kwargs.pop("Uid")

    self.__education_id = kwargs.pop("UtbildningUID")

    self.__code = kwargs.pop("Utbildningskod")
    description = kwargs.pop("Benamning")
    if isinstance(description, dict):
      self.__description = get_translation("sv", description)
    else:
      self.__description = description

    self.__credits = kwargs.pop("Omfattning")
    self.__unit = kwargs.pop("Enhet")

    ladok = kwargs.pop("ladok")
    grade_scale_id = kwargs.pop("BetygsskalaID")
    self.__grade_scale = ladok.get_grade_scales(id=grade_scale_id)[0]

  @property
  def instance_id(self):
    return self.__instance_id

  @property
```

```
    def education_id(self):
      return self.__education_id

    @property
    def code(self):
      """Returns the name of the component (as in syllabus)"""
      return self.__code

    @property
    def description(self):
      """Returns description of component (as in syllabus)"""
      return self.__description

    @property
    def unit(self):
      """Returns the unit for the credits"""
      return self.__unit

    @property
    def credits(self):
      """Returns the number of credits"""
      return self.__credits

    @property
    def grade_scale(self):
      return self.__grade_scale

    def __str__(self):
      return self.code

    def __eq__(self, other):
      if isinstance(other, str):
        return self.code == other
      return self.__dict__ == other.__dict__
```

We have the latter method, `__eq__`, so that we can filter on keys and compare with strings.

7.4 Results for a course round

We can access the results of all the students during a course round. We do this through the method `results`, which takes keyword arguments to possibly filter the results.

34 ⟨*CourseRound data methods* 34⟩≡ (27) 35a▷

```
  def results(self, /, **kwargs):
    """Returns all students' results on the course"""
    try:
      self.__results
    except:
      self.__fetch_results()
    return filter_on_keys(self.__results, **kwargs)
```

To fetch the results from LADOK, we must do the following query.

35a ⟨*CourseRound data methods* 34⟩+≡ (27) ◁34 35b▷

```
  def __fetch_results(self):
    raise NotImplementedError(
      f"{type(self).__name__}.__fetch_results not implemented")
```

7.5 Participants for a course round

We want to get a list of participants for a course round, i.e., a list of `Student` objects.

35b ⟨*CourseRound data methods* 34⟩+≡ (27) ◁35a 35c▷

```
  def participants(self, /, **kwargs):
    """Returns a Student object for each participant in the course."""
    try:
      self.__participants
    except:
      self.__fetch_participants()

    return filter_on_keys(self.__participants, **kwargs)
```

When we fetch the participants, we don't create new `Student` objects. We use the `get_student` method of the LADOK session object to fetch objects from the cache if they already exist.

35c ⟨*CourseRound data methods* 34⟩+≡ (27) ◁35b

```
  def __fetch_participants(self):
    self.__participants = []
    for student in self.ladok.participants_JSON(self.round_id):
      self.__participants.append(
        self.ladok.get_student(student["Student"]["Uid"]))
```

Chapter 8

Course objects related to students

8.1 Course registrations and a student's results

Students register for a course. The `CourseRegistration` class represents this data from LADOK. This is the object that must be used to get results for a student.

36 ⟨*classes* 5⟩+≡ (3a) ◁33 38b▷

```
class CourseRegistration(CourseInstance):
  """Represents a student's participation in a course instance"""
  def __init__(self, /, **kwargs):
    super().__init__(**kwargs)

    self.__student = kwargs.pop("student")

    # ett Ladok-ID för kursomgången
    self.__round_id = kwargs.pop("UtbildningstillfalleUID")
    self.__round_code = kwargs.pop("Utbildningstillfalleskod")

    dates = kwargs.pop("Studieperiod")
    self.__start = datetime.date.fromisoformat(dates["Startdatum"])
    self.__end = datetime.date.fromisoformat(dates["Slutdatum"])

  @property
  def round_id(self):
    """Returns LADOK ID for the course round (kursomgång)"""
    return self.__round_id

  @property
  def round_code(self):
    """Returns the human-readable round code (tillfälleskod)"""
    return self.__round_code

  @property
  def start(self):
    return self.__start

  @property
  def end(self):
    return self.__end

  def __str__(self):
    return f"{self.code} {self.round_code} ({self.start}-{self.end})"

  def __repr__(self):
    return f"{self.code}:{self.round_code}:{self.start}-{self.end}"

  def results(self, /, **kwargs):
    """Returns the student's results on the course, filtered on keywords"""
    try:
      return filter_on_keys(self.__results, **kwargs)
    except:
      self.__fill_results()
    return filter_on_keys(self.__results, **kwargs)

  def __fill_results(self):
    """Helper method to fetch results from LADOK"""
    ⟨pull existing CourseResult objects from LADOK 37⟩
    ⟨add new CourseResult objects for missing components 38a⟩

  def push(self):
    """Pushes any new results"""
    for result in self.results():
      result.push()
```

We can pull these from LADOK as well. We construct `CourseResult` objects using a `CourseRegistration` object. From the response we get from LADOK, we construct `CourseResult` objects that deal with the details.

37 ⟨*pull existing CourseResult objects from LADOK* 37⟩≡ (36)

```
response = self.ladok.student_results_JSON(
  self.__student.ladok_id, self.round_id
)

self.__results_id = response["Uid"]
self.__results = []
for result in response["ResultatPaUtbildningar"]:
  try:
    self.__results.append(CourseResult(
      ladok=self.ladok,
      components=self.components(),
      student=self.__student,
      study_results_id=self.__results_id,
      **result))
  except TypeError:
      pass
```

Now we can see which components from `self.components` are missing from `self.__results` and just add empty results for those.

38a ⟨*add new CourseResult objects for missing components* 38a⟩≡ (36)

```
  for component in self.components():
    if not list(filter_on_keys(self.__results, component=component.code)):
      self.__results.append(
        CourseResult(
          ladok=self.ladok,
          component=component,
          student=self.__student,
          study_results_id=self.__results_id))
```

8.2 Course results

The `CourseResult` objects have the following form. We have two cases: the result exists in LADOK and the result doesn't exist in LADOK. The difference is in how results are pushed to LADOK, i.e., the implementation of the `push` method.

38b ⟨*classes* 5⟩+≡ (3a) ◁36

```
  class CourseResult(LadokRemoteData):
    """Represents a result on a course module"""
    def __init__(self, /, **kwargs):
      """To construct this object we must give existing data, i.e.
      Arbetsunderlag or SenastAttesteradeResultat directly from LADOK."""
      super().__init__(**kwargs)

      self.__student = kwargs.pop("student")
      self.__study_results_id = kwargs.pop("study_results_id")

      if "component" in kwargs:
        self.__component = kwargs.pop("component")
        self.__attested = False
        self.__populate_attributes()
      elif "components" in kwargs and \
          ("Arbetsunderlag" in kwargs or "SenastAttesteradeResultat" in kwargs):
        components = kwargs.pop("components")

        if "Arbetsunderlag" in kwargs:
          self.__attested = False
          data = kwargs.pop("Arbetsunderlag")
        elif "SenastAttesteradeResultat" in kwargs:
          self.__attested = True
          data = kwargs.pop("SenastAttesteradeResultat")

        self.__populate_attributes(**data, components=components)
      else:
        raise TypeError("not enough keys given to construct object")
```

```
def __populate_attributes(self, /, **data):
  if not data:
    ⟨populate CourseResult attributes for empty result 40b⟩
  else:
    ⟨populate CourseResult attributes from data 40a⟩

@property
def component(self):
  """Returns the component the results is for"""
  return self.__component

@property
def grade_scale(self):
  """Returns the grade scale for the component"""
  return self.__grade_scale

@property
def grade(self):
  """Returns the grade set for the component"""
  return self.__grade

def set_grade(self, grade, date):
  """Sets a new grade and date for the component"""
  if self.attested:
    raise AttributeError("can't change already attested grade")

  ⟨set the grade for CourseResult 41a⟩

  self.__modified = True
  self.push()

def finalize(self, notify=False):
  """Finalizes the set grade"""
  if self.modified:
    self.push()

  ⟨finalize the grade for CourseResult 42b⟩

@property
def modified(self):
  """Returns True if there are unpushed changes"""
  return self.__modified

@property
def date(self):
  """Returns the date of the grade"""
  return self.__date

@property
```

```
  def attested(self):
    """Returns True if the grade has been attested in LADOK"""
    return self.__attested

  def push(self):
    if self.__uid:
      ⟨push the existing CourseResult grade data to LADOK 41b⟩
    else:
      ⟨push the new CourseResult grade data to LADOK 42a⟩
    self.__modified = False
```

If we get the data from LADOK, then we can fill in all the attributes.

40a ⟨*populate CourseResult attributes from data* 40a⟩≡ (38b)

```
self.__uid = data.pop("Uid")
self.__instance_id = data.pop("UtbildningsinstansUID")
self.__results_id = data.pop("ResultatUID")
self.__study_results_id = data.pop("StudieresultatUID")

grade_scale_id = data.pop("BetygsskalaID")
grade = data.pop("Betygsgrad")

self.__date = data.pop("Examinationsdatum")
self.__grade_scale = self.ladok.get_grade_scales(id=grade_scale_id)[0]
self.__grade = self.__grade_scale.grades(id=grade)[0]

if "components" in data:
  components = data.pop("components")
  component_list = filter_on_keys(components, instance_id=self.__instance_id)
  self.__component = component_list[0] if component_list \
                                       else None

self.__last_modified = data.pop("SenasteResultatandring")
self.__modified = False
```

However, if this is a completely new result, it doesn't exist in LADOK. Then we lack quite a few of the attributes.

40b ⟨*populate CourseResult attributes for empty result* 40b⟩≡ (38b)

```
self.__uid = None
self.__instance_id = self.__component.instance_id

self.__date = None
self.__grade_scale = self.__component.grade_scale
self.__grade = None

self.__modified = False
self.__last_modified = None
```

8.2.1 Setting the grade

We can accept the grade either as a grade object or as a string. We must check both values for correctness before we update either of the attributes — otherwise we might end up in an inconsistent state.

41a ⟨*set the grade for CourseResult* 41a⟩≡ (38b)

```
if isinstance(grade, Grade) and grade not in self.grade_scale.grades():
  raise TypeError(f"The grade {grade} is not in"
    f"the scale {self.grade_scale.code}")
elif isinstance(grade, str):
  try:
    grade = self.grade_scale.grades(code=grade)[0]
  except:
    raise TypeError(
      f"The grade {grade} is not in the scale {self.grade_scale.code}")
else:
  raise TypeError(f"Can't use type {type(grade)} for grade")

if isinstance(date, str):
  date = datetime.date.fromisoformat(date)
elif not isinstance(date, datetime.date):
  raise TypeError(f"Type {type(date)} not supported for date")

self.__grade = grade
self.__date = date
```

8.2.2 Working with existing results

To push an updated result to LADOK, we do the following.

41b ⟨*push the existing CourseResult grade data to LADOK* 41b⟩≡ (38b)

```
try:
  response = self.ladok.update_result_JSON(
    self.grade.id, self.grade_scale.id, self.date.isoformat(),
    self.__uid, self.__last_modified
  )
except Exception as err:
  raise Exception(
    f"couldn't update {self.component.code} to {self.grade} ({self.date})"
    f" to LADOK: {err}"
  )

self.__populate_attributes(**response[0])
```

8.2.3 Adding new results

We treat new results differently. Since we don't update an already existing item, we must add a new result to LADOK. Particularly, we must add something to the student's study results.

42a ⟨*push the new CourseResult grade data to LADOK* 42a⟩≡ (38b)
```
try:
  response = self.ladok.create_result_JSON(
    self.grade.id, self.grade_scale.id, self.date.isoformat(),
    self.__study_results_id, self.__instance_id
  )
except Exception as err:
  raise Exception("Couldn't register "
    f"{self.component} {self.grade} {self.date}: {err}")

self.__populate_attributes(**response[0])
```

8.2.4 Finalizing a result

When we finalize the result, we must know who reported the result.

42b ⟨*finalize the grade for CourseResult* 42b⟩≡ (38b)
```
reporter_id = self.ladok.user_info_JSON()["AnvandareUID"]

if notify:
  response = self.ladok.finalize_result_JSON(
    self.__results_id, self.__last_modified, reporter_id, reporter_id
  )
else:
  response = self.ladok.finalize_result_JSON(
    self.__results_id, self.__last_modified, reporter_id
  )

self.__populate_attributes(**response)
```

Part II

Logging in at different universities

Chapter 9

Logging in at KTH

In this chapter we detail how we log into LADOK using KTH. This means that we want to provide a child class of `ladok3.LadokSession` which has the `saml_login` method implemented.

44a ⟨*kth.py* 44a⟩≡
```
import html
import ladok3
import re

class LadokSession(ladok3.LadokSession):
  def __init__(self, username, password, test_environment=False):
    """Initialize KTH version of LadokSession"""
    super().__init__(test_environment=test_environment)
    self.__username = username
    self.__password = password

  ⟨KTH login methods 47a⟩

  def saml_login(self, url):
    """Do the SSO login"""
    ⟨KTH login procedure 45⟩
```

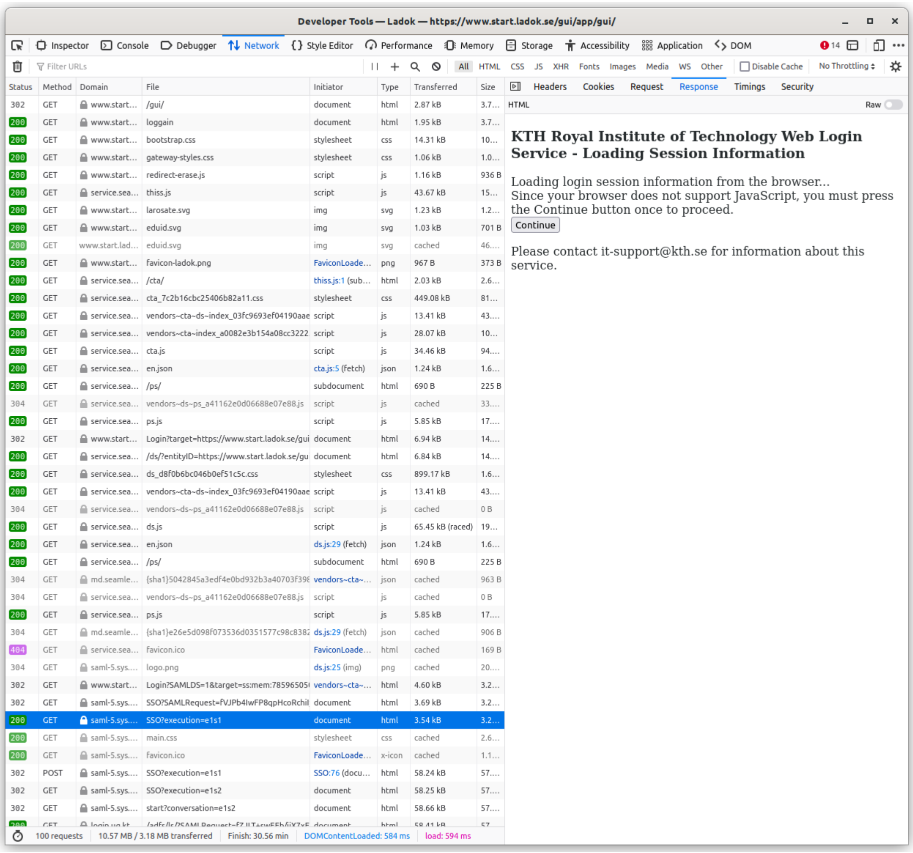
We also test this login functionality.

44b ⟨*test kth.py* 44b⟩≡
```
import json
import ladok3.kth
import os

ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments

def test_LadokSession():
  assert ladok.session is not None
```

To figure out how to log in at KTH, we investigate how the browser does it. We use the web debugger to find the web requests that we also must make. See


Developer Tools — Ladok — https://www.start.ladok.se/gui/app/gui/

Status	Method	Domain	File	Initiator	Type	Transferred	Size
302	GET	www.start...	/gui/	document	html	2.87 kB	3.7...
200	GET	www.start...	loggain	document	html	1.95 kB	3.7...
200	GET	www.start...	bootstrap.css	stylesheet	css	14.31 kB	10...
200	GET	www.start...	gateway-styles.css	stylesheet	css	1.06 kB	1.0...
200	GET	www.start...	redirect-erase.js	script	js	1.16 kB	936 B
200	GET	service.sea...	thiss.js	script	js	43.67 kB	15...
200	GET	www.start...	larosate.svg	img	svg	1.23 kB	1.2...
200	GET	www.start...	eduid.svg	img	svg	1.03 kB	701 B
200	GET	www.start.lad...	eduid.svg	img	svg	cached	46....
200	GET	www.start...	favicon-ladok.png	FaviconLoade...	png	967 B	373 B
200	GET	service.sea...	/cta/	thiss.js:1 (sub...	html	2.03 kB	2.6...
200	GET	service.sea...	cta_7c2b16cbc25406b82a11.css	stylesheet	css	449.08 kB	81...
200	GET	service.sea...	vendors~cta~ds~index_03fc9693ef04190aae	script	js	13.41 kB	43....
200	GET	service.sea...	vendors~cta~index_a0082e3b154a08cc3222	script	js	28.07 kB	10...
200	GET	service.sea...	cta.js	script	js	34.46 kB	94....
200	GET	service.sea...	en.json	cta.js:5 (fetch)	json	1.24 kB	1.6...
200	GET	service.sea...	/ps/	subdocument	html	690 B	225 B
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	33....
200	GET	service.sea...	ps.js	script	js	5.85 kB	17....
302	GET	www.start...	Login?target=https://www.start.ladok.se/gui	document	html	6.94 kB	14....
200	GET	service.sea...	/ds/?entityID=https://www.start.ladok.se/gu	document	html	6.84 kB	14....
200	GET	service.sea...	ds_d8f0b6bc046b0ef51c5c.css	stylesheet	css	899.17 kB	1.6...
200	GET	service.sea...	vendors~cta~ds~index_03fc9693ef04190aae	script	js	13.41 kB	43....
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	0 B
200	GET	service.sea...	ds.js	script	js	65.45 kB (raced)	19...
200	GET	service.sea...	en.json	ds.js:29 (fetch)	json	1.24 kB	1.6...
200	GET	service.sea...	/ps/	subdocument	html	690 B	225 B
304	GET	md.seamle...	{sha1}5042845a3edf4e0bd932b3a40703f398	vendors~cta~...	json	cached	963 B
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	0 B
200	GET	service.sea...	ps.js	script	js	5.85 kB	17....
304	GET	md.seamle...	{sha1}e26e5d098f073536d0351577c98c8382	ds.js:29 (fetch)	json	cached	906 B
404	GET	service.sea...	favicon.ico	FaviconLoade...	html	cached	169 B
304	GET	saml-5.sys....	logo.png	ds.js:25 (img)	png	cached	20....
302	GET	www.start...	Login?SAMLDS=1&target=ss:mem:78596505	vendors~cta~...	html	4.60 kB	3.2...
302	GET	saml-5.sys....	SSO?SAMLRequest=fVJPb4IwFP8qpHcoRchil	document	html	3.69 kB	3.2...
200	GET	saml-5.sys....	SSO?execution=e1s1	document	html	3.54 kB	3.2...
200	GET	saml-5.sys....	main.css	stylesheet	css	cached	2.6...
200	GET	saml-5.sys....	favicon.ico	FaviconLoade...	x-icon	cached	1.1...
302	POST	saml-5.sys....	SSO?execution=e1s1	SSO:76 (docu...	html	58.24 kB	57....
302	GET	saml-5.sys....	SSO?execution=e1s2	document	html	58.25 kB	57....
302	GET	saml-5.sys....	start?conversation=e1s2	document	html	58.66 kB	57....
200	GET	login.ug.kt	/adfs/ls/?SAMLRequest=fZJLT+swFEh/iiX7xF	document	html	58.41 kB	57

100 requests | 10.57 MB / 3.18 MB transferred | Finish: 30.56 min | DOMContentLoaded: 584 ms | load: 594 ms

HTML

KTH Royal Institute of Technology Web Login Service - Loading Session Information

Loading login session information from the browser...
Since your browser does not support JavaScript, you must press the Continue button once to proceed.

Continue

Please contact it-support@kth.se for information about this service.

Figure 9.1: Firefox Developer Tools showing the requests when loggin in to LADOK through KTH.

Fig. 9.1.

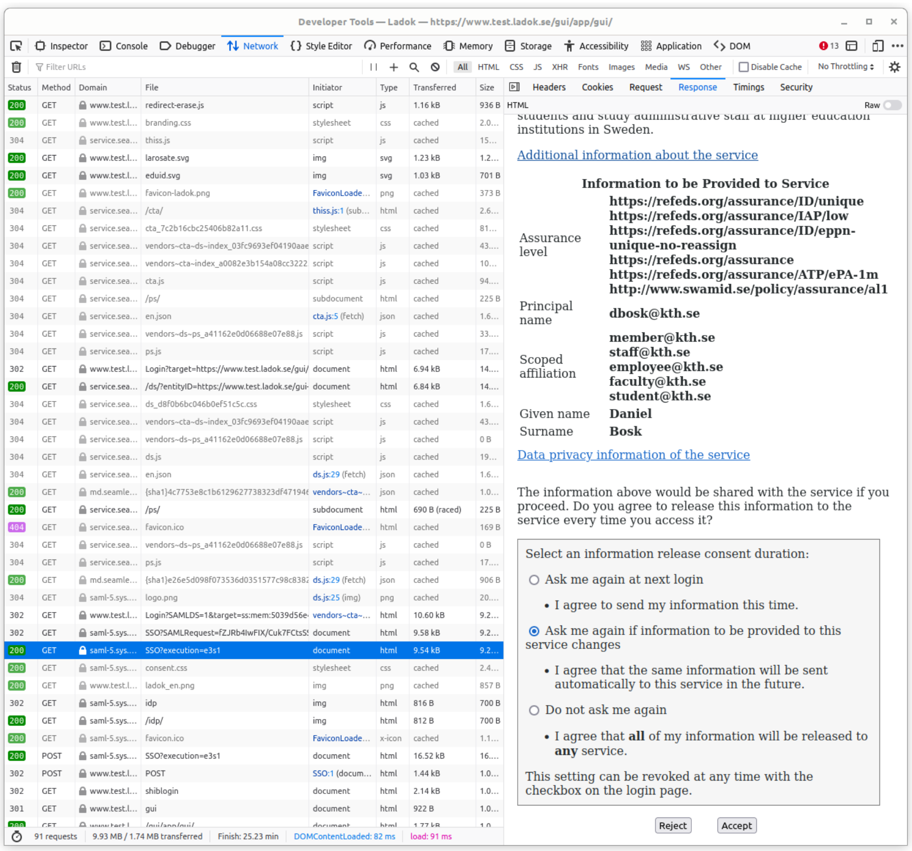
The only downside to our approach is that we don't handle the case when KTH asks if it is fine to pass our data to LADOK. This happens the first time we log in to a place using KTH's SSO. See Fig. 9.2.

9.1 Logging in

LADOK gets a URL to the KTH login system. This is the `url` parameter to the `saml_login` method. Now, we can use that URL to authenticate to the university. We must simulate what we do in the browser. In the case of the KTH UG/LDAP system, we do the following.

45 ⟨*KTH login procedure* 45⟩≡ (44a)

```
  response = self.ladok_run_shib_login(url)
  response = self.ug_post_user_pass(response)
  response = self.perform_redirects_back_to_ladok(response)
  return response
```



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.test.l...	redirect-erase.js	script	js	1.16 kB	936 B
200	GET	www.test.l...	branding.css	stylesheet	css	cached	2.0...
304	GET	service.sea...	thiss.js	script	js	cached	15...
200	GET	www.test.l...	larosate.svg	img	svg	1.23 kB	1.2...
200	GET	www.test.l...	eduid.svg	img	svg	1.03 kB	701 B
200	GET	www.test.l...	favicon-ladok.png	FaviconLoade...	png	cached	373 B
304	GET	service.sea...	/cta/	thiss.js:1 (sub...	html	cached	2.6...
304	GET	service.sea...	cta_7c2b16cbc25406b82a11.css	stylesheet	css	cached	81...
304	GET	service.sea...	vendors~cta~ds~index_03fc9693ef04190aae	script	js	cached	43....
304	GET	service.sea...	vendors~cta~index_a0082e3b154a08cc3222	script	js	cached	10...
304	GET	service.sea...	cta.js	script	js	cached	94....
304	GET	service.sea...	/ps/	subdocument	html	cached	225 B
304	GET	service.sea...	en.json	cta.js:5 (fetch)	json	cached	1.6...
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	33....
304	GET	service.sea...	ps.js	script	js	cached	17....
302	GET	www.test.l...	Login?target=https://www.test.ladok.se/gui/	document	html	6.94 kB	14....
200	GET	service.sea...	/ds/?entityID=https://www.test.ladok.se/gui-	document	html	6.84 kB	14....
304	GET	service.sea...	ds_d8f0b6bc046b0ef51c5c.css	stylesheet	css	cached	1.6...
304	GET	service.sea...	vendors~cta~ds~index_03fc9693ef04190aae	script	js	cached	43....
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	0 B
304	GET	service.sea...	ds.js	script	js	cached	19...
304	GET	service.sea...	en.json	ds.js:29 (fetch)	json	cached	1.6...
200	GET	md.seamle...	{sha1}4c7753e8c1b6129627738323df47194	vendors~cta~...	json	cached	1.0...
200	GET	service.sea...	/ps/	subdocument	html	690 B (raced)	225 B
404	GET	service.sea...	favicon.ico	FaviconLoade...	html	cached	169 B
304	GET	service.sea...	vendors~ds~ps_a41162e0d06688e07e88.js	script	js	cached	0 B
304	GET	service.sea...	ps.js	script	js	cached	17....
200	GET	md.seamle...	{sha1}e26e5d098f073536d0351577c98c838	ds.js:29 (fetch)	json	cached	906 B
304	GET	saml-5.sys....	logo.png	ds.js:25 (img)	png	cached	20....
302	GET	www.test.l...	Login?SAMLDS=1&target=ss:mem:5039d56e	vendors~cta~...	html	10.60 kB	9.2...
302	GET	saml-5.sys....	SSO?SAMLRequest=fZJRb4IwFIX/Cuk7FCtsS	document	html	9.58 kB	9.2...
200	GET	saml-5.sys....	SSO?execution=e3s1	document	html	9.54 kB	9.2...
200	GET	saml-5.sys....	consent.css	stylesheet	css	cached	2.4...
200	GET	www.test.l...	ladok_en.png	img	png	cached	857 B
302	GET	saml-5.sys....	idp	img	html	816 B	700 B
200	GET	saml-5.sys....	/idp/	img	html	812 B	700 B
200	GET	saml-5.sys....	favicon.ico	FaviconLoade...	x-icon	cached	1.1...
200	POST	saml-5.sys....	SSO?execution=e3s1	document	html	16.52 kB	16....
302	POST	www.test.l...	POST	SSO:1 (docum...	html	1.44 kB	1.0...
302	GET	www.test.l...	shiblogin	document	html	2.14 kB	1.0...
301	GET	www.test.l...	gui	document	html	922 B	1.0...
200	GET	www.test.l...	/gui/app/gui/	document	html	1.77 kB	1.0

91 requests | 9.93 MB / 1.74 MB transferred | Finish: 25.23 min | DOMContentLoaded: 82 ms | load: 91 ms

students and study administrative staff at higher education institutions in Sweden.

Additional information about the service

Information to be Provided to Service

Assurance level	**https://refeds.org/assurance/ID/unique** **https://refeds.org/assurance/IAP/low** **https://refeds.org/assurance/ID/eppn-unique-no-reassign** **https://refeds.org/assurance** **https://refeds.org/assurance/ATP/ePA-1m** **http://www.swamid.se/policy/assurance/al1**
Principal name	**dbosk@kth.se**
Scoped affiliation	**member@kth.se** **staff@kth.se** **employee@kth.se** **faculty@kth.se** **student@kth.se**
Given name	**Daniel**
Surname	**Bosk**

Data privacy information of the service

The information above would be shared with the service if you proceed. Do you agree to release this information to the service every time you access it?

Select an information release consent duration:

○ Ask me again at next login

- I agree to send my information this time.

[✓ Ask me again if information to be provided to this service changes]

- I agree that the same information will be sent automatically to this service in the future.

○ Do not ask me again

- I agree that **all** of my information will be released to **any** service.

This setting can be revoked at any time with the checkbox on the login page.

Figure 9.2: Firefox Developer Tools showing the requests when loggin in to LADOK through KTH for the first time. Pay attention to the page rendered on the right side, different from Fig. 9.1.

9.2 Running the request through Shibboleth

The first thing we want to do is to ask LADOK to run the Shibboleth login with the KTH SAML URL.

47a ⟨*KTH login methods* 47a⟩≡ (44a) 47b ▷

```
def ladok_run_shib_login(self, url):
  response = self.session.get(
    url=url+'&entityID=https://saml.sys.kth.se/idp/shibboleth')

  action = re.search(
    '<form ?[^>]* action="(.*?)"',
    response.text).group(1)

  csrf_token = re.search(
    '<input ?[^>]* name="csrf_token" ?[^>]* value="(.*?)"',
    response.text).group(1)

  post_data = {
    'csrf_token': csrf_token,
    'shib_idp_ls_exception.shib_idp_session_ss': '',
    'shib_idp_ls_success.shib_idp_session_ss': 'true',
    'shib_idp_ls_value.shib_idp_session_ss': '',
    'shib_idp_ls_exception.shib_idp_persistent_ss': '',
    'shib_idp_ls_success.shib_idp_persistent_ss': 'true',
    'shib_idp_ls_value.shib_idp_persistent_ss': '',
    'shib_idp_ls_supported': 'true',
    '_eventId_proceed': ''
  }

  response = self.session.post(
    url="https://saml-5.sys.kth.se" + action,
    data=post_data)

  return response
```

Note that in the last post request, it is important to use the 'saml-5' name and not the alias 'saml' (both resolve to the same address).

9.3 Posting username and password

After a series of redirects, we need to post our username and password to KTH UG login system. Note that we must add the suffix '@ug.kth.se' to the username.

47b ⟨*KTH login methods* 47a⟩+≡ (44a) ◁ 47a 48 ▷

```
def ug_post_user_pass(self, shib_response):
  action = re.search('<form ?[^>]* id="loginForm" ?[^>]* action="(.*?)"',
    shib_response.text).group(1)

  post_data = {
    'username': self.__username if "@" in self.__username \
                              else self.__username + "@ug.kth.se",
    'password': self.__password,
    'Kmsi': True,
    'AuthMethod': "FormsAuthentication"
  }

  response = self.session.post(
    url='https://login.ug.kth.se' + action,
    data=post_data)

  return response
```

9.4 Redirecting back to LADOK

Next, we must pass through a series of redirects. These are automated with JavaScript in the browser, we must simulate clicking the submit buttons when JavaScript is disabled. We do this by extracting the URL to post to and the values to include in the post.

48 ⟨*KTH login methods* 47a⟩+≡ (44a) ◁47b

```
def perform_redirects_back_to_ladok(self, ug_response):
  action = re.search('<form ?[^>]* action="(.*?)"',
    ug_response.text)
  if action is None:
    raise Exception('Invalid username or password OR possibly the SAML \
      configuration has changed, manually login an accept the changed \
      information.')
  action = html.unescape(action.group(1))

  relay_state = re.search(
    '<input ?[^>]* name="RelayState" ?[^>]* value="(.*?)"',
    ug_response.text)
  try:
    relay_state = html.unescape(relay_state.group(1))
  except AttributeError:
    raise Exception(
      "Try to log in using a web browser and accept sharing data.")

  saml_response = re.search(
    '<input ?[^>]* name="SAMLResponse" ?[^>]* value="(.*?)"',
    ug_response.text)
  saml_response = html.unescape(saml_response.group(1))

  post_data = {
      'RelayState': relay_state,
      'SAMLResponse': saml_response
  }

  response = self.session.post(url=action, data=post_data)
```

```
ladok_action = re.search(
  '<form ?[^>]* action="(.*?)"',
  response.text)
ladok_action = html.unescape(ladok_action.group(1))

relay_state = re.search(
  '<input ?[^>]* name="RelayState" ?[^>]* value="([^"]+)"',
  response.text)
relay_state = html.unescape(relay_state.group(1))

saml_response = re.search(
  '<input ?[^>]* name="SAMLResponse" ?[^>]* value="(.*?)"',
  response.text)
saml_response = html.unescape(saml_response.group(1))

post_data = {
    'RelayState': relay_state,
    'SAMLResponse': saml_response
}

response = self.session.post(url=ladok_action, data=post_data)

return response
```

Part III

API calls

Chapter 10

Overview of helper functions and methods

We will now document some possible API calls to LADOK. We also document the tests of these functions to illustrate their use.

10.1 HTTP queries to LADOK

To make things easier, we will add three methods: `get_query`, `put_query` and `post_query`, which are shortcuts to make GET, PUT and POST queries to LADOK.

51 ⟨*LadokSession data methods* 51⟩≡ 54a ▷

```
def get_query(self, path, content_type="application/vnd.ladok-resultat+json"):
  """Returns GET query response for path on the LADOK server"""
  headers = self.headers.copy()
  headers["Content-Type"] = content_type

  return self.session.get(
    url=self.base_gui_proxy_url + path,
    headers=headers)

def put_query(self, path, put_data,
  content_type="application/vnd.ladok-resultat+json"):
  """Returns PUT query response for path on the LADOK server"""
  headers = self.headers.copy()
  headers["Content-Type"] = content_type
  headers["X-XSRF-TOKEN"] = self.get_xsrf_token()
  headers["Referer"] = self.base_gui_url

  return self.session.put(
    url=self.base_gui_proxy_url + path,
    json=put_data,
    headers=headers)

def post_query(self, path, post_data,
  content_type="application/vnd.ladok-resultat+json"):
  """Returns POST query response for path on the LADOK server"""
  headers = self.headers.copy()
  headers["Content-Type"] = content_type
  headers["X-XSRF-TOKEN"] = self.get_xsrf_token()
  headers["Referer"] = self.base_gui_url

  return self.session.post(
    url=self.base_gui_proxy_url + path,
    json=post_data,
    headers=headers)
```

10.2 Cleaning data for printing

We sometimes want to print the data, for instance, example output in this document. For this reason we introduce some cleaning functions. These recursively transcends the JSON structure removing the data that should be removed.

52a ⟨*functions* 52a⟩≡ 52b▷

```
  def clean_data(json_obj):
    remove_links(json_obj)
    pseudonymize(json_obj)
```

The `remove_links` functions removes the `link` key–value pairs. The `link` values contains URLs for all requests that data are based on.

52b ⟨*functions* 52a⟩+≡ ◁52a 52c▷

```
  def remove_links(json_obj):
    """Recursively removes all "link" keys and values"""
    if isinstance(json_obj, dict):
      if "link" in json_obj:
        json_obj.pop("link")
      for key, value in json_obj.items():
        remove_links(value)
    elif isinstance(json_obj, list):
      for item in json_obj:
        remove_links(item)
```

The `pseudonymize` function replaces names and personnummer with dummy entries.

52c ⟨*functions* 52a⟩+≡ ◁52b

```
  def pseudonymize(json_obj):
    """Recursively pseudonymizes a JSON data record"""
    if isinstance(json_obj, dict):
      if "Fornamn" in json_obj:
        json_obj["Fornamn"] = "Student"
      if "Efternamn" in json_obj:
        json_obj["Efternamn"] = "Studentsson"
      if "Personnummer" in json_obj:
        json_obj["Personnummer"] = "191234561234"
      for key, value in json_obj.items():
        pseudonymize(value)
    elif isinstance(json_obj, list):
      for item in json_obj:
        pseudonymize(item)
```

10.3 Example code and test code for the API

We will provide both example code and test code. The example code is so that we can show example outputs. The test code is the test cases we want to be able to run with `pytest` later.

The example code will be typeset with syntax highlighting. It will all have the following block of code included.

```python
import json
import ladok3.kth
import os

ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments
```

We will use the following to test the API methods.

53 ⟨*test api.py* 53⟩≡

```
import json
import ladok3.kth
import os

ladok = ladok3.kth.LadokSession(
        os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
        test_environment=True) # for experiments

student_uid = "de709f81-a867-11e7-8dbf-78e86dc2470c"
dasak_instance_id = "39c56d6a-73d8-11e8-b4e0-063f9afb40e3"
dasak_education_id = "39cf7fb4-73d8-11e8-afa7-8e408e694e54"
dasak_round_id = "79b4b213-73da-11e8-b4e0-063f9afb40e3"
LAB1_uid = "39ca27fe-73d8-11e8-afa7-8e408e694e54"
LAB1_instance_id = "39ca2861-73d8-11e8-b4e0-063f9afb40e3"
KTH_org_id = "2474f616-dc41-11e8-8cc1-eaeeb71b497f"
```

⟨*test functions* 54b⟩

We will then add `test_X()` functions in the ⟨*test functions* 54b⟩ code block.

Chapter 11

Grade-related API calls

11.1 grade_scales_JSON

To request the grading scales from LADOK, we request all of them and return a list of JSON data objects containing the grading scale data.

54a ⟨*LadokSession data methods* 51⟩+≡ ◁51 55▷

```
def grade_scales_JSON(self):
  response = self.get_query('/resultat/grunddata/betygsskala')

  if response.status_code == 200:
    return response.json()["Betygsskala"]
  return None
```

We add the following test. If we can convert the return value do JSON, it's probably correct. (No this isn't the best of tests.)

54b ⟨*test functions* 54b⟩≡ (53) 56a▷

```
def test_grade_scales_JSON():
  assert json.dumps(ladok.grade_scales_JSON()[0], indent=2, ensure_ascii=False)
```

This method is used as follows.

```
print(json.dumps(ladok.grade_scales_JSON()[0], indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

Chapter 12

Student-related API calls

Here we have collected API-calls related to students.

12.1 `get_student_data_JSON` and `get_student_data_by_uid_JSON`

This method fetches the basic information about a student based on personnummer. To pull the data based on personnummer we must actually form a search query. (This code is a merge of a slight adaptation of the code by Baltatzis and that of Chip.)

55 ⟨*LadokSession data methods* 51⟩+≡ ◁54a 56b▷

```
######################################################################
#
# get_student_data_JSON
#
# person_nr          - personnummer, flera format accepteras enligt regex:
#                      (\d\d)?(\d\d)(\d\d\d\d)[+\-]?(\w\w\w\w)
#
# lang               - language code 'en' or 'sv', defaults to 'sv'
#
# RETURNERAR en dictionary med för- och efternamn and more
def get_student_data_JSON(self, person_nr_raw, lang = 'sv'):
  person_nr =  format_personnummer(person_nr_raw)

  if not person_nr: raise Exception('Invalid person nr ' + person_nr_raw)

  response = self.session.get(
    url=self.base_gui_proxy_url +
      '/studentinformation/student/filtrera?limit=2&orderby=EFTERNAMN_ASC&orderby=FORNAM
        + person_nr + '&skipCount=false&sprakkod='+lang,
    headers=self.headers)

  if response.status_code == requests.codes.ok:
    record = response.json()["Resultat"]
  else:
    raise ValueError(
      f"can’t find student based on personnummer {person_nr_raw}")

  if len(record) != 1:
    raise ValueError(
      f"can’t find student based on personnummer {person_nr_raw}")

  return record[0]
```

To test this function, we do the following.

56a ⟨*test functions* 54b⟩+≡ (53) ◁54b 56c▷

```
  def test_get_student_data_JSON():
    assert ladok.get_student_data_JSON("8506097891")
```

We also have the corresponding for LADOK's UID, which fetches the record directly.

56b ⟨*LadokSession data methods* 51⟩+≡ ◁55 56d▷

```
  ########################################################################
  #
  # get_student_data_by_uid_JSON
  #
  # uid                - Ladok ID
  #
  # RETURNERAR en dictionary med för- och efternamn and more
  def get_student_data_by_uid_JSON(self, uid):
    response = self.session.get(
      url = self.base_gui_proxy_url +
        '/studentinformation/student/'+uid, headers = self.headers)
    if response.status_code == requests.codes.ok:
      return response.json()
    raise AttributeError(f"can’t fetch student attributes by LADOK ID {uid}")
```

To test this function, we do the following.

56c ⟨*test functions* 54b⟩+≡ (53) ◁56a 57a▷

```
  def test_get_student_data_by_uid_JSON():
    assert ladok.get_student_data_by_uid_JSON(student_uid)
```

12.2 get_student_contact_data_JSON

We want to get the contact info for a student from LADOK. This data includes email, postal address and phone number. It also includes when they were last updated.

56d ⟨*LadokSession data methods* 51⟩+≡ ◁56b 57b▷

```
  def get_student_contact_data_JSON(self, student_id):
    """Returns contact data for student with student_id, returns JSON"""
    response = self.get_query(
      f"/studentinformation/internal/student/{student_id}/kontaktuppgifter",
      "application/vnd.ladok-studentinformation+json")

    if response.status_code == requests.codes.ok:
      return response.json()
    return None
```

We test this function.

57a ⟨*test functions* 54b⟩+≡ (53) ◁56c 57c▷

```
  def test_get_student_contact_data_JSON():
    assert ladok.get_student_contact_data_JSON(student_uid)
```

12.3 get_student_suspensions_JSON

We want to check if any student is suspended from studies. We can get a list of all suspension periods from LADOK.

57b ⟨*LadokSession data methods* 51⟩+≡ ◁56d 57d▷

```
  def get_student_suspensions_JSON(self, student_id):
    """
    Returns suspensions from studies for student with student_id,
    returns JSON
    """
    response = self.get_query(
      f"/studentinformation/internal/avstangning/student/{student_id}",
      "application/vnd.ladok-studentinformation+json")

    if response.status_code == requests.codes.ok:
      return response.json()
    return None
```

We test this function.

57c ⟨*test functions* 54b⟩+≡ (53) ◁57a 58a▷

```
  def test_get_student_suspensions_JSON():
    assert ladok.get_student_suspensions_JSON(student_uid)
```

12.4 registrations_JSON

This methods returns *all* registrations for a student, i.e., registrations on courses and programmes.

57d ⟨*LadokSession data methods* 51⟩+≡ ◁57b 58b▷

```
  def registrations_JSON(self, student_id):
    """Return all registrations for student with ID student_id."""
    response = self.get_query(
      '/studiedeltagande/tillfallesdeltagande/kurstillfallesdeltagande/student/'+
        student_id,
      "application/vnd.ladok-studiedeltagande+json")

    if response.status_code == 200:
      return response.json()["Tillfallesdeltaganden"]
    return None
```

We provide the following test.

58a ⟨*test functions* 54b⟩+≡ (53) ◁57c 58c▷

```
def test_registrations_JSON():
  results = ladok.registrations_JSON(student_uid)
  assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")

results = ladok.registrations_JSON(me.ladok_id)

ladok3.clean_data(results)
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

12.5 registrations_on_course_JSON

This method returns all registrations for a particular course for a particular student. This way we can check if a student has been registered several times on a course.

58b ⟨*LadokSession data methods* 51⟩+≡ ◁57d 60a▷

```
def registrations_on_course_JSON(self,
    course_education_id, student_id):
  """Return a list of registrations on course with education_id for student
  with student_id. JSON format."""
  response = self.get_query(
    "/studiedeltagande/tillfallesdeltagande"
      f"/utbildning/{course_education_id}/student/{student_id}",
    "application/vnd.ladok-studiedeltagande+json")

  if response.status_code == 200:
    return response.json()["Tillfallesdeltaganden"]
  return None
```

We add the following test.

58c ⟨*test functions* 54b⟩+≡ (53) ◁58a 60b▷

```
def test_registrations_on_course_JSON():
  results = ladok.registrations_on_course_JSON(dasak_education_id, student_uid)
  assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")
dasak = me.courses(code="DD2395")[0]

results = ladok.registrations_on_course_JSON(dasak.education_id,
  me.ladok_id)

ladok3.clean_data(results)
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

Chapter 13

Course-related API calls

13.1 course_rounds_JSON

This method fetches all course rounds that uses the given course instance.

60a ⟨*LadokSession data methods* 51⟩+≡ ◁58b 60c▷
```
  def course_rounds_JSON(self, course_instance_id):
    """Requires course instance ID"""
    response = self.get_query(
      f"/resultat/kurstillfalle/kursinstans/{course_instance_id}")

    if response.status_code == 200:
      return response.json()["Utbildningstillfalle"]
    return None
```

We add the following test.

60b ⟨*test functions* 54b⟩+≡ (53) ◁58c 61a▷
```
  def test_course_rounds_JSON():
    results = ladok.course_rounds_JSON(dasak_instance_id)
    assert json.dumps(results[:1]+results[-1:], indent=2, ensure_ascii=False)
```

This method is used as follows.

```
dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
results = ladok.course_rounds_JSON(dasak10.instance_id)

ladok3.clean_data(results)
print(json.dumps(results[:1]+results[-1:], indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

13.2 course_instance_JSON

This method fetches the data for a given course instance. It requires the course instance ID. (This is a slightly rewritten version of Maguire's original method.)

60c ⟨*LadokSession data methods* 51⟩+≡ ◁60a 61b▷
```
  def course_instance_JSON(self, instance_id):
    """Returns course instance data for a course with instance ID instance_id"""
    response = self.get_query(
      f"/resultat/utbildningsinstans/kursinstans/{instance_id}")

    if response.status_code == 200:
      return response.json()
    return None
```

We add the following test.

61a ⟨*test functions* 54b⟩+≡ (53) ◁60b 61c▷

```
  def test_course_instance_JSON():
    results = ladok.course_instance_JSON(dasak_instance_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
results = ladok.course_instance_JSON(dasak10.instance_id)

ladok3.clean_data(results)
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

13.3 Course components

There are two ways to get the components for a course.

13.3.1 course_round_components_JSON

This method fetches the course components of a course round from LADOK. It requires the course round ID. This one includes data such as the number of registered students as well, unlike the method in the next section.

61b ⟨*LadokSession data methods* 51⟩+≡ ◁60c 62a▷

```
  def course_round_components_JSON(self, round_id):
    response = self.put_query(
      "/resultat/kurstillfalle/moment",
      {"Identitet": [round_id]}
    )

    if response.status_code == 200:
      return response.json()["MomentPerKurstillfallen"]
    raise Exception(response.json()["Meddelande"])
```

We add the following test.

61c ⟨*test functions* 54b⟩+≡ (53) ◁61a 62b▷

```
  def test_course_round_components_JSON():
    components = ladok.course_round_components_JSON(dasak_round_id)
    assert json.dumps(components, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
try:
  components = ladok.course_round_components_JSON(dasak10.round_id)
except Exception as err:
  print(f"error: {err}")
else:
  ladok3.clean_data(components)
  print(json.dumps(components, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

13.3.2 course_instance_components_JSON

This method fetches the course components for a course instance, i.e., a version of the syllabus.

62a ⟨*LadokSession data methods* 51⟩+≡ ◁61b 63▷

```
  def course_instance_components_JSON(self, course_instance_id):
    response = self.put_query(
      "/resultat/utbildningsinstans/moduler",
      {"Identitet": [course_instance_id]}
    )

    if response.status_code == 200:
      return response.json()["Utbildningsinstans"][0]
    raise Exception(response.json()["Meddelande"])
```

We add the following test code.

62b ⟨*test functions* 54b⟩+≡ (53) ◁61c 64a▷

```
  def test_course_instance_components_JSON():
    components = ladok.course_instance_components_JSON(dasak_instance_id)
    assert json.dumps(components, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
try:
  components = ladok.course_instance_components_JSON(dasak10.instance_id)
except Exception as err:
  print(f"error: {err}")
else:
  ladok3.clean_data(components)
  print(json.dumps(components, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

Chapter 14

Results-related API calls

In this chapter we look at how to fetch results from LADOK and report new results to LADOK.

14.1 Reported results

There are two ways to get results for a course. One method gives more data than the other.

14.1.1 search_reported_results_JSON

This method searches for student results for a given component on a given course round.

63 ⟨*LadokSession data methods* 51⟩+≡ ◁62a 64b▷

```
def search_reported_results_JSON(self, course_round_id, component_instance_id):
  """Requires:
  course_round_id: round_id for a course,
  component_instance_id: instance_id for a component of the course.
  """
  put_data = {
    "Filtrering": ["OBEHANDLADE", "UTKAST", "ATTESTERADE"],
    "KurstillfallenUID": [course_round_id],
    "OrderBy": [
      "EFTERNAMN_ASC",
      "FORNAMN_ASC",
      "PERSONNUMMER_ASC"
    ],
    "Limit": 400,
    "Page": 1,
    "StudenterUID": []
  }

  response = self.put_query(
    '/resultat/studieresultat/rapportera/utbildningsinstans/' +
      component_instance_id + '/sok',
    put_data)

  if response.status_code == 200:
    return response.json()["Resultat"]
  return None
```

We write the following test.

64a ⟨*test functions* 54b⟩+≡ (53) ◁62b 64c▷

```
  def test_search_reported_results_JSON():
    results = ladok.search_reported_results_JSON(dasak_round_id,
      LAB1_instance_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
LAB1 = dasak10.components(code="LAB1")[0]

results = ladok.search_reported_results_JSON(dasak10.round_id, LAB1.instance_id)

ladok3.clean_data(results)
results = list(filter(
  lambda x: x["Student"]["Uid"] == me.ladok_id,
  results))
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

14.1.2 search_course_results_JSON

Another method, which gives slightly different results is the following.

64b ⟨*LadokSession data methods* 51⟩+≡ ◁63 65a▷

```
  def search_course_results_JSON(self, course_round_id, component_instance_id):
    put_data = {
      "KurstillfallenUID": [course_round_id],
      "Tillstand": ["REGISTRERAD", "AVKLARAD", "AVBROTT"],
      "OrderBy": ["EFTERNAMN_ASC", "FORNAMN_ASC"],
      "Limit": 400,
      "Page": 1,
    }

    response = self.put_query(
      "/resultat/resultatuppfoljning/resultatuppfoljning/sok",
      put_data)

    if response.status_code == 200:
      return response.json()["Resultat"]
    return None
```

We test this by the following.

64c ⟨*test functions* 54b⟩+≡ (53) ◁64a 65b▷

```
  def test_search_course_results_JSON():
    results = ladok.search_course_results_JSON(dasak_round_id, LAB1_instance_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
LAB1 = dasak10.components(code="LAB1")[0]

results = ladok.search_course_results_JSON(dasak10.round_id, LAB1.instance_id)

ladok3.clean_data(results)
results = list(filter(
  lambda x: x["Student"]["Uid"] == me.ladok_id,
  results))
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

14.2 Results for a student: `student_results_JSON`

This method pulls results for an individual student for a particular course. LADOK changed this API request in 2022.

65a ⟨*LadokSession data methods* 51⟩+≡ ◁64b 66a▷

```
  def student_results_JSON(self, student_id, course_education_id):
    """Returns the results for a student on a course"""
    response = self.get_query(
      "/resultat/internal/studentenskurser/kursinformation"
      f"/student/{student_id}/kursUID/{course_education_id}"
    )

    if response.status_code == requests.codes.ok:
      return response.json()
    raise Exception(response.json()["Meddelande"])
```

We test this in the following way.

65b ⟨*test functions* 54b⟩+≡ (53) ◁64c 67b▷

```
  def test_student_results_JSON():
    results = ladok.student_results_JSON(student_uid, dasak_education_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
results = ladok.student_results_JSON(me.ladok_id, dasak10.round_id)

ladok3.clean_data(results)
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

14.3 Modifying results

We have two alternatives: add a new result or update an existing result draft.

14.3.1 `create_result_JSON`

This method is used to create a new result. This result will be a draft. It must later be finalized and then attested.

Note that since this is a new result, we must provide the `StudieresultatUID` where we want to add the result. When we've done this, we'll get a `ResultatUID` back in the response. From then on, it's the `ResultatUID` that's interesting.

66a ⟨*LadokSession data methods* 51⟩+≡ ◁65a 66b▷

```
def create_result_JSON(self,
      grade_id, grade_scale_id, date,
      study_result_id, instance_id,
      notes=[]):
  """Creates a new result"""
  response = self.post_query(
    "/resultat/studieresultat/skapany",
    [{"Resultat": [{
      "Betygsgrad": grade_id,
      "BetygsskalaID": grade_scale_id,
      "Examinationsdatum": date,
      "Noteringar": notes,
      "StudieresultatUID": study_result_id,
      "UtbildningsinstansUID": instance_id
    }]}]
  )

  if response.status_code == requests.codes.ok:
    return response.json()["Studieresultat"]
  raise Exception(response.json()["Meddelande"])
```

LADOK changed this API request in 2022.

14.3.2 `update_result_JSON`

This method updates an existing result draft. Note that we cannot use this method to update a finalized result. Note also that we use the `ResultatUID` and not the `StudieresultatUID` as we did for `create_result_JSON`.

66b ⟨*LadokSession data methods* 51⟩+≡ ◁66a 67a▷

```
def update_result_JSON(self,
      grade_id, grade_scale_id, date,
      result_id, last_modified, notes=[]):
  response = self.put_query(
    '/resultat/studieresultat/uppdatera',
    {
      'Resultat': [{
        'ResultatUID': result_id,
        'Betygsgrad': grade_id,
        'BetygsskalaID': grade_scale_id,
        'Noteringar': notes,
        'Examinationsdatum': date,
        'SenasteResultatandring': last_modified
      }]
    }
  )

  if response.status_code == 200:
    return response.json()["Resultat"]
  raise Exception(response.json()["Meddelande"])
```

14.4 Finalizing a result

Here we cover the API calls needed to finalize (klarmarkera) a result in LADOK.

14.4.1 `result_attestants_JSON` and `result_reporters_JSON`

To finalize a result, we must know two things: who is reporting and who can attest. We start with who can attest.

67a ⟨*LadokSession data methods* 51⟩+≡ ◁66b 68a▷

```
def result_attestants_JSON(self, result_id):
  """Returns a list of result attestants"""
  response = self.put_query(
    "/resultat/internal/anvandare/resultatrattighet/attestanter/kurstillfallesrapporteri
    {"Identitet": [result_id]}
  )

  if response.status_code == 200:
    return response.json()["Anvandare"]
  raise Exception(response.json()["Meddelande"])
```

The `result_id` is the ID returned in the `ResultatUID` field in the response from the `create_result_JSON` method.

LADOK changed this API request in 2022: the `internal` is mandatory part of the path.

We can test this in the following way.

67b ⟨*test functions* 54b⟩+≡ (53) ◁65b 68b▷

```
def test_result_attestants_JSON():
  attestants = ladok.result_attestants_JSON(
    "319558de-55c6-11ed-b4c5-e36d0ff64f20")
  assert json.dumps(attestants, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
attestants = ladok.result_attestants_JSON(
  "319558de-55c6-11ed-b4c5-e36d0ff64f20")

print(json.dumps(attestants[0], indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

Now, we get a list of who can report (basically anyone registered in the entire organization).

68a ⟨*LadokSession data methods* 51⟩+≡ ◁67a 68c▷

```
def result_reporters_JSON(self, organization_id):
  """Returns a list of who can report results in an organization"""
  response = self.get_query(
    "/kataloginformation/anvandare/organisation/" +
      organization_id + "/resultatrapportorer",
    "application/vnd.ladok-kataloginformation+json"
  )

  if response.status_code == 200:
    return response.json()["Anvandare"]
  raise Exception(response.text)
```

We can test this as follows.

68b ⟨*test functions* 54b⟩+≡ (53) ◁67b 68d▷

```
def test_result_reporters_JSON():
  reporters = ladok.result_reporters_JSON(KTH_org_id)
  assert json.dumps(reporters, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
reporters = ladok.result_reporters_JSON(components["OrganisationUID"])
ladok3.remove_links(reporters)

print(json.dumps(reporters[8], indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

14.4.2 user_info_JSON

Usually, we want to set the reporter to the logged-in user. We can use the following API call to get information about the logged-in user.

68c ⟨*LadokSession data methods* 51⟩+≡ ◁68a 69a▷

```
def user_info_JSON(self):
  response = self.get_query(
    "/kataloginformation/anvandare/anvandarinformation",
    "application/vnd.ladok-kataloginformation+json"
  )

  if response.status_code == 200:
    return response.json()
  raise Exception(response.text)
```

We test this as follows.

68d ⟨*test functions* 54b⟩+≡ (53) ◁68b 70▷

```
def test_user_info_JSON():
  me_teacher = ladok.user_info_JSON()
  assert json.dumps(me_teacher, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
me_teacher = ladok.user_info_JSON()
ladok3.remove_links(me_teacher)

print(json.dumps(me_teacher, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

14.4.3 finalize_result_JSON

Finally, we can finalize the reported grade. If `attestant_id` is not `None`, then LADOK will send a notification to that person. (LADOK changed this API request in 2022.)

69a ⟨*LadokSession data methods* 51⟩+≡ ◁68c 69b▷

```
  def finalize_result_JSON(self,
      result_id, last_modified, reporter_id, attestant_id=None):
    """Marks a result as finalized (klarmarkera)"""
    response = self.put_query(
      f"/resultat/studieresultat/resultat/klarmarkerany",
      {
        "Klarmarkering": {
          "Beslutsfattare": [attestant_id] if attestant_id else [],
          "RattadAv": [reporter_id],
          "ResultatetsSenastSparad": last_modified
        },
        "ResultatUIDer": [result_id]
      }
    )

    if response.status_code == requests.codes.ok:
      return response.json()
    raise Exception(response.json()["Meddelande"])
```

This method returns a copy of the finalized result.

14.5 participants_JSON

The method returns JSON data containing a list of students (the participants in the course round). (This is an extension of Maguire's original `participants_JSON` method. The essential difference is keyword arguments to filter which students to include.)

69b ⟨*LadokSession data methods* 51⟩+≡ ◁69a

```
  def participants_JSON(self, course_round_id, /, **kwargs):
    """Returns JSON record containing participants in a course identified by
    round ID.
    Filters in kwargs: not_started, ongoing, registered, finished, cancelled"""
    participants_types = []
    if "not_started" in kwargs and kwargs["not_started"]:
      participants_types.append("EJ_PABORJAD")
    if "ongoing" in kwargs and kwargs["ongoing"]:
      participants_types.append("PAGAENDE")
    if "registered" in kwargs and kwargs["registered"]:
      participants_types.append("REGISTRERAD")
    if "finished" in kwargs and kwargs["finished"]:
      participants_types.append("AVKLARAD")
    if "cancelled" in kwargs and kwargs["cancelled"]:
      participants_types.append("AVBROTT")
    # 'ATERBUD', # Withdrawal
    # 'PAGAENDE_MED_SPARR', # on-going block exists
    # 'EJ_PAGAENDE_TILLFALLESBYTE', # not on-going due to instance exchange
    # 'UPPEHALL', # not on-going due to approved leave from studies

    if not kwargs:
      participants_types = ["PAGAENDE", "REGISTRERAD", "AVKLARAD"]

    put_data = {
      'page': 1,
      'limit': 400,
      'orderby': ['EFTERNAMN_ASC',
                  'FORNAMN_ASC',
                  'PERSONNUMMER_ASC',
                  'KONTROLLERAD_KURS_ASC'],
      'deltagaretillstand': participants_types,
      'utbildningstillfalleUID': [course_round_id]
    }

    response = self.put_query(
      '/studiedeltagande/deltagare/kurstillfalle',
      put_data,
      "application/vnd.ladok-studiedeltagande+json")
    if response.status_code == 200:
      return response.json()["Resultat"]
    return None
```

We test this as follows.

70 ⟨*test functions* 54b⟩+≡ (53) ◁68d

```
  def test_participants_JSON():
    results = ladok.participants_JSON(dasak_round_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
results = ladok.participants_JSON(dasak10.round_id)

ladok3.clean_data(results)
results = list(filter(
  lambda x: x["Student"]["Uid"] == me.ladok_id,
  results))
print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. **?? PythonTeX ??**

Chapter 15

Baltatzis' and Maguire's original `LadokSession` methods

Below are the original `LadokSession` methods by Baltatzis and Maguire. The one missing is the constructor, that has been replaced by the code above. Also, the code has been adapted to use `self.session` instead of `self.__session` etc.

15.1 `get_results`, `save_result`

71 ⟨*LadokSession data methods* 71⟩≡ 75▷

```
##################################################################
#
# LadokSession
#
# get_results      returnerar en dictionary med momentnamn och resultat
# save_result      sparar resultat för en student som utkast
#
# The original LadokSession code is from Alexander Baltatzis <alba@kth.se> on
# 2020-07-20
#
# I (Gerald Q. Maguire Jr.) have extended on 2020-07-21 and later with the code
# as noted below.
#
# I (Daniel Bosk) adapted (on 2021-01-08) the methods to a refactored
# LadokSession class.

#########################################################################
#
# get_results
#
# person_nr          - personnummer, siffror i strängformat
#            t.ex. 19461212-1212
# course_code          - kurskod t.ex. DD1321
#
```

```
# RETURNERAR en dictionary från ladok med momentnamn, resultat
#
# {'LABP': {'date': '2019-01-14', 'grade': 'P', 'status': 'attested'},
#  'LABD': {'date': '2019-03-23', 'grade': 'E', 'status': 'pending(1)'},
#  'TEN1': {'date': '2019-03-13', 'grade': 'F', 'status': 'pending(2)'}}
#
#  status:  kan ha följande värden vilket gissningsvis betyder:
#           attested   - attesterad
#           pending(1) - utkast
#           pending(2) - klarmarkerad
#
def get_results(self, person_nr_raw, course_code):
  person_nr_raw = str(person_nr_raw)
  person_nr =  format_personnummer(person_nr_raw)
  if not person_nr: raise Exception('Invalid person nr ' + person_nr_raw)

  student_data = self.__get_student_data(person_nr)

  student_course = next(x
    for x in self.__get_student_courses(student_data['id'])
      if x['code'] == course_code)

  # get attested results
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/resultat/studentresultat/attesterade/student/' +
        student_data['id'],
    headers=self.headers).json()

  results_attested_current_course = None
  results = {}  # return value

  for course in r['StudentresultatPerKurs']:
    if course['KursUID'] == student_course['education_id']:
      results_attested_current_course = course['Studentresultat']
      break


  if results_attested_current_course:
    for result in results_attested_current_course:
      try:
          d = { 'grade' : result['Betygsgradskod'],
                'status': 'attested',
                'date'  : result['Examinationsdatum'] }
          results[ result['Utbildningskod'] ] = d
      except:
          pass  # tillgodoräknanden har inga betyg och då är result['Utbildningskod'] ==

  # get pending results
  r = self.session.get(
    url=self.base_gui_proxy_url + '/resultat/resultat/resultat/student/' +
      student_data['id'] + '/kurs/' + student_course['education_id'] +
        '?resultatstatus=UTKAST&resultatstatus=KLARMARKERAT',
    headers=self.headers).json()

  for result in r['Resultat']:
      r = self.session.get(
        url=self.base_gui_proxy_url + '/resultat/utbildningsinstans/' +
          result['UtbildningsinstansUID'],
        headers=self.headers).json()
      d_grade = result['Betygsgradsobjekt']['Kod']
      d_status = "pending(" + str(result['ProcessStatus']) + ")"
      # utkast har inte datum tydligen ...
      d_date = "0" if 'Examinationsdatum' not in result \
                    else result['Examinationsdatum']
      d = { 'grade' : d_grade ,
            'status': d_status,
            'date'  : d_date      }
      results[ r['Utbildningskod'] ] = d
  return results

######################################################################
#
# save_result
#
# person_nr          - personnummer, flera format accepteras enligt regex:
#                      (\d\d)?(\d\d)(\d\d\d\d)[+\-]?(\w\w\w\w)
# course_code        - kurskod t.ex. DD1321
# course_moment      - ladokmoment/kursbetyg t.ex. TEN1, LAB1, DD1321 (!)
#                      om labmomententet är samma som course_code så sätts kursbetyg!
# result_date        - betygsdatum, flera format accepteras enligt regex
#                      (\d\d)?(\d\d)-?(\d\d)-?(\d\d)
# grade_code         - det betyg som ska sättas
# grade_scale        - betygsskala t.ex. AF eller PF. Möjliga betygsskalor
#                      listas i self.__grade_scales.
#
# RETURNERAR True om det gått bra, kastar (förhoppningsvis) undantag
#            om det går dåligt.
def save_result(self, person_nr_raw, course_code, course_moment,
  result_date_raw, grade_raw, grade_scale):
  if grade_raw in ["AF", "PF"]:
      raise Exception('Invalid grade: ' + grade_raw + ' looks like a grade_scale')

  if (grade_raw == 'P' and grade_scale == "AF") or \
     (grade_raw in "ABCDE" and grade_scale == "PF"):
    raise Exception('Invalid grade: ' + grade_raw +
      ' does not match grade_scale ' + grade_scale)

  person_nr =  format_personnummer(person_nr_raw)
  if not person_nr: raise Exception('Invalid person nr ' + person_nr_raw)
```

```
result_date = self.__validate_date(result_date_raw)
if not result_date:
  raise Exception('Invalid grade date: ' + result_date_raw + ' pnr: ' +
    person_nr_raw + ' moment: ' + course_moment)

student_data = self.__get_student_data(person_nr)
student_course = next(x
  for x in self.__get_student_courses(student_data['id'])
    if x['code'] == course_code)

# momentkod = kurskod => vi hanterar kursbetyg
if course_moment == student_course['code']:
    course_moment_id = student_course['instance_id']
else:
    for x in self.__get_student_course_moments(student_course['round_id'],
      student_data['id']):
      if x['code'] == course_moment:
        course_moment_id = x['course_moment_id']

student_course_results = self.__get_student_course_results(
  student_course['round_id'], student_data['id'])

grade_scale = self.__get_grade_scale_by_code(grade_scale)
grade = grade_scale.grades(code=grade_raw)[0]

headers = self.headers.copy()
headers['Content-Type'] = 'application/vnd.ladok-resultat+json'
headers['X-XSRF-TOKEN'] = self.__get_xsrf_token()
headers['Referer'] = self.base_gui_url

previous_result = None

for result in student_course_results['results']:
    if result['pending'] is not None:
        if result['pending']['moment_id'] == course_moment_id:
            previous_result = result['pending']
            break

# uppdatera befintligt utkast
if previous_result:
    put_data = {
        'Resultat': [{
            'ResultatUID': previous_result['id'],
            'Betygsgrad': grade.id,
            'Noteringar': [],
            'BetygsskalaID': grade_scale.id,
            'Examinationsdatum': result_date,
            'SenasteResultatandring': previous_result['last_modified']
        }]
    }

    r = self.session.put(
      url=self.base_gui_proxy_url + '/resultat/studieresultat/uppdatera',
      json=put_data,
      headers=headers)

# lägg in nytt betygsutkast
else:
    post_data = {
        'Resultat': [{
            'StudieresultatUID': student_course_results['id'],
            'UtbildningsinstansUID': course_moment_id,
            'Betygsgrad': grade.id,
            'Noteringar': [],
            'BetygsskalaID': grade_scale.id,
            'Examinationsdatum': result_date
        }]
    }
    r = self.session.post(
      url=self.base_gui_proxy_url + '/resultat/studieresultat/skapa',
      json=post_data,
      headers=headers)

if not 'Resultat' in r.json():
  raise Exception("Couldn't register " +
    course_moment + "=" + grade_raw + " " + result_date_raw + ": " +
      r.json()["Meddelande"])

return True
```

15.2 get_student_data, get_student_name

75 ⟨*LadokSession data methods* 71⟩+≡ ◁71 76a▷

```
  ######################################################################
  #
  # get_student_data
  #
  # person_nr          - personnummer, flera format accepteras enligt regex:
  #                        (\d\d)?(\d\d)(\d\d\d\d)[+\-]?(\w\w\w\w)
  #
  # RETURNERAR {'id': 'xxxx', 'first_name': 'x', 'last_name': 'y', 'person_nr': 'xxx', 'al

  def get_student_data(self, person_nr_raw):
    person_nr =  format_personnummer(person_nr_raw)

    if not person_nr: raise Exception('Invalid person nr ' + person_nr_raw)

    student_data = self.__get_student_data(person_nr)
    return student_data

  #####################################################################
  #
  # get_student_name
  #
  # person_nr          - personnummer, flera format accepteras enligt regex:
  #                      (\d\d)?(\d\d)(\d\d\d\d)[+\-]?(\w\w\w\w)
  #
  # RETURNERAR en dictionary med för- och efternamn
  #
  # {"first_name" : 'Anna', "last_name : 'Andersson'}
  #
  def get_student_name(self, person_nr_raw):
    person_nr =  format_personnummer(person_nr_raw)

    if not person_nr: raise Exception('Invalid person nr ' + person_nr_raw)

    student_data = self.__get_student_data(person_nr)
    return {
      "first_name": student_data["first_name"],
      "last_name" : student_data["last_name"]
    }
```

15.3 all_grading_scale

76a ⟨*LadokSession data methods* 71⟩+≡ ◁75 76b▷

```
  # added by GQMJr
  #####################################################################
  #
  # all_grading_scale
  #
  #
  # RETURNERAR en dictionary of the grading scales
  def all_grading_scale(self):
    return self.get_grade_scales()
```

15.4 grading_rights

76b ⟨*LadokSession data methods* 71⟩+≡ ◁76a 77a▷

```
  # added by GQMJr
  #####################################################################
  #
  # grading_rights
  #
  #
  # RETURNERAR en dictionary of the grading rights (of the logged in user)
```

```
def grading_rights(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/resultat/resultatrattighet/listaforinloggadanvandare',
    headers=self.headers).json()
  return r['Resultatrattighet']
```

15.5 change_locale

77a ⟨*LadokSession data methods* 71⟩+≡ ◁76b 77b▷

```
# added by GQMJr
#####################################################################
#
# change_locale
#
# lang              - language code 'en' or 'sv', defaults to 'sv'
#
# RETURNERAR reponse to the request
def change_locale(self, lang = 'sv'):
  r = self.session.get(
    url=self.base_gui_url+'/services/i18n/changeLocale?lang='+lang,
    headers=self.headers).json()
  return r
```

15.6 course_instances_JSON

77b ⟨*LadokSession data methods* 71⟩+≡ ◁77a 78a▷

```
# added by GQMJr
#####################################################################
#
# course_instances_JSON
#
# course_code       - course code, such as "II2202"
#
# lang              - language code 'en' or 'sv', defaults to 'sv'
#
# RETURNERAR JSON of resultat/kurstillfalle
#
# Example: ladok_session.course_instances('II2202', 'en')
def course_instances_JSON(self, course_code, lang = 'sv'):
  # note that there seems to be a limit of 403 for the number of pages
  r = self.session.get(
    url=self.base_gui_proxy_url + '/resultat/kurstillfalle/filtrera?kurskod=' +
      course_code + '&page=1&limit=100&skipCount=false&sprakkod=' + lang,
    headers=self.headers).json()
  return r
```

15.7 organization_info_JSON

78a ⟨*LadokSession data methods* 71⟩+≡ ◁77b 78b▷
```
  # added by GQMJr
  #####################################################################
  #
  # organization_info_JSON
  #
  # RETURNERAR en dictionary of organization information for the entire institution of the
  def organization_info_JSON(self):
    r = self.session.get(
      url=self.base_gui_proxy_url + '/resultat/organisation/utanlankar',
      headers=self.headers).json()
    return r
```

15.8 period_info_JSON

78b ⟨*LadokSession data methods* 71⟩+≡ ◁78a 78c▷
```
  # added by GQMJr
  #####################################################################
  #
  # period_info_JSON
  #
  # RETURNERAR JSON of /resultat/grunddata/period
  def period_info_JSON(self):
    r = self.session.get(
      url=self.base_gui_proxy_url + '/resultat/grunddata/period',
      headers=self.headers).json()
    return r
```

15.9 instance_info

78c ⟨*LadokSession data methods* 71⟩+≡ ◁78b 79a▷
```
  # added by GQMJr
  #####################################################################
  #
  # instance_info
  #
  # course_code        - course code, such as "II2202"
  #
  # instance_code      - instance of the course ('TillfallesKod')
  #
  # lang               - language code 'en' or 'sv', defaults to 'sv'
  #
  # RETURNERAR en dictionary of course instance information
  #
  # Example: ii=ladok_session.instance_info('II2202', instance_code, 'en')
```

```
def instance_info(self, course_code, instance_code, lang = 'sv'):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/resultat/kurstillfalle/filtrera?kurskod=' + course_code +
        '&page=1&limit=25&skipCount=false&sprakkod=' + lang,
    headers=self.headers)
  if r.status_code == requests.codes.ok:
    rj=r.json()
    for course in rj['Resultat']:
      if course['TillfallesKod'] == instance_code:
        return course
  return None
```

15.10 instance_info_uid

79a ⟨*LadokSession data methods* 71⟩+≡ ◁78c 79b▷

```
# added by GQMJr
#####################################################################
#
# instance_info_uid
#
# instance_uid       - course's Uid (from course_integration_id)
#
# RETURNERAR en dictionary of course instance information
#
# Example: ii=ladok_session.instance_info_uid(instance_uid)
def instance_info_uid(self, instance_uid):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/resultat/kurstillfalle/'+instance_uid,
    headers=self.headers).json()
  return r
```

15.11 studystructure_student_JSON

79b ⟨*LadokSession data methods* 71⟩+≡ ◁79a 80a▷

```
# added by GQMJr
#####################################################################
#
# studystructure_student_JSON
#
# uid                - uid of a student
#
# RETURNERAR en dictionary of student information
def studystructure_student_JSON(self, uid):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/studiedeltagande/studiestruktur/student/'+uid,
    headers=self.headers)
  if r.status_code == 200:
    return r.json()
  return None
```

15.12 larosatesinformation_JSON

80a ⟨*LadokSession data methods* 71⟩+≡ ◁79b 80b▷

```
  # added by GQMJr
  #####################################################################
  #
  # larosatesinformation_JSON
  #
  # RETURNERAR JSON of the university or college information
  def larosatesinformation_JSON(self):
    r = self.session.get(
      url=self.base_gui_proxy_url +
        '/kataloginformation/grunddata/larosatesinformation',
      headers=self.headers).json()
    return r

  # {   'Larosatesinformation': [   {   'Benamning': {   'en': 'Royal Institute of '
  #                                                            'Technology',
  #                                                      'sv': 'Kungliga Tekniska '
  #                                                            'högskolan'},
  #                                     'Beskrivning': {},
  #                                     'Giltighetsperiod': {'link': []},
  #                                     'ID': '29',
  #                                     'Kod': 'KTH',
  #                                     'LarosateID': 29,
  #                                     'OrtID': 18,
  #                                     'link': []}],
  #     'link': []}
```

15.13 undervisningssprak_JSON

80b ⟨*LadokSession data methods* 71⟩+≡ ◁80a 83▷

```
  # added by GQMJr
  #####################################################################
  #
  # undervisningssprak
  #
  # RETURNERAR en dictionary of languages used for instruction
  def undervisningssprak_JSON(self):
    r = self.session.get(
      url=self.base_gui_proxy_url +
        '/kataloginformation/grunddata/undervisningssprak',
      headers=self.headers).json()
    return r
  # {   'Undervisningssprak': [   {   'Benamning': {   'en': 'English',
  #                                                    'sv': 'Engelska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '2',
  #                               'Kod': 'ENG',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {'en': 'Russian', 'sv': 'Ryska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '4',
  #                               'Kod': 'RUS',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Sign Language',
  #                                                'sv': 'Teckenspråk'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '5',
  #                               'Kod': 'SGN',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Spanish',
  #                                                'sv': 'Spanska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '3',
  #                               'Kod': 'SPA',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Swedish',
  #                                                'sv': 'Svenska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '1',
  #                               'Kod': 'SWE',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {'en': 'Danish', 'sv': 'Danska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '109804',
  #                               'Kod': 'DAN',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Finnish',
  #                                                'sv': 'Finska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109805',
  #                                   'Kod': 'FIN',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {   'en': 'Italian',
  #                                                    'sv': 'Italienska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109806',
  #                                   'Kod': 'ITA',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {   'en': 'Japanese',
  #                                                    'sv': 'Japanska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109807',
  #                                   'Kod': 'JPN',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {   'en': 'Norwegian',
  #                                                    'sv': 'Norska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109808',
  #                                   'Kod': 'NOR',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {   'en': 'Portugese',
  #                                                    'sv': 'Portugisiska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109809',
  #                                   'Kod': 'POR',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {   'en': 'French',
  #                                                    'sv': 'Franska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109810',
  #                                   'Kod': 'FRE',
  #                                   'LarosateID': -1,
  #                                   'link': []},
  #                               {   'Benamning': {'en': 'German', 'sv': 'Tyska'},
  #                                   'Beskrivning': {},
  #                                   'Giltighetsperiod': {'link': []},
  #                                   'ID': '109811',
  #                               'Kod': 'GER',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Chinese',
  #                                                'sv': 'Kinesiska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '111033',
  #                               'Kod': 'CHI',
  #                               'LarosateID': -1,
  #                               'link': []},
  #                           {   'Benamning': {   'en': 'Arabic',
  #                                                'sv': 'Arabiska'},
  #                               'Beskrivning': {},
  #                               'Giltighetsperiod': {'link': []},
  #                               'ID': '111032',
  #                               'Kod': 'ARA',
  #                               'LarosateID': -1,
  #                               'link': []}],
  # 'link': []}
```

15.14 i18n_translation_JSON

83 ⟨*LadokSession data methods* 71⟩+≡ ◁80b 84a▷

```
# added by GQMJr
#####################################################################
#
# i18n_translation_JSON
#
# lang              - language code 'en' or 'sv', defaults to 'sv'
# RETURNERAR JSON of i18n translations used in Ladok3
def i18n_translation_JSON(self, lang = 'sv'):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/i18n/oversattningar/sprakkod/' + lang,
    headers=self.headers).json()
  return r

# the above i18n translations are used for example in:
# 'Utbildningstillfallestyp': {   'Benamningar': {   'en': 'Course instance', 'sv': 'Kur
#                                                    'Giltighetsperiod': {   'link': [
#                                                    'Grundtyp': 'KURS',
#                                                    'ID': 52,
#                                                    'Kod': '2007KTF',
#                                                    'RegelverkForUtbildningstyp': {   '
#
#
#
#
```

```
# All of the things of the form "commons-domain.*" are i18n keys to look the actual text
# for example:
# in Swedish:
#{   'I18nNyckel': 'commons.domain.regel.ingar.i.grupp.overfors.till.nya',
#    'Text': 'Ingår i grupp: Överförs till NyA',
#    'link': []},
# In English:
# {   'I18nNyckel': 'commons.domain.regel.ingar.i.grupp.overfors.till.nya',
#     'Text': 'Part of group: Transferred to NyA',
#     'link': []},
```

15.15 svenskorter_JSON

84a ⟨*LadokSession data methods* 71⟩+≡ ◁83 84b▷

```
# added by GQMJr
#####################################################################
#
# svenskorter_JSON
#
# RETURNERAR JSON of places in Sweden with their KommunID
def svenskorter_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/svenskort',
    headers=self.headers).json()
  return r

# returns:
# {   'SvenskOrt': [   {   'Benamning': {   'en': 'Stockholm (Botkyrka)',
#                                           'sv': 'Stockholm (Botkyrka)'},
#                          'Beskrivning': {},
#                          'Giltighetsperiod': {'link': []},
#                          'ID': '110990',
#                          'Kod': 'L0127',
#                          'KommunID': '8',
#                          'LarosateID': -1,
#                          'link': []},
# ... ], 'link': []}
```

15.16 kommuner_JSON

84b ⟨*LadokSession data methods* 71⟩+≡ ◁84a 85▷

```
# added by GQMJr
#####################################################################
#
# kommuner_JSON
#
# RETURNERAR JSON of places in Sweden with their KommunID
```

```
def kommuner_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/kommun',
    headers=self.headers).json()
  return r

# returns:
# {   'Kommun': [   {   'Benamning': {'en': 'Knivsta', 'sv': 'Knivsta'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {'link': []},
#                   'ID': '29',
#                   'Kod': '0330',
#                   'LanID': 2,
#                   'LarosateID': -1,
#                   'link': []},
#               {   'Benamning': {'en': 'Heby', 'sv': 'Heby'},
#                   'Beskrivning': {   'sv': 'Överförd från Västmanlands '
#                                          'till Uppsala län'},
#                   'Giltighetsperiod': {   'Startdatum': '2007-01-01',
#                                         'link': []},
#                   'ID': '30',
#                   'Kod': '0331',
#                   'LanID': 2,
#                   'LarosateID': -1,
#                   'link': []},
# ], 'link': []}
```

15.17 lander_JSON

85 ⟨*LadokSession data methods* 71⟩+≡ ◁84b 86▷

```
# added by GQMJr
#####################################################################
#
# lander_JSON
#
# RETURNERAR JSON of countries
def lander_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/land',
    headers=self.headers).json()
  return r

# returns:
# {   'Land': [   {   'Benamning': {'en': 'Bolivia', 'sv': 'Bolivia'},
#                     'Beskrivning': {},
#                     'Giltighetsperiod': {'link': []},
#                     'ID': '20',
#                     'Kod': 'BO',
#                     'LarosateID': -1,
#                     'link': []},
#                 {   'Benamning': {'en': 'Brazil', 'sv': 'Brasilien'},
#                     'Beskrivning': {},
#                     'Giltighetsperiod': {'link': []},
#                     'ID': '21',
#                     'Kod': 'BR',
#                     'LarosateID': -1,
#                     'link': []},
# ... ],    'link': []}
```

15.18 undervisningstid_JSON

86 ⟨*LadokSession data methods* 71⟩+≡ ◁85 87▷

```
# added by GQMJr
#####################################################################
#
# undervisningstid_JSON
#
# RETURNERAR JSON of teaching times
def undervisningstid_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/undervisningstid',
    headers=self.headers).json()
  return r

#returns:
# {   'Undervisningstid': [   {   'Benamning': {   'en': 'Mixed-time',
#                                                  'sv': 'Blandad '
#                                                  'undervisningstid'},
#                                 'Beskrivning': {},
#                                 'Giltighetsperiod': {'link': []},
#                                 'ID': '101051',
#                                 'Kod': 'BLA',
#                                 'LarosateID': -1,
#                                 'link': []},
#                             {   'Benamning': {'en': 'Day-time', 'sv': 'Dagtid'},
#                                 'Beskrivning': {},
#                                 'Giltighetsperiod': {'link': []},
#                                 'ID': '101052',
#                                 'Kod': 'DAG',
#                                 'LarosateID': -1,
#                                 'link': []},
#                             {   'Benamning': {   'en': 'Afternoon-time',
#                                                  'sv': 'Eftermiddagstid'},
#                                 'Beskrivning': {},
#                                 'Giltighetsperiod': {'link': []},
#                                 'ID': '101053',
#                                 'Kod': 'EFT',
#                                'LarosateID': -1,
#                                'link': []},
#                            {   'Benamning': {   'en': 'No teaching',
#                                                 'sv': 'Ingen '
#                                                 'undervisningstid'},
#                                'Beskrivning': {},
#                                'Giltighetsperiod': {   'Slutdatum': '2016-04-30',
#                                                        'link': []},
#                                'ID': '101054',
#                                'Kod': 'ING',
#                                'LarosateID': -1,
#                                'link': []},
#                            {   'Benamning': {   'en': 'Evening-time',
#                                                 'sv': 'Kvällstid'},
#                                'Beskrivning': {},
#                                'Giltighetsperiod': {'link': []},
#                                'ID': '101055',
#                                'Kod': 'KVÄ',
#                                'LarosateID': -1,
#                                'link': []},
#                            {   'Benamning': {   'en': 'Weekends',
#                                                 'sv': 'Veckoslut'},
#                                'Beskrivning': {},
#                                'Giltighetsperiod': {'link': []},
#                                'ID': '101056',
#                                'Kod': 'VSL',
#                                'LarosateID': -1,
#                                'link': []}],
#     'link': []}
```

15.19 successivfordjupning_JSON

87 ⟨*LadokSession data methods* 71⟩+≡ ◁86 88▷

```
# added by GQMJr
######################################################################
#
# successivfordjupning_JSON
#
# RETURNERAR JSON of Successive Specializations
def successivfordjupning_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/successivfordjupning',
    headers=self.headers).json()
  return r

#returns:
# {   'SuccessivFordjupning': [   {   'Benamning': {   'en': 'Second cycle, '
#                                                      'contains degree '
#                                                     'project for Master '
#                                                     'of Arts/Master of '
#                                                     'Science (60 '
#                                                     'credits)',
#                                               'sv': 'Avancerad nivå, '
#                                                     'innehåller '
#                                                     'examensarbete för '
#                                                     'magisterexamen'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {'link': []},
#                                   'ID': '1',
#                                   'Kod': 'A1E',
#                                   'LarosateID': -1,
#                                   'NivaInomStudieordningID': 2,
#                                   'link': []},
#                                 { 'Benamning': {   'en': 'Second cycle, has '
#                                                     'second-cycle '
#                                                     'course/s as entry '
#                                                     'requirements',
#                                               'sv': 'Avancerad nivå, '
#                                                     'har kurs/er på '
#                                                     'avancerad nivå som '
#                                                     'förkunskapskrav'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {'link': []},
#                                   'ID': '2',
#                                   'Kod': 'A1F',
#                                   'LarosateID': -1,
#                                   'NivaInomStudieordningID': 2,
#                                   'link': []},
# ... ], 'link': []}
```

15.20 undervisningsform_JSON

88 ⟨*LadokSession data methods* 71⟩+≡ ◁87 90▷

```
# added by GQMJr
#####################################################################
#
# undervisningsform_JSON
#
# RETURNERAR JSON of forms of education
def undervisningsform_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/undervisningsform',
    headers=self.headers).json()
  return r

#returns:
```

```
# {   'Undervisningsform': [   {   'Benamning': {   'en': '- No translation '
#                                                         'available -',
#                                                   'sv': 'IT-baserad distans'},
#                                  'Beskrivning': {},
#                                  'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                          'link': []},
#                                  'ID': '133253',
#                                  'Kod': 'ITD',
#                                  'LarosateID': 29,
#                                  'link': []},
#                              {   'Benamning': {   'en': '- No translation '
#                                                         'available -',
#                                                   'sv': 'Undervisningsområdet'},
#                                  'Beskrivning': {},
#                                  'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                          'link': []},
#                                  'ID': '133252',
#                                  'Kod': 'LU',
#                                  'LarosateID': 29,
#                                  'link': []},
#                              {   'Benamning': {   'en': 'Distance learning',
#                                                   'sv': 'Distans'},
#                                  'Beskrivning': {   'sv': 'Obligatoriska '
#                                                           'träffar kan '
#                                                           'förekomma'},
#                                  'Giltighetsperiod': {'link': []},
#                                  'ID': '2',
#                                  'Kod': 'DST',
#                                  'LarosateID': -1,
#                                  'link': []},
#                              {   'Benamning': {   'en': 'No teaching',
#                                                   'sv': 'Ingen undervisning'},
#                                  'Beskrivning': {},
#                                  'Giltighetsperiod': {   'Slutdatum': '2016-04-30',
#                                                          'link': []},
#                                  'ID': '4',
#                                  'Kod': 'ING',
#                                  'LarosateID': -1,
#                                  'link': []},
#                              {   'Benamning': {   'en': 'Web-based distance '
#                                                         'learning',
#                                                   'sv': 'IT-baserad '
#                                                         'distansutbildning'},
#                                  'Beskrivning': {   'sv': 'Ingen platsbunden '
#                                                           'undervisning'},
#                                  'Giltighetsperiod': {   'Slutdatum': '2016-04-30',
#                                                          'link': []},
#                                  'ID': '3',
#                                  'Kod': 'ITD',
#                                  'LarosateID': -1,
#                                'link': []},
#                            {   'Benamning': {   'en': 'Normal teaching',
#                                                 'sv': 'Normal'},
#                                'Beskrivning': {},
#                                'Giltighetsperiod': {'link': []},
#                                'ID': '1',
#                                'Kod': 'NML',
#                                'LarosateID': -1,
#                                'link': []}],
#     'link': []}
```

15.21 LokalaPerioder_JSON

90 ⟨*LadokSession data methods* 71⟩+≡ ◁88 91▷

```
# added by GQMJr
######################################################################
#
# LokalaPerioder_JSON
#
# RETURNERAR JSON of local periods
def LokalaPerioder_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/period',
    headers=self.headers).json()
  return r

# returns:
# {   'Period': [
#...
#                 {   'Benamning': {   'en': 'Calendar year 2020',
#                                      'sv': 'Kalenderår 2020'},
#                     'Beskrivning': {},
#                     'FromDatum': '2020-01-01',
#                     'Giltighetsperiod': {   'Slutdatum': '2020-12-31',
#                                             'Startdatum': '2020-01-01',
#                                             'link': []},
#                     'ID': '29151',
#                     'Kod': '2020',
#                     'LarosateID': 29,
#                     'PeriodtypID': 1,
#                     'TomDatum': '2020-12-31',
#                     'link': []},
#                 {   'Benamning': {   'en': 'Last six months of 2020',
#                                      'sv': 'Andra halvår 2020'},
#                     'Beskrivning': {},
#                     'FromDatum': '2020-07-01',
#                     'Giltighetsperiod': {   'Slutdatum': '2020-12-31',
#                                             'Startdatum': '2020-07-01',
#                                             'link': []},
#                   'ID': '29252',
#                   'Kod': '2020H',
#                   'LarosateID': 29,
#                   'PeriodtypID': 3,
#                   'TomDatum': '2020-12-31',
#                   'link': []},
#               {   'Benamning': {   'en': 'First six months of 2020',
#                                    'sv': 'Första halvår 2020'},
#                   'Beskrivning': {},
#                   'FromDatum': '2020-01-01',
#                   'Giltighetsperiod': {   'Slutdatum': '2020-06-30',
#                                           'Startdatum': '2020-01-01',
#                                           'link': []},
#                   'ID': '29324',
#                   'Kod': '2020V',
#                   'LarosateID': 29,
#                   'PeriodtypID': 3,
#                   'TomDatum': '2020-06-30',
#                   'link': []},
#...
# 'link': []}
```

15.22 nivainomstudieordning_JSON

91 ⟨*LadokSession data methods* 71⟩+≡ ◁90 92▷

```
# added by GQMJr
######################################################################
#
# nivainomstudieordning_JSON
#
# RETURNERAR JSON of education levels
def nivainomstudieordning_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/nivainomstudieordning',
    headers=self.headers).json()
  return r

# returns:
# {   'NivaInomStudieordning': [   {   'Benamning': {   'en': 'First cycle',
#                                                       'sv': 'Grundnivå'},
#                                      'Beskrivning': {},
#                                      'Giltighetsperiod': {   'Startdatum': '2007-07-01
#                                      'ID': '1',
#                                      'Kod': '1',
#                                      'LarosateID': -1,
#                                      'link': []},
#                                  {   'Benamning': {   'en': 'Second cycle',
#                                                       'sv': 'Avancerad nivå'},
#                                           'Beskrivning': {},
#                                           'Giltighetsperiod': {   'Startdatum': '2007-07-01
#                                           'ID': '2',
#                                           'Kod': '2',
#                                           'LarosateID': -1,
#                                           'link': []},
#                                       {   'Benamning': {   'en': 'Third cycle',
#                                                            'sv': 'Forskarnivå'},
#                                           'Beskrivning': {},
#                                           'Giltighetsperiod': {   'Startdatum': '2007-07-01
#                                           'ID': '3',
#                                           'Kod': '3',
#                                           'LarosateID': -1,
#                                           'link': []},
#                                       {   'Benamning': {   'en': 'Postgraduate '
#                                                                  'level',
#                                                            'sv': 'Forskarutbildning'},
#                                           'Beskrivning': {},
#                                           'Giltighetsperiod': {   'Slutdatum': '2007-06-30'
#                                                                   'Startdatum': '1977-07-01
#                                           'ID': '5',
#                                           'Kod': 'F',
#                                           'LarosateID': -1,
#                                           'link': []},
#                                       {   'Benamning': {   'en': 'Undergraduate '
#                                                                  'level',
#                                                            'sv': 'Grundutbildning'},
#                                           'Beskrivning': {},
#                                           'Giltighetsperiod': {   'Slutdatum': '2007-06-30'
#                                                                   'Startdatum': '1977-07-01
#                                           'ID': '4',
#                                           'Kod': 'G',
#                                           'LarosateID': -1,
#                                           'link': []}],
#     'link': []}
```

15.23 amnesgrupp_JSON

92 ⟨*LadokSession data methods* 71⟩+≡ ◁91 93▷

```
# added by GQMJr
#####################################################################
#
# amnesgrupp_JSON
#
# RETURNERAR JSON of subject area groups
def amnesgrupp_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/amnesgrupp',
    headers=self.headers).json()
  return r

# returns:
# {   'Amnesgrupp': [   {   'Benamning': {   'en': 'Archival Science',
#                                            'sv': 'Arkivvetenskap'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '10',
#                           'Kod': 'AV1',
#                           'LarosateID': -1,
#                           'link': []},
# ...
#                   {   'Benamning': {'en': 'Philosophy', 'sv': 'Filosofi'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '42',
#                           'Kod': 'FI2',
#                           'LarosateID': -1,
#                           'link': []},
# ...
#                   {   'Benamning': {   'en': 'Informatics/Computer and '
#                                                  'Systems Sciences',
#                                            'sv': 'Informatik/data- och '
#                                                  'systemvetenskap'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '68',
#                           'Kod': 'IF1',
#                           'LarosateID': -1,
#                           'link': []},
# ...
#                   {   'Benamning': {   'en': 'Electronics',
#                                            'sv': 'Elektronik'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '30',
#                           'Kod': 'EL1',
#                           'LarosateID': -1,
#                           'link': []},
# ... ],
#     'link': []}
```

15.24 studietakt_JSON

93 ⟨*LadokSession data methods* 71⟩+≡ ◁92 114a▷

```
# added by GQMJr
#####################################################################
#
# studietakt_JSON
```

```
#
# RETURNERAR JSON of study tempos
def studietakt_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/studietakt',
    headers=self.headers).json()
  return r

# returns:
# {   'Studietakt': [   {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Noll'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133268',
#                         'Kod': '0',
#                         'LarosateID': 29,
#                         'Takt': 0,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Kvartstid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133270',
#                         'Kod': '1',
#                         'LarosateID': 29,
#                         'Takt': 1,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Kvartstid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133282',
#                         'Kod': '11',
#                         'LarosateID': 29,
#                         'Takt': 11,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Kvartstid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133281',
#                         'Kod': '13',
#                         'LarosateID': 29,
#                         'Takt': 13,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                       'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133278',
#                       'Kod': '14',
#                       'LarosateID': 29,
#                       'Takt': 14,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                       'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133283',
#                       'Kod': '15',
#                       'LarosateID': 29,
#                       'Takt': 15,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                       'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133280',
#                       'Kod': '16',
#                       'LarosateID': 29,
#                       'Takt': 16,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                       'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133289',
#                       'Kod': '18',
#                       'LarosateID': 29,
#                       'Takt': 18,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                       'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133279',
#                       'Kod': '19',
#                       'LarosateID': 29,
#                       'Takt': 19,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                   'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133271',
#                       'Kod': '2',
#                       'LarosateID': 29,
#                       'Takt': 2,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                   'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133285',
#                       'Kod': '20',
#                       'LarosateID': 29,
#                       'Takt': 20,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                   'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133287',
#                       'Kod': '21',
#                       'LarosateID': 29,
#                       'Takt': 21,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                   'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133290',
#                       'Kod': '22',
#                       'LarosateID': 29,
#                       'Takt': 22,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                   'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133284',
#                       'Kod': '23',
#                       'LarosateID': 29,
#                       'Takt': 23,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133286',
#                       'Kod': '24',
#                       'LarosateID': 29,
#                       'Takt': 24,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133288',
#                       'Kod': '26',
#                       'LarosateID': 29,
#                       'Takt': 26,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133291',
#                       'Kod': '27',
#                       'LarosateID': 29,
#                       'Takt': 27,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133292',
#                       'Kod': '28',
#                       'LarosateID': 29,
#                       'Takt': 28,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133294',
#                       'Kod': '29',
#                       'LarosateID': 29,
#                       'Takt': 29,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                        ’sv’: ’Kvartstid’},
#                       ’Beskrivning’: {},
#                       ’Giltighetsperiod’: {   ’Slutdatum’: ’2018-06-19’,
#                                               ’link’: []},
#                       ’ID’: ’133274’,
#                       ’Kod’: ’3’,
#                       ’LarosateID’: 29,
#                       ’Takt’: 3,
#                       ’link’: []},
#                   {   ’Benamning’: {   ’en’: ’- No translation available -’,
#                                        ’sv’: ’Kvartstid’},
#                       ’Beskrivning’: {},
#                       ’Giltighetsperiod’: {   ’Slutdatum’: ’2018-06-19’,
#                                               ’link’: []},
#                       ’ID’: ’133293’,
#                       ’Kod’: ’30’,
#                       ’LarosateID’: 29,
#                       ’Takt’: 30,
#                       ’link’: []},
#                   {   ’Benamning’: {   ’en’: ’- No translation available -’,
#                                        ’sv’: ’Kvartstid’},
#                       ’Beskrivning’: {},
#                       ’Giltighetsperiod’: {   ’Slutdatum’: ’2018-06-19’,
#                                               ’link’: []},
#                       ’ID’: ’133296’,
#                       ’Kod’: ’31’,
#                       ’LarosateID’: 29,
#                       ’Takt’: 31,
#                       ’link’: []},
#                   {   ’Benamning’: {   ’en’: ’- No translation available -’,
#                                        ’sv’: ’Kvartstid’},
#                       ’Beskrivning’: {},
#                       ’Giltighetsperiod’: {   ’Slutdatum’: ’2018-06-19’,
#                                               ’link’: []},
#                       ’ID’: ’133295’,
#                       ’Kod’: ’32’,
#                       ’LarosateID’: 29,
#                       ’Takt’: 32,
#                       ’link’: []},
#                   {   ’Benamning’: {   ’en’: ’- No translation available -’,
#                                        ’sv’: ’Kvartstid’},
#                       ’Beskrivning’: {},
#                       ’Giltighetsperiod’: {   ’Slutdatum’: ’2018-06-19’,
#                                               ’link’: []},
#                       ’ID’: ’133299’,
#                       ’Kod’: ’34’,
#                       ’LarosateID’: 29,
#                       ’Takt’: 34,
#                       ’link’: []},
#                   {   ’Benamning’: {   ’en’: ’- No translation available -’,
#                                 'sv': 'Kvartstid'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                           'link': []},
#                   'ID': '133298',
#                   'Kod': '35',
#                   'LarosateID': 29,
#                   'Takt': 35,
#                   'link': []},
#               {   'Benamning': {   'en': '- No translation available -',
#                                 'sv': 'Kvartstid'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                           'link': []},
#                   'ID': '133297',
#                   'Kod': '36',
#                   'LarosateID': 29,
#                   'Takt': 36,
#                   'link': []},
#               {   'Benamning': {   'en': '- No translation available -',
#                                 'sv': 'Kvartstid'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                           'link': []},
#                   'ID': '133307',
#                   'Kod': '37',
#                   'LarosateID': 29,
#                   'Takt': 37,
#                   'link': []},
#               {   'Benamning': {   'en': '- No translation available -',
#                                 'sv': 'Halvtid'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                           'link': []},
#                   'ID': '133303',
#                   'Kod': '38',
#                   'LarosateID': 29,
#                   'Takt': 38,
#                   'link': []},
#               {   'Benamning': {   'en': '- No translation available -',
#                                 'sv': 'Halvtid'},
#                   'Beskrivning': {},
#                   'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                           'link': []},
#                   'ID': '133300',
#                   'Kod': '39',
#                   'LarosateID': 29,
#                   'Takt': 39,
#                   'link': []},
#               {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Kvartstid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133269',
#                       'Kod': '4',
#                       'LarosateID': 29,
#                       'Takt': 4,
#                       'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                    'sv': 'Halvtid'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                               'link': []},
#                       'ID': '133302',
#                       'Kod': '40',
#                       'LarosateID': 29,
#                       'Takt': 40,
#                       'link': []},
#                   {   'Benamning': {   'en': 'One-tenth-time',
#                                    'sv': 'Tiondelsfart'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {'link': []},
#                       'ID': '8',
#                       'Kod': '10',
#                       'LarosateID': -1,
#                       'Takt': 10,
#                       'link': []},
#                   {   'Benamning': {'en': 'Full-time', 'sv': 'Helfart'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {'link': []},
#                       'ID': '1',
#                       'Kod': '100',
#                       'LarosateID': -1,
#                       'Takt': 100,
#                       'link': []},
#                   {   'Benamning': {   'en': 'One-eight-time',
#                                    'sv': 'Åttondelsfart'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {'link': []},
#                       'ID': '128190',
#                       'Kod': '12',
#                       'LarosateID': -1,
#                       'Takt': 12,
#                       'link': []},
#                   {   'Benamning': {   'en': 'One-sixth-time',
#                                    'sv': 'Sjättedelsfart'},
#                       'Beskrivning': {},
#                       'Giltighetsperiod': {'link': []},
#                       'ID': '7',
#                           'Kod': '17',
#                           'LarosateID': -1,
#                           'Takt': 17,
#                           'link': []},
#                       {   'Benamning': {   'en': 'One-quarter-time',
#                                            'sv': 'Kvartsfart'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '6',
#                           'Kod': '25',
#                           'LarosateID': -1,
#                           'Takt': 25,
#                           'link': []},
#                       {   'Benamning': {   'en': 'One-third-time',
#                                            'sv': 'Tredjedelsfart'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '5',
#                           'Kod': '33',
#                           'LarosateID': -1,
#                           'Takt': 33,
#                           'link': []},
#                       {   'Benamning': {'en': 'Half-time', 'sv': 'Halvfart'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '4',
#                           'Kod': '50',
#                           'LarosateID': -1,
#                           'Takt': 50,
#                           'link': []},
#                       {   'Benamning': {   'en': 'Two-thirds-time',
#                                            'sv': 'Tvåtredjedelsfart'},
#                           'Beskrivning': {   'sv': 'Ändras till '
#                                                    '"Tvåtredjedelsfart" efter '
#                                                    'fullbordad '
#                                                    'produktionssättning.'},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '3',
#                           'Kod': '67',
#                           'LarosateID': -1,
#                           'Takt': 67,
#                           'link': []},
#                       {   'Benamning': {   'en': 'Three-quarters-time',
#                                            'sv': 'Trekvartsfart'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {'link': []},
#                           'ID': '2',
#                           'Kod': '75',
#                           'LarosateID': -1,
#                           'Takt': 75,
#                            'link': []},
#                        {   'Benamning': {   'en': '- No translation available -',
#                                             'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133304',
#                            'Kod': '41',
#                            'LarosateID': 29,
#                            'Takt': 41,
#                            'link': []},
#                        {   'Benamning': {   'en': '- No translation available -',
#                                             'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133305',
#                            'Kod': '42',
#                            'LarosateID': 29,
#                            'Takt': 42,
#                            'link': []},
#                        {   'Benamning': {   'en': '- No translation available -',
#                                             'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133301',
#                            'Kod': '43',
#                            'LarosateID': 29,
#                            'Takt': 43,
#                            'link': []},
#                        {   'Benamning': {   'en': '- No translation available -',
#                                             'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133312',
#                            'Kod': '44',
#                            'LarosateID': 29,
#                            'Takt': 44,
#                            'link': []},
#                        {   'Benamning': {   'en': '- No translation available -',
#                                             'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133308',
#                            'Kod': '45',
#                            'LarosateID': 29,
#                            'Takt': 45,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133306',
#                           'Kod': '46',
#                           'LarosateID': 29,
#                           'Takt': 46,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133310',
#                           'Kod': '47',
#                           'LarosateID': 29,
#                           'Takt': 47,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133309',
#                           'Kod': '48',
#                           'LarosateID': 29,
#                           'Takt': 48,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133311',
#                           'Kod': '49',
#                           'LarosateID': 29,
#                           'Takt': 49,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Kvartstid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133276',
#                           'Kod': '5',
#                           'LarosateID': 29,
#                           'Takt': 5,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133318',
#                           'Kod': '51',
#                           'LarosateID': 29,
#                           'Takt': 51,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133315',
#                           'Kod': '52',
#                           'LarosateID': 29,
#                           'Takt': 52,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133314',
#                           'Kod': '53',
#                           'LarosateID': 29,
#                           'Takt': 53,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133313',
#                           'Kod': '54',
#                           'LarosateID': 29,
#                           'Takt': 54,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133316',
#                           'Kod': '55',
#                           'LarosateID': 29,
#                           'Takt': 55,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133319',
#                           'Kod': '56',
#                           'LarosateID': 29,
#                           'Takt': 56,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133317',
#                           'Kod': '57',
#                           'LarosateID': 29,
#                           'Takt': 57,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133320',
#                           'Kod': '58',
#                           'LarosateID': 29,
#                           'Takt': 58,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133322',
#                           'Kod': '59',
#                           'LarosateID': 29,
#                           'Takt': 59,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Kvartstid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133272',
#                           'Kod': '6',
#                           'LarosateID': 29,
#                           'Takt': 6,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133323',
#                           'Kod': '60',
#                           'LarosateID': 29,
#                           'Takt': 60,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133321',
#                           'Kod': '61',
#                           'LarosateID': 29,
#                           'Takt': 61,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133332',
#                           'Kod': '62',
#                           'LarosateID': 29,
#                           'Takt': 62,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133331',
#                           'Kod': '63',
#                           'LarosateID': 29,
#                           'Takt': 63,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133325',
#                           'Kod': '64',
#                           'LarosateID': 29,
#                           'Takt': 64,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133324',
#                           'Kod': '65',
#                           'LarosateID': 29,
#                           'Takt': 65,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133327',
#                           'Kod': '66',
#                           'LarosateID': 29,
#                           'Takt': 66,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133328',
#                           'Kod': '68',
#                           'LarosateID': 29,
#                           'Takt': 68,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133329',
#                           'Kod': '69',
#                           'LarosateID': 29,
#                           'Takt': 69,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Kvartstid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133277',
#                           'Kod': '7',
#                           'LarosateID': 29,
#                           'Takt': 7,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133330',
#                            'Kod': '70',
#                            'LarosateID': 29,
#                            'Takt': 70,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133326',
#                            'Kod': '71',
#                            'LarosateID': 29,
#                            'Takt': 71,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Halvtid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133333',
#                            'Kod': '72',
#                            'LarosateID': 29,
#                            'Takt': 72,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133334',
#                            'Kod': '73',
#                            'LarosateID': 29,
#                            'Takt': 73,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                            'Beskrivning': {},
#                            'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                    'link': []},
#                            'ID': '133342',
#                            'Kod': '74',
#                            'LarosateID': 29,
#                            'Takt': 74,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133335',
#                           'Kod': '76',
#                           'LarosateID': 29,
#                           'Takt': 76,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133347',
#                           'Kod': '77',
#                           'LarosateID': 29,
#                           'Takt': 77,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133339',
#                           'Kod': '78',
#                           'LarosateID': 29,
#                           'Takt': 78,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133338',
#                           'Kod': '79',
#                           'LarosateID': 29,
#                           'Takt': 79,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Kvartstid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133275',
#                           'Kod': '8',
#                           'LarosateID': 29,
#                           'Takt': 8,
#                            'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133337',
#                           'Kod': '80',
#                           'LarosateID': 29,
#                           'Takt': 80,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133336',
#                           'Kod': '81',
#                           'LarosateID': 29,
#                           'Takt': 81,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133340',
#                           'Kod': '82',
#                           'LarosateID': 29,
#                           'Takt': 82,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133345',
#                           'Kod': '83',
#                           'LarosateID': 29,
#                           'Takt': 83,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133341',
#                           'Kod': '84',
#                           'LarosateID': 29,
#                           'Takt': 84,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133343',
#                           'Kod': '85',
#                           'LarosateID': 29,
#                           'Takt': 85,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133344',
#                           'Kod': '86',
#                           'LarosateID': 29,
#                           'Takt': 86,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133346',
#                           'Kod': '87',
#                           'LarosateID': 29,
#                           'Takt': 87,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133348',
#                           'Kod': '88',
#                           'LarosateID': 29,
#                           'Takt': 88,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133349',
#                           'Kod': '89',
#                           'LarosateID': 29,
#                           'Takt': 89,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Kvartstid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133273',
#                           'Kod': '9',
#                           'LarosateID': 29,
#                           'Takt': 9,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133350',
#                           'Kod': '90',
#                           'LarosateID': 29,
#                           'Takt': 90,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133352',
#                           'Kod': '91',
#                           'LarosateID': 29,
#                           'Takt': 91,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133358',
#                           'Kod': '92',
#                           'LarosateID': 29,
#                           'Takt': 92,
#                           'link': []},
#                       {   'Benamning': {   'en': '- No translation available -',
#                                            'sv': 'Nästan heltid'},
#                           'Beskrivning': {},
#                           'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                   'link': []},
#                           'ID': '133353',
#                           'Kod': '93',
#                           'LarosateID': 29,
#                           'Takt': 93,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Nästan heltid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133354',
#                         'Kod': '94',
#                         'LarosateID': 29,
#                         'Takt': 94,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Nästan heltid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133351',
#                         'Kod': '95',
#                         'LarosateID': 29,
#                         'Takt': 95,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Nästan heltid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133355',
#                         'Kod': '96',
#                         'LarosateID': 29,
#                         'Takt': 96,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Nästan heltid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133357',
#                         'Kod': '97',
#                         'LarosateID': 29,
#                         'Takt': 97,
#                         'link': []},
#                     {   'Benamning': {   'en': '- No translation available -',
#                                          'sv': 'Nästan heltid'},
#                         'Beskrivning': {},
#                         'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                 'link': []},
#                         'ID': '133356',
#                         'Kod': '98',
#                         'LarosateID': 29,
#                         'Takt': 98,
#                        'link': []},
#                   {   'Benamning': {   'en': '- No translation available -',
#                                        'sv': 'Nästan heltid'},
#                        'Beskrivning': {},
#                        'Giltighetsperiod': {   'Slutdatum': '2018-06-19',
#                                                'link': []},
#                        'ID': '133359',
#                        'Kod': '99',
#                        'LarosateID': 29,
#                        'Takt': 99,
#                        'link': []}],
# 'link': []}
```

15.25 finansieringsform_JSON

114a ⟨*LadokSession data methods* 71⟩+≡ ◁93 114b▷

```
# added by GQMJr
######################################################################
#
# finansieringsform_JSON
#
# RETURNERAR JSON forms of financing
def finansieringsform_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/finansieringsform',
    headers=self.headers).json()
  return r

# returns:
# {   'Finansieringsform': [   {   'Benamning': {   'en': '- No translation '
#                                                       'available -',
#                                                 'sv': 'Alumn, med examen fr '
#                                                       'KTH (ej HST-HPR)'},
#                                 'Beskrivning': {},
#                                 'Giltighetsperiod': {'link': []},
#                                 'ID': '131704',
#                                 'Kod': 'ALU',
#                                 'LarosateID': 29,
#                                 'link': []},
#                             {   'Benamning': {   'en': '- No translation '
# ...
```

15.26 utbildningsomrade_JSON

114b ⟨*LadokSession data methods* 71⟩+≡ ◁114a 116▷

```
# added by GQMJr
```

```
######################################################################
#
# utbildningsomrade_JSON
#
# RETURNERAR JSON of subjects
def utbildningsomrade_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/utbildningsomrade',
    headers=self.headers).json()
  return r

# returns:
# {   'Utbildningsomrade': [   {   'Benamning': {   'en': 'Dance',
#                                                   'sv': 'Dansområdet'},
#                                  'Beskrivning': {   'sv': 'Infördes från '
#                                                          'budgetåret 1994/95. '
#                                                          'Tidigare fanns bara '
#                                                          'ett konstnärligt '
#                                                          'utbildningsområde '
#                                                          '(vars '
#                                                          'ersättningsbelopp '
#                                                          'varierade mellan '
#                                                          'lärosätena).'},
#                                  'Giltighetsperiod': {   'Startdatum': '1994-07-01',
#                                                          'link': []},
#                                  'ID': '1',
#                                  'Kod': 'DA',
#                                  'LarosateID': -1,
#                                  'link': []},
# ...
#                              {   'Benamning': {   'en': 'Education',
#                                                   'sv': 'Undervisningsområdet'},
#                                  'Beskrivning': {   'sv': 'Avser utbildning '
#                                                          'inom det allmänna '
#                                                          'utbildningsområdet '
#                                                          'och den '
#                                                          'utbildningsvetenskapliga '
#                                                          'kärnan. T.o.m. '
#                                                          '2012-12-31 avsågs '
#                                                          'även övrig '
#                                                          'verksamhetsförlagd '
#                                                          'utbildning (se '
#                                                          'VU).'},
#                                  'Giltighetsperiod': {   'Startdatum': '1993-07-01',
#                                                          'link': []},
#                                  'ID': '9',
#                                  'Kod': 'LU',
#                                  'LarosateID': -1,
#                                  'link': []},
# ...
#                           {   'Benamning': {   'en': 'Natural sciences',
#                                                'sv': 'Naturvetenskapliga '
#                                                      'området'},
#                               'Beskrivning': {},
#                               'Giltighetsperiod': {   'Startdatum': '1993-07-01',
#                                                       'link': []},
#                               'ID': '13',
#                               'Kod': 'NA',
#                               'LarosateID': -1,
#                               'link': []},
# ...
#                           {   'Benamning': {   'en': 'Technology',
#                                                'sv': 'Tekniska området'},
#                               'Beskrivning': {},
#                               'Giltighetsperiod': {   'Startdatum': '1993-07-01',
#                                                       'link': []},
#                               'ID': '19',
#                               'Kod': 'TE',
#                               'LarosateID': -1,
#                               'link': []},
# ... ],
# 'link': []}
```

15.27 kravpatidigarestudier_JSON

116 ⟨*LadokSession data methods* 71⟩+≡ ◁114b 117▷

```
# added by GQMJr
#####################################################################
#
# kravpatidigarestudier_JSON
#
# RETURNERAR JSON of krequirements for earlier studies
def kravpatidigarestudier_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/kravpatidigarestudier',
    headers=self.headers).json()
  return r

# returns
# {   'KravPaTidigareStudier': [   {   'Benamning': {   'en': 'University '
#                                                             'studies required',
#                                                       'sv': 'Tidigare '
#                                                             'högskolestudier '
#                                                             'krävs'},
#                                    'Beskrivning': {},
#                                    'Emilvarde': 'uh',
#                                    'Giltighetsperiod': {'link': []},
#                                   'ID': '1',
#                                   'Kod': 'UH',
#                                   'LarosateID': -1,
#                                   'link': []},
#                               {   'Benamning': {   'en': 'Upper secondary '
#                                                          'or equivalent',
#                                                    'sv': 'Inga tidigare '
#                                                          'högskolestudier '
#                                                          'krävs'},
#                                   'Beskrivning': {},
#                                   'Emilvarde': 'grundlaggande',
#                                   'Giltighetsperiod': {'link': []},
#                                   'ID': '2',
#                                   'Kod': 'GR',
#                                   'LarosateID': -1,
#                                   'link': []},
#                               {   'Benamning': {   'en': 'No general entry '
#                                                          'requirements '
#                                                          'needed',
#                                                    'sv': 'Ingen '
#                                                          'grundläggande '
#                                                          'behörighet krävs'},
#                                   'Beskrivning': {},
#                                   'Emilvarde': 'inga',
#                                   'Giltighetsperiod': {'link': []},
#                                   'ID': '3',
#                                   'Kod': 'IN',
#                                   'LarosateID': -1,
#                                   'link': []}],
# 'link': []}
```

15.28 studieordning_JSON

117 ⟨*LadokSession data methods* 71⟩+≡ ◁116 123▷

```
# added by GQMJr
####################################################################
#
# studieordning_JSON
#
# RETURNERAR JSON of study regulation
def studieordning_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/studieordning',
    headers=self.headers).json()
  return r

# returns:
# {   'Studieordning': [   {   'Benamning': {   'en': 'Higher education, study '
#                                                  'regulation of 1993',
#                                            'sv': 'Högskoleutbildning, 1993 '
#                                                  'års studieordning'},
#                             'Beskrivning': {   'sv': 'Avser i Ladok 1993 års '
#                                                    'studieordning inklusive '
#                                                    'dess föregångare'},
#                             'EnhetID': 9,
#                             'Giltighetsperiod': {'link': []},
#                             'ID': '2',
#                             'Kod': 'HÖ93',
#                             'LarosateID': -1,
#                             'UtbildningsformID': 1,
#                             'link': []},
#                         {   'Benamning': {   'en': 'Higher education, study '
#                                                  'regulation of 2007',
#                                            'sv': 'Högskoleutbildning, 2007 '
#                                                  'års studieordning'},
#                             'Beskrivning': {},
#                             'EnhetID': 2,
#                             'Giltighetsperiod': {   'Startdatum': '2007-07-01',
#                                                     'link': []},
#                             'ID': '1',
#                             'Kod': 'HÖ07',
#                             'LarosateID': -1,
#                             'UtbildningsformID': 1,
#                             'link': []},
#                         {   'Benamning': {   'en': 'Access education (hours)',
#                                            'sv': 'Behörighetsgivande '
#                                                  'förutbildning (timmar)'},
#                             'Beskrivning': {},
#                             'EnhetID': 5,
#                             'Giltighetsperiod': {'link': []},
#                             'ID': '12',
#                             'Kod': 'ÖVBT',
#                             'LarosateID': -1,
#                             'UtbildningsformID': 100970,
#                             'link': []},
#                         {   'Benamning': {   'en': 'Access education (fup)',
#                                            'sv': 'Behörighetsgivande '
#                                                  'förutbildning (poäng)'},
#                             'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                    'förordning (2018:1519) '
#                                                    'om behörighetsgivande '
#                                                    'och '
#                                                    'högskoleintroducerande '
#                                                    'utbildning resp. '
#                                                    'tidigare gällande '
#                                                    'förordning (2007:432) '
#                                                    'om behörighetsgivande '
#                                                    'förutbildning vid '
#                                                     'universitet och '
#                                                     'högskolor'},
#                               'EnhetID': 4,
#                               'Giltighetsperiod': {'link': []},
#                               'ID': '3',
#                               'Kod': 'FÖPO',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 2,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Access education (weeks)',
#                                                'sv': 'Behörighetsgivande '
#                                                    'förutbildning (veckor)'},
#                               'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                     'förordning (2018:1519) '
#                                                     'om behörighetsgivande '
#                                                     'och '
#                                                     'högskoleintroducerande '
#                                                     'utbildning resp. '
#                                                     'tidigare gällande '
#                                                     'förordning (2007:432) '
#                                                     'om behörighetsgivande '
#                                                     'förutbildning vid '
#                                                     'universitet och '
#                                                     'högskolor'},
#                               'EnhetID': 1,
#                               'Giltighetsperiod': {'link': []},
#                               'ID': '4',
#                               'Kod': 'FÖVE',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 2,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Internal education (ORU)',
#                                                'sv': 'Högskoleintern utbildning '
#                                                    '(ORU)'},
#                               'Beskrivning': {   'sv': 'Intern utbildning vid '
#                                                     'Örebro universitet'},
#                               'EnhetID': 6,
#                               'Giltighetsperiod': {   'Slutdatum': '2007-06-30',
#                                                      'Startdatum': '2004-01-01',
#                                                      'link': []},
#                               'ID': '15',
#                               'Kod': 'ÖVHI',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100970,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Older defence education',
#                                                'sv': 'Äldre utbildning vid '
#                                                    'Försvarshögskolan'},
#                               'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                     'förordningen '
#                                                '(1996:1476) med '
#                                                'instruktion för '
#                                                'Försvarshögskolan'},
#                                 'EnhetID': 10,
#                                 'Giltighetsperiod': {   'Slutdatum': '2007-12-31',
#                                                         'link': []},
#                                 'ID': '16',
#                                 'Kod': 'ÖVFU',
#                                 'LarosateID': -1,
#                                 'UtbildningsformID': 100970,
#                                 'link': []},
#                             {   'Benamning': {   'en': 'Post-secondary vocational '
#                                                        'education and training',
#                                                  'sv': 'Kvalificerad '
#                                                        'yrkesutbildning'},
#                                 'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                          'förordningen '
#                                                          '(2001:1131) om '
#                                                          'kvalificerad '
#                                                          'yrkesutbildning '
#                                                          '(upphävd 2009-04-15)'},
#                                 'EnhetID': 7,
#                                 'Giltighetsperiod': {   'Slutdatum': '2013-12-31',
#                                                         'link': []},
#                                 'ID': '13',
#                                 'Kod': 'KY02',
#                                 'LarosateID': -1,
#                                 'UtbildningsformID': 100968,
#                                 'link': []},
#                             {   'Benamning': {   'en': 'Preparatory education',
#                                                  'sv': 'Preparandutbildning'},
#                                 'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                          'förordningen (1985:681) '
#                                                          'om preparandutbildning '
#                                                          'i svenska (upphävd '
#                                                          '1993-07-01)'},
#                                 'EnhetID': 6,
#                                 'Giltighetsperiod': {   'Slutdatum': '1993-06-30',
#                                                         'link': []},
#                                 'ID': '14',
#                                 'Kod': 'ÖVPR',
#                                 'LarosateID': -1,
#                                 'UtbildningsformID': 100970,
#                                 'link': []},
#                             {   'Benamning': {   'en': 'Contract education '
#                                                        '(hours)',
#                                                  'sv': 'Uppdragsutbildning, Övrig '
#                                                        'utbildning (timmar)'},
#                                 'Beskrivning': {},
#                                 'EnhetID': 5,
#                               'Giltighetsperiod': {   'Slutdatum': '2018-12-31',
#                                                       'link': []},
#                               'ID': '9',
#                               'Kod': 'ÖVUT',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100970,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Higher vocational '
#                                                      'education',
#                                                'sv': 'Yrkeshögskoleutbildning'},
#                               'Beskrivning': {   'sv': 'Utbildning enligt '
#                                                        'förordning (2009:130) '
#                                                        'om yrkeshögskolan'},
#                               'EnhetID': 8,
#                               'Giltighetsperiod': {   'Startdatum': '2009-07-01',
#                                                       'link': []},
#                               'ID': '5',
#                               'Kod': 'YH09',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 4,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Contract education '
#                                                      '(credits)',
#                                                'sv': 'Uppdragsutbildning '
#                                                      '(högskolepoäng)'},
#                               'Beskrivning': {},
#                               'EnhetID': 2,
#                               'Giltighetsperiod': {   'Startdatum': '2007-07-01',
#                                                       'link': []},
#                               'ID': '17',
#                               'Kod': 'UPHP',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100928,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Contract education '
#                                                      '(weeks)',
#                                                'sv': 'Uppdragsutbildning '
#                                                      '(veckor)'},
#                               'Beskrivning': {},
#                               'EnhetID': 1,
#                               'Giltighetsperiod': {'link': []},
#                               'ID': '18',
#                               'Kod': 'UPVE',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100928,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Police education',
#                                                'sv': 'Polisutbildning'},
#                               'Beskrivning': {   'sv': 'Polisutbildning som ej '
#                                                        'uppfyller alla '
#                                                       'kvalitetskrav för '
#                                                       'högskoleutbildning.'},
#                               'EnhetID': 2,
#                               'Giltighetsperiod': {'link': []},
#                               'ID': '109831',
#                               'Kod': 'PU99',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100969,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Police education, equal '
#                                                      'to Higher education',
#                                                'sv': 'Polisutbildning, '
#                                                      'motsvarande '
#                                                      'högskoleutbildning'},
#                               'Beskrivning': {   'sv': 'Polisutbildning som '
#                                                        'uppfyller '
#                                                        'kvalitetskraven för '
#                                                        'högskoleutbildning'},
#                               'EnhetID': 2,
#                               'Giltighetsperiod': {   'Startdatum': '2018-07-01',
#                                                       'link': []},
#                               'ID': '135370',
#                               'Kod': 'PU18',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100969,
#                               'link': []},
#                           {   'Benamning': {'en': 'Ö-Fel', 'sv': 'Ö-Fel'},
#                               'Beskrivning': {},
#                               'EnhetID': 1,
#                               'Giltighetsperiod': {   'Slutdatum': '1900-01-01',
#                                                       'link': []},
#                               'ID': '138710',
#                               'Kod': 'FEL',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100970,
#                               'link': []},
#                           {   'Benamning': {   'en': 'Contract education '
#                                                      '(hours)',
#                                                'sv': 'Uppdragsutbildning '
#                                                      '(timmar)'},
#                               'Beskrivning': {},
#                               'EnhetID': 5,
#                               'Giltighetsperiod': {   'Startdatum': '2019-01-01',
#                                                       'link': []},
#                               'ID': '147898',
#                               'Kod': 'UPTI',
#                               'LarosateID': -1,
#                               'UtbildningsformID': 100928,
#                               'link': []}],
#     'link': []}
```

15.29 enhet_JSON

123 ⟨*LadokSession data methods* 71⟩+≡ ◁117 125▷

```
# added by GQMJr
######################################################################
#
# enhet_JSON
#
# RETURNERAR JSON of units
def enhet_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/grunddata/enhet',
    headers=self.headers).json()
  return r

# returns:
# {   'Enhet': [   {   'Benamning': {   'en': 'Credit points',
#                                     'sv': 'Poäng (Övrig utbildning)'},
#                  'Beskrivning': {},
#                  'Giltighetsperiod': {   'Slutdatum': '2018-01-01',
#                                          'link': []},
#                  'Helarsvarde': 40,
#                  'ID': '6',
#                  'Kod': 'AUP',
#                  'LarosateID': -1,
#                  'link': []},
#              {   'Benamning': {   'en': 'Pre-education credits',
#                                     'sv': 'Förutbildningspoäng'},
#                  'Beskrivning': {},
#                  'Giltighetsperiod': {'link': []},
#                  'Helarsvarde': 60,
#                  'ID': '4',
#                  'Kod': 'FUP',
#                  'LarosateID': -1,
#                  'link': []},
#              {   'Benamning': {'en': 'Credits', 'sv': 'Högskolepoäng'},
#                  'Beskrivning': {   'sv': 'Översattes 2007-2010 med Higher '
#                                           'education credits'},
#                  'Giltighetsperiod': {   'Startdatum': '2007-07-01',
#                                          'link': []},
#                  'Helarsvarde': 60,
#                  'ID': '2',
#                  'Kod': 'HP',
#                  'LarosateID': -1,
#                  'link': []},
#              {   'Benamning': {   'en': 'Converted credits',
#                                     'sv': 'Konverterade högskolepoäng'},
#                  'Beskrivning': {   'sv': 'Enheten poäng konverterades '
#                                           'till högskolepoäng i Ladok för '
#                                           '1993 års studieordning i '
#                                          'samband med övergången till '
#                                          '2007 års studieordning'},
#                      'Giltighetsperiod': {   'Slutdatum': '2007-06-30',
#                                              'link': []},
#                      'Helarsvarde': 60,
#                      'ID': '9',
#                      'Kod': 'HP-K',
#                      'LarosateID': -1,
#                      'link': []},
#                  {   'Benamning': {   'en': 'Internal credits',
#                                       'sv': 'Interna poäng'},
#                      'Beskrivning': {   'sv': 'Har enbart använts av '
#                                          'Försvarshögskolan'},
#                      'Giltighetsperiod': {   'Slutdatum': '2016-07-01',
#                                              'link': []},
#                      'Helarsvarde': 60,
#                      'ID': '10',
#                      'Kod': 'IP',
#                      'LarosateID': -1,
#                      'link': []},
#                  {   'Benamning': {'en': 'KY credits', 'sv': 'KY-poäng'},
#                      'Beskrivning': {},
#                      'Giltighetsperiod': {   'Slutdatum': '2013-12-31',
#                                              'Startdatum': '2002-01-01',
#                                              'link': []},
#                      'Helarsvarde': 40,
#                      'ID': '7',
#                      'Kod': 'KYP',
#                      'LarosateID': -1,
#                      'link': []},
#                  {   'Benamning': {'en': 'Hours', 'sv': 'Timmar'},
#                      'Beskrivning': {},
#                      'Giltighetsperiod': {'link': []},
#                      'Helarsvarde': 1600,
#                      'ID': '5',
#                      'Kod': 'T',
#                      'LarosateID': -1,
#                      'link': []},
#                  {   'Benamning': {'en': 'Weeks', 'sv': 'Veckor'},
#                      'Beskrivning': {},
#                      'Giltighetsperiod': {'link': []},
#                      'Helarsvarde': 40,
#                      'ID': '1',
#                      'Kod': 'V',
#                      'LarosateID': -1,
#                      'link': []},
#                  {   'Benamning': {   'en': 'HVE credits',
#                                       'sv': 'Yrkeshögskolepoäng'},
#                      'Beskrivning': {},
#                      'Giltighetsperiod': {   'Startdatum': '2009-07-01',
#                                         'link': []},
#                  'Helarsvarde': 200,
#                  'ID': '8',
#                  'Kod': 'YHP',
#                  'LarosateID': -1,
#                  'link': []}],
# 'link': []}
```

15.30 studielokalisering_JSON

125 ⟨*LadokSession data methods* 71⟩+≡ ◁123 133▷

```
# added by GQMJr
######################################################################
#
# studielokalisering_JSON
#
# RETURNERAR JSON of study location
def studielokalisering_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/studielokalisering',
    headers=self.headers).json()
  return r

# returns:
# {   'Studielokalisering': [   {   'Benamning': {   'en': 'Botkyrka',
#                                                    'sv': 'Botkyrka'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1997-12-31',
#                                                           'Startdatum': '1997-08-01',
#                                                           'link': []},
#                                   'ID': '131772',
#                                   'Kod': '0127',
#                                   'LarosateID': 29,
#                                   'OrtID': 110990,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Haninge',
#                                                    'sv': 'Haninge'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2016-06-30',
#                                                           'Startdatum': '1994-08-01',
#                                                           'link': []},
#                                   'ID': '131773',
#                                   'Kod': '0136',
#                                   'LarosateID': 29,
#                                   'OrtID': 110991,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Nyköping',
#                                                    'sv': 'Nyköping'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2005-06-30',
#                                                           'Startdatum': '2005-01-01',
#                                                           'link': []},
#                                   'ID': '131777',
#                                   'Kod': '0480',
#                                   'LarosateID': 29,
#                                   'OrtID': 44,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Valdemarsvik',
#                                                    'sv': 'Valdemarsvik'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1999-12-31',
#                                                           'link': []},
#                                   'ID': '131780',
#                                   'Kod': '0563',
#                                   'LarosateID': 29,
#                                   'OrtID': 60,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {'en': 'Visby', 'sv': 'Visby'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2000-12-31',
#                                                           'Startdatum': '1991-08-01',
#                                                           'link': []},
#                                   'ID': '131781',
#                                   'Kod': '0980',
#                                   'LarosateID': 29,
#                                   'OrtID': 108,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Ängelholm',
#                                                    'sv': 'Ängelholm'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1997-12-31',
#                                                           'Startdatum': '1997-12-31',
#                                                           'link': []},
#                                       'ID': '131782',
#                                       'Kod': '1292',
#                                       'LarosateID': 29,
#                                       'OrtID': 154,
#                                       'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                       'method': 'GET',
#                                                       'rel': 'svenskort',
#                                                       'uri': 'https://api.ladok.se:443/kat
#                                   {   'Benamning': {'en': 'Falun', 'sv': 'Falun'},
#                                       'Beskrivning': {},
#                                       'Giltighetsperiod': {   'Slutdatum': '1998-12-31',
#                                                               'Startdatum': '1994-08-01',
#                                                               'link': []},
#                                       'ID': '131787',
#                                       'Kod': '2080',
#                                       'LarosateID': 29,
#                                       'OrtID': 263,
#                                       'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                       'method': 'GET',
#                                                       'rel': 'svenskort',
#                                                       'uri': 'https://api.ladok.se:443/kat
#                                   {   'Benamning': {'en': 'Gävle', 'sv': 'Gävle'},
#                                       'Beskrivning': {},
#                                       'Giltighetsperiod': {   'Slutdatum': '1998-12-31',
#                                                               'Startdatum': '1997-08-01',
#                                                               'link': []},
#                                       'ID': '131784',
#                                       'Kod': '2180',
#                                       'LarosateID': 29,
#                                       'OrtID': 273,
#                                       'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                       'method': 'GET',
#                                                       'rel': 'svenskort',
#                                                       'uri': 'https://api.ladok.se:443/kat
#                                   {   'Benamning': {   'en': 'Stockholm School of '
#                                                              'Economics',
#                                                        'sv': 'Handelshögskolan'},
#                                       'Beskrivning': {},
#                                       'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                               'link': []},
#                                       'ID': '135191',
#                                       'Kod': 'HANDELS',
#                                       'LarosateID': 29,
#                                       'OrtID': 18,
#                                       'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                       'method': 'GET',
#                                                       'rel': 'svenskort',
#                                                       'uri': 'https://api.ladok.se:443/kat
#                                   {   'Benamning': {   'en': 'KI Flemingsberg',
#                                                        'sv': 'KI Flemingsberg'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135192',
#                                    'Kod': 'KI_FLEMINGS',
#                                    'LarosateID': 29,
#                                    'OrtID': 7,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'University of Arts, '
#                                                           'Crafts and Design',
#                                                     'sv': 'Konstfack'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-08-27',
#                                                            'link': []},
#                                    'ID': '147750',
#                                    'Kod': 'KONSTFACK',
#                                    'LarosateID': 29,
#                                    'OrtID': 18,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'KTH Campus',
#                                                     'sv': 'KTH Campus'},
#                                    'Beskrivning': {   'sv': '"Campus" '
#                                                             'Valhallavägen'},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135195',
#                                    'Kod': 'KTHCAMPUS',
#                                    'LarosateID': 29,
#                                    'OrtID': 18,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {'en': 'Täby', 'sv': 'Täby'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Slutdatum': '1999-12-31',
#                                                            'link': []},
#                                    'ID': '131774',
#                                    'Kod': '0160',
#                                    'LarosateID': 29,
#                                    'OrtID': 14,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Stockholm',
#                                                    'sv': 'Stockholm'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2019-01-14',
#                                                           'Startdatum': '1917-10-19',
#                                                           'link': []},
#                                   'ID': '131778',
#                                   'Kod': '0180',
#                                   'LarosateID': 29,
#                                   'OrtID': 18,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Sundsvall',
#                                                    'sv': 'Sundsvall'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1997-12-31',
#                                                           'Startdatum': '1996-08-01',
#                                                           'link': []},
#                                   'ID': '131785',
#                                   'Kod': '2281',
#                                   'LarosateID': 29,
#                                   'OrtID': 281,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Örnsköldsvik',
#                                                    'sv': 'Örnsköldsvik'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1997-12-31',
#                                                           'Startdatum': '1997-01-01',
#                                                           'link': []},
#                                   'ID': '131788',
#                                   'Kod': '2284',
#                                   'LarosateID': 29,
#                                   'OrtID': 284,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'KTH Solna',
#                                                    'sv': 'KTH Solna'},
#                                   'Beskrivning': {   'en': 'SciLife Labs, '
#                                                            'Solna',
#                                                      'sv': 'SciLife Labs, '
#                                                            'Solna'},
#                                   'Giltighetsperiod': {   'Startdatum': '2018-06-18',
#                                                            'link': []},
#                                    'ID': '135610',
#                                    'Kod': 'SCILIFELAB',
#                                    'LarosateID': 29,
#                                    'OrtID': 25,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'KTH Södertälje',
#                                                     'sv': 'KTH Södertälje'},
#                                    'Beskrivning': {'sv': '"Campus" Södertälje'},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135196',
#                                    'Kod': 'SODERTALJE',
#                                    'LarosateID': 29,
#                                    'OrtID': 20,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'Järfälla',
#                                                     'sv': 'Järfälla'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Slutdatum': '1996-06-30',
#                                                            'Startdatum': '1994-08-01',
#                                                            'link': []},
#                                    'ID': '131775',
#                                    'Kod': '0123',
#                                    'LarosateID': 29,
#                                    'OrtID': 5,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'Huddinge',
#                                                     'sv': 'Huddinge'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Slutdatum': '2019-01-14',
#                                                            'Startdatum': '2002-01-01',
#                                                            'link': []},
#                                    'ID': '131779',
#                                    'Kod': '0126',
#                                    'LarosateID': 29,
#                                    'OrtID': 109830,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Södertälje',
#                                                    'sv': 'Södertälje'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2019-01-14',
#                                                           'Startdatum': '1987-01-01',
#                                                           'link': []},
#                                   'ID': '131771',
#                                   'Kod': '0181',
#                                   'LarosateID': 29,
#                                   'OrtID': 20,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Norrtälje',
#                                                    'sv': 'Norrtälje'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '2005-06-30',
#                                                           'Startdatum': '1997-08-01',
#                                                           'link': []},
#                                   'ID': '131776',
#                                   'Kod': '0188',
#                                   'LarosateID': 29,
#                                   'OrtID': 28,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {'en': 'Örebro', 'sv': 'Örebro'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1998-06-30',
#                                                           'Startdatum': '1996-08-01',
#                                                           'link': []},
#                                   'ID': '131786',
#                                   'Kod': '1880',
#                                   'LarosateID': 29,
#                                   'OrtID': 237,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Västerås',
#                                                    'sv': 'Västerås'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Slutdatum': '1999-12-31',
#                                                           'Startdatum': '1994-08-01',
#                                                           'link': []},
#                                   'ID': '131783',
#                                   'Kod': '1980',
#                                   'LarosateID': 29,
#                                    'OrtID': 248,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'AlbaNova',
#                                                     'sv': 'AlbaNova'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135190',
#                                    'Kod': 'ALBANOVA',
#                                    'LarosateID': 29,
#                                    'OrtID': 18,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'KTH Flemingsberg',
#                                                     'sv': 'KTH Flemingsberg'},
#                                    'Beskrivning': {   'sv': '"campus" '
#                                                             'Flemingsberg'},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135198',
#                                    'Kod': 'FLEMINGSB',
#                                    'LarosateID': 29,
#                                    'OrtID': 7,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'KI Solna',
#                                                     'sv': 'KI Solna'},
#                                    'Beskrivning': {},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                    'ID': '135193',
#                                    'Kod': 'KI_SOLNA',
#                                    'LarosateID': 29,
#                                    'OrtID': 25,
#                                    'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                    'method': 'GET',
#                                                    'rel': 'svenskort',
#                                                    'uri': 'https://api.ladok.se:443/kat
#                                {   'Benamning': {   'en': 'KTH Kista',
#                                                     'sv': 'KTH Kista'},
#                                    'Beskrivning': {'sv': '"Campus" Kista'},
#                                    'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                            'link': []},
#                                   'ID': '135194',
#                                   'Kod': 'KISTA',
#                                   'LarosateID': 29,
#                                   'OrtID': 18,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#                               {   'Benamning': {   'en': 'Stockholm University',
#                                                    'sv': 'Stockholms '
#                                                          'universitet'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                                           'link': []},
#                                   'ID': '135197',
#                                   'Kod': 'SU',
#                                   'LarosateID': 29,
#                                   'OrtID': 18,
#                                   'link': [   {   'mediaType': 'application/vnd.ladok+
#                                                   'method': 'GET',
#                                                   'rel': 'svenskort',
#                                                   'uri': 'https://api.ladok.se:443/kat
#     'link': []}
```

15.31 antagningsomgang_JSON

133 ⟨*LadokSession data methods* 71⟩+≡ ◁125 134▷

```
# added by GQMJr
######################################################################
#
# antagningsomgang_JSON
#
# RETURNERAR JSON of admission round
def antagningsomgang_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/antagningsomgang',
    headers=self.headers).json()
  return r

# returns:
# {   'Antagningsomgang': [   {   'Benamning': {   'en': 'Application to courses '
#                                                    'within programme at '
#                                                    'KTH HT2020',
#                                                'sv': 'Anmälan till kurs inom '
#                                                    'program på KTH HT2020'},
#                             'Beskrivning': {},
#                             'Giltighetsperiod': {'link': []},
#                             'ID': '150233',
#                              'Kod': '29AKPHT20',
#                              'LarosateID': 29,
#                              'SistaAnmalningsdag': '2020-05-15',
#                              'SistaAnnonseringsdag': '2021-05-15',
#                              'Studieavgiftsbelagd': True,
#                              'link': []},
#                         {   'Benamning': {   'en': 'Application to courses '
#                                                    'within programme at '
#                                                    'KTH VT2019',
#                                              'sv': 'Antagning till kurs '
#                                                    'inom program KTH '
#                                                    'VT2019'},
#                              'Beskrivning': {},
#                              'Giltighetsperiod': {'link': []},
#                              'ID': '142134',
#                              'Kod': '29AKPVT19',
#                              'LarosateID': 29,
#                              'SistaAnmalningsdag': '2018-11-15',
#                              'SistaAnnonseringsdag': '2019-02-28',
#                              'Studieavgiftsbelagd': True,
#                              'link': []},
# ... ],
# 'link': []}
```

15.32 organisation_by_uid_JSON

134 ⟨*LadokSession data methods* 71⟩+≡ ◁133 135▷

```
# added by GQMJr
#####################################################################
#
# organisation_by_uid_JSON
#
# organisationUid           - organization's UID
#
# RETURNERAR JSON of selected organization
def organisation_by_uid_JSON(self, organisationUid):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/organisation/'+organisationUid,
    headers=self.headers).json()
  return r

# returns:
# {   'Benamning': {'en': 'EECS/Computer Science', 'sv': 'EECS/Datavetenskap'},
# 'Giltighetsperiod': {'Startdatum': '2019-01-01', 'link': []},
# 'Organisationsenhetstyp': 1,
# 'Organisationskod': 'JH',
# 'Uid': '2474f616-dc41-11e8-8cc1-eaeeb71b497f',
# 'link': [   {   'mediaType': 'application/vnd.ladok+xml,application/vnd.ladok-katalogi
#                 'method': 'GET',
#                 'rel': 'self',
#                 'uri': 'https://api.ladok.se:443/kataloginformation/organisation/2474f
```

15.33 utbildningstyp_JSON

135 ⟨*LadokSession data methods* 71⟩+≡ ◁134 137▷

```
# added by GQMJr
#####################################################################
#
# utbildningstyp_JSON
#
# RETURNERAR JSON of types of education
# for information about these see https://ladok.se/wp-content/uploads/2018/01/Funktionsb
def utbildningstyp_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/utbildningstyp',
    headers=self.headers).json()
  return r

# returns:
# {   'Utbildningstyp': [   {   'AvserTillfalle': False,
#                               'Benamning': {   'en': 'Module without scope',
#                                                'sv': 'Modul utan omfattning'},
#                               'Beskrivning': {},
#                               'Giltighetsperiod': {'link': []},
#                               'Grundtyp': 'MODUL',
#                               'ID': '3',
#                               'Kod': '2007MUO',
#                               'LarosateID': -1,
#                               'Regelverk': {   'Regelvarden': [   {   'Regelnamn': 'co
#                                                                       'Uid': 'd31a4080
#                                                                       'Varde': 'true',
#                                                                       'link': [   ]},
#                                                                   {   'Regelnamn': 'co
#                                                                       'Uid': 'd31a4084
#                                                                       'Varde': 'true',
#                                                                       'link': [   ]},
#                                                                   {   'Regelnamn': 'co
#                                                                       'Uid': 'd31a4086
#                                                                       'Varde': 'false'
#                                                                       'link': [   ]},
#                                                                   {   'Regelnamn': 'co
#                                                                       'Uid': 'd31a4085
#                                                                       'Varde': 'true',
#                                                                       'link': [   ]},
#                                                                   {   'Regelnamn': 'co
#                                                                       'Uid': 'd31a4082
#                                                           'Varde': 'false'
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'd31a196f
#                                                           'Varde': 'grupp.
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'd31a4081
#                                                           'Varde': 'grupp.
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'd31a4083
#                                                           'Varde': 'false'
#                                                           'link': [   ]}],
#                                           'Uid': '0441f5ba-1b3f-11e6-aff0-464044
#                                           'link': []},
#                               'Sorteringsordning': 50,
#                               'StudieordningID': 1,
#                               'TillfalleInomUtbildningstyper': [],
#                               'UtbildningstyperInomUtbildningstyp': [],
#                               'link': []},
#                           {   'AvserTillfalle': False,
#                               'Benamning': {'en': 'Module', 'sv': 'Modul'},
#                               'Beskrivning': {},
#                               'Giltighetsperiod': {'link': []},
#                               'Grundtyp': 'MODUL',
#                               'ID': '4',
#                               'Kod': '2007MOD',
#                               'LarosateID': -1,
#                               'Regelverk': {   'Regelvarden': [   {   'Regelnamn': 'co
#                                                           'Uid': 'b18301be
#                                                           'Varde': 'false'
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'b18301c1
#                                                           'Varde': 'true',
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'b18301bb
#                                                           'Varde': 'true',
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'b18301bf
#                                                           'Varde': 'false'
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                           'Uid': 'b18301bd
#                                                           'Varde': 'grupp.
#                                                           'link': [   ]},
#                                                       {   'Regelnamn': 'co
#                                                         'Uid': 'b18301bc
#                                                         'Varde': 'true',
#                                                         'link': [   ]},
#                                                     {   'Regelnamn': 'co
#                                                         'Uid': 'b18301c0
#                                                         'Varde': 'grupp.
#                                                         'link': [   ]}],
#                                       'Uid': '0461b2c4-1b3f-11e6-aff0-4640446
#                                       'link': []},
#                               'Sorteringsordning': 50,
#                               'StudieordningID': 1,
#                               'TillfalleInomUtbildningstyper': [],
#                               'UtbildningstyperInomUtbildningstyp': [],
#                               'link': []},
# ...
# ],
#     'link': []}
```

15.34 aktivitetstillfallestyp_JSON

137 ⟨*LadokSession data methods* 71⟩+≡ ◁135 138▷

```
# added by GQMJr
####################################################################
#
# aktivitetstillfallestyp_JSON
#
# RETURNERAR JSON of activities
def aktivitetstillfallestyp_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/aktivitetstillfallestyp',
    headers=self.headers).json()
  return r

# returns:
# {   'Aktivitetstillfallestyp': [   {   'Benamning': {   'en': 'Partial Exam',
#                                                        'sv': 'Kontrollskrivning'},
#                                        'Beskrivning': {},
#                                        'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                        'ID': '135201',
#                                        'Kod': 'KS',
#                                        'LarosateID': 29, 'link': []},
#                                    {   'Benamning': {   'en': 'Re-examination',
#                                                        'sv': 'Omtentamen'},
#                                        'Beskrivning': {},
#                                        'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                        'ID': '135200',
#                                        'Kod': 'OMTENTA',
#                                        'LarosateID': 29, 'link': []},
#                               {   'Benamning': {   'en': 'Examination',
#                                                    'sv': 'Tentamen'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {   'Startdatum': '2018-06-20',
#                                   'ID': '135199',
#                                   'Kod': 'TENTAMEN',
#                                   'LarosateID': 29, 'link': []},
#                               {   'Benamning': {   'en': 'Unspecified '
#                                                          'activity',
#                                                    'sv': 'Övrigt '
#                                                          'aktivitetstillfälle'},
#                                   'Beskrivning': {},
#                                   'Giltighetsperiod': {'link': []},
#                                   'ID': '1',
#                                   'Kod': 'ÖV',
#                                   'LarosateID': -1, 'link': []}],
# 'link': []}
```

15.35 catalog_service_index__JSON

138 ⟨*LadokSession data methods* 71⟩+≡ ◁137 139a▷

```
# added by GQMJr
#####################################################################
#
# catalog_service_index__JSON
#
# RETURNERAR JSON of admission round
def catalog_service_index__JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + '/kataloginformation/service/index',
    headers=self.headers).json()
  return r

# returns:
# {   'ServiceName': 'Ladok3 REST-tjänst för kataloginformation',
# 'link': [   {   'mediaType': 'application/vnd.ladok+xml,application/vnd.ladok-katalogi
#                 'method': 'GET',
#                 'rel': 'http://relations.ladok.se/kataloginformation/utbildningstyp',
#                 'uri': 'https://api.ladok.se:443/kataloginformation/grunddata/utbildni
#             {   'mediaType': 'application/vnd.ladok+xml,application/vnd.ladok-katalogi
#                 'method': 'GET',
#                 'rel': 'http://relations.ladok.se/kataloginformation/betygsskala',
#                 'uri': 'https://api.ladok.se:443/kataloginformation/grunddata/betygssk
# ...
#             {   'mediaType': 'application/vnd.ladok+xml,application/vnd.ladok-katalogi
#                 'method': 'GET',
#                 'rel': 'http://relations.ladok.se/kataloginformation/anvandarbehorighe
#                 'uri': 'https://api.ladok.se:443/kataloginformation/behorigheter'}]}
```

15.36 omradesbehorighet_JSON

139a ⟨*LadokSession data methods* 71⟩+≡ ◁138 139b▷
```
# added by GQMJr
######################################################################
#
# omradesbehorighet_JSON
#
# RETURNERAR JSON of "omradesbehorighet"
# for information see https://antagning.se/globalassets/omradesbehorigheter-hogskolan.pd
def omradesbehorighet_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/kataloginformation/grunddata/omradesbehorighet',
    headers=self.headers).json()
  return r
```

15.37 hamtaStudieResultatForStudent_JSON

139b ⟨*LadokSession data methods* 71⟩+≡ ◁139a 139c▷
```
# added by GQMJr
######################################################################
#
# hamtaStudieResultatForStudent_JSON
#
# studentUID             - student's UID
# RETURNERAR JSON of results

# NOTE: These are a work in progress and not ready yet
# def hamtaStudieResultatForStudent_JSON (self, studentUID):
#     r = self.session.get(url = self.base_gui_proxy_url + '/resultat/studieresultat/res
#     return r

# def student_participation_JSON (self, studentUID):
#     headers = self.headers.copy()
#     headers['Content-Type'] = 'application/vnd.ladok-studiedeltagande'
#     headers['Accept'] = headers['Accept']+', application/vnd.ladok-studiedeltagande'
#     r = self.session.get(url = self.base_gui_proxy_url + '/studiedeltagande/tillfalles
#     return {r.status_code, r.text}
```

15.38 examen_student_uid_JSON

139c ⟨*LadokSession data methods* 71⟩+≡ ◁139b 140▷
```
# added by GQMJr
######################################################################
#
# examen_student_uid_JSON
```

```
#
# studentUID             - student's UID
# RETURNERAR JSON of admission round
def examen_student_uid_JSON(self):
  r = self.session.get(
    url=self.base_gui_proxy_url + 'examen/student/+studentUID',
    headers=self.headers).json()
  return r
```

15.39 Helper methods

140 ⟨*LadokSession data methods* 71⟩+≡ ◁139c

```
##################################################################
##
## private methods
##

def __get_xsrf_token(self):
  cookies = self.session.cookies.get_dict()
  return next(cookies[cookie] for cookie in cookies if cookie == 'XSRF-TOKEN')

def get_xsrf_token(self):
  return self.__get_xsrf_token()


# returns None or a LADOK-formated date
def __validate_date(self, date_raw):
  datregex = re.compile(r"(\d\d)?(\d\d)-?(\d\d)-?(\d\d)")
  dat = datregex.match(date_raw)
  if dat:
    if dat.group(1) == None: # add 20, ladok3 won't survive till 2100
      return "20" + dat.group(2) + "-" + dat.group(3) + "-" + dat.group(4)
    else:
      return dat.group(1) + dat.group(2) + \
        "-" + dat.group(3) + "-" + dat.group(4)
  else:
    return None

def __get_grade_scale_by_id(self, grade_scale_id):
  return next(grade_scale
    for grade_scale in self.get_grade_scales()
      if grade_scale.id == grade_scale_id)


def __get_grade_scale_by_code(self, grade_scale_code):
  return next(grade_scale
    for grade_scale in self.get_grade_scales()
      if grade_scale.code == grade_scale_code)
```

```
def __get_grade_by_id(self, grade_id):
  for grade_scale in self.get_grade_scales():
    for grade in grade_scale.grades():
      if grade.id == grade_id:
        return grade

  return None


def __get_student_data(self, person_nr):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/studentinformation/student/filtrera?limit=2&orderby=EFTERNAMN_ASC&orderby=FORNAM
        + person_nr + '&skipCount=false&sprakkod=sv',
    headers=self.headers).json()['Resultat']

  if len(r) != 1: return None

  r = r[0]
  # from schemas/schemas.ladok.se-studentinformation.xsd
  #   <xs:complexType name="Student">
  #   <xs:complexContent>
  #     <xs:extension base="base:BaseEntitet">
  #       <xs:sequence>
  #         <xs:element name="Avliden" type="xs:boolean"/>
  #         <xs:element minOccurs="0" name="Efternamn" type="xs:string"/>
  #         <xs:element minOccurs="0" name="ExterntUID" type="xs:string"/>
  #         <xs:element name="FelVidEtableringExternt" type="xs:boolean"/>
  #         <xs:element minOccurs="0" name="Fodelsedata" type="xs:string"/>
  #         <xs:element minOccurs="0" name="Fornamn" type="xs:string"/>
  #         <xs:element minOccurs="0" name="KonID" type="xs:int"/>
  #         <xs:element minOccurs="0" name="Personnummer" type="xs:string"/>
  #         <xs:element minOccurs="0" name="Skyddsstatus" type="xs:string"/>
  #         <xs:element minOccurs="0" ref="si:UnikaIdentifierare"/>
  #       </xs:sequence>
  #     </xs:extension>
  #   </xs:complexContent>
  # </xs:complexType>

  return {
    'id':         r['Uid'], # Ladok-ID
    'first_name': r['Fornamn'],
    'last_name':  r['Efternamn'],
    'person_nr':  r['Personnummer'], # tolv siffror, utan bindestreck eller plustecken
    'alive':  not r['Avliden']
  }

# detta är egentligen kurstillfällen, inte kurser (ID-numret är alltså ett ID-nummer för
def __get_student_courses(self, student_id):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/studiedeltagande/tillfallesdeltagande/kurstillfallesdeltagande/student/'
        + student_id,
    headers=self.headers).json()

  results = []

  for course in r['Tillfallesdeltaganden']:
    if not course['Nuvarande'] or \
       'Utbildningskod' not in course['Utbildningsinformation']:
      continue

    results.append({
      'id': course['Uid'],
      'round_id': course['Utbildningsinformation']['UtbildningstillfalleUID'], # ett Lad
      'education_id': course['Utbildningsinformation']['UtbildningUID'], # ett Ladok-ID
      'instance_id': course['Utbildningsinformation']['UtbildningsinstansUID'], # ett La
      'code': course['Utbildningsinformation']['Utbildningskod'], # kurskod KOPPS
      'name': course['Utbildningsinformation']['Benamning']['sv']
    })

  return results


def __get_student_course_moments(self, course_round_id, student_id):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/resultat/kurstillfalle/' + str(course_round_id) +
        '/student/' + str(student_id) + '/moment',
    headers=self.headers).json()

  return [{
    'course_moment_id': moment['UtbildningsinstansUID'],
    'code': moment['Utbildningskod'],
    'education_id': moment['UtbildningUID'],
    'name': moment['Benamning']['sv']
  } for moment in r['IngaendeMoment']]


def __get_student_course_results(self, course_round_id, student_id):
  r = self.session.get(
    url=self.base_gui_proxy_url +
      '/resultat/studieresultat/student/' + student_id +
        '/utbildningstillfalle/' + course_round_id,
    headers=self.headers).json()

  return {
    'id': r['Uid'],
    'results': [{
      'education_id': result['UtbildningUID'],
      'pending': {
        'id': result['Arbetsunderlag']['Uid'],
        'moment_id': result['Arbetsunderlag']['UtbildningsinstansUID'],
        'grade': self.__get_grade_by_id(result['Arbetsunderlag']['Betygsgrad']),
        'date': result['Arbetsunderlag']['Examinationsdatum'],
        'grade_scale': self.__get_grade_scale_by_id(result['Arbetsunderlag']['Betygsskal
        # behövs vid uppdatering av betygsutkast
        'last_modified': result['Arbetsunderlag']['SenasteResultatandring']
      } if 'Arbetsunderlag' in result else None,
      'attested': {
        'id': result['SenastAttesteradeResultat']['Uid'],
        'moment_id': result['SenastAttesteradeResultat']['UtbildningsinstansUID'],
        'grade': self.__get_grade_by_id(result['SenastAttesteradeResultat']['Betygsgrad']
        'date': result['SenastAttesteradeResultat']['Examinationsdatum'],
        'grade_scale': self.__get_grade_scale_by_id(result['SenastAttesteradeResultat'][
      } if 'SenastAttesteradeResultat' in result else None
    } for result in r['ResultatPaUtbildningar']]
  }
```

Part IV

A command-line interface

Chapter 16

The base interface

This is the documentation of the command-line interface module (`cli`) of the `ladok3` package, ⟨*cli.py* 145⟩.

The command-line interface is divided into subcommands, similar to Git. Currently, we provide the following commands:

data The command outputs all results for all rounds of a course in CSV format.

16.1 Overview of the source code and dependencies

We use `argparse` with `argcomplete` to handle the command-line interface. We use `appdirs` to handle configuration directories on various systems, we set up a global `dirs` that we can use.

145 ⟨*cli.py* 145⟩≡

```
#!/bin/env python3
"""A command-line interface for LADOK 3"""

import appdirs
import argcomplete, argparse
import base64
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
import getpass
import json
import keyring
import ladok3.kth
import os
import pickle
import re
import sys
import traceback

⟨modules 152b⟩
```

```
dirs = appdirs.AppDirs("ladok", "dbosk@kth.se")

⟨functions 146b⟩
⟨command-line interface 146a⟩
```

16.2 A command-line interface

The idea is that we have the basic command that can take some options and ultimately some subcommand. The subcommands do the heavy lifting. We use a standard Pythonic if-main construction.

146a ⟨*command-line interface* 146a⟩≡ (145)

```
def main():
  """Run the command-line interface for the ladok command"""
  ⟨process command-line options 146c⟩

if __name__ == "__main__":
  try:
    main()
    sys.exit(0)
  except Exception as e:
    err(-1, e)
```

We want uniform error handling. We will use the function `err` for errors and `warn` for warnings, both inspired by err(3) and warn(3) in the BSD world.

146b ⟨*functions* 146b⟩≡ (145) 148a ▷

```
def err(rc, msg):
  print(f"{sys.argv[0]}: error: {msg}", file=sys.stderr)
  sys.exit(rc)

def warn(msg):
  print(f"{sys.argv[0]}: {msg}", file=sys.stderr)
```

16.2.1 Process command-line options

The main part done here is to set up a `LadokSession` object `ls`. We can do this in two ways: (1) create a new object using the user's credentials to log in; or (2) restore the `ls` object from a previous run.

We'll use `argparse` to parse the command-line options.

146c ⟨*process command-line options* 146c⟩≡ (146a)

```
argp = argparse.ArgumentParser(
  description="This is a CLI-ification of LADOK3's web GUI.",
  epilog="Web: https://github.com/dbosk/ladok3"
)
⟨add global configuration options 149b⟩
subp = argp.add_subparsers(
  title="commands",
  dest="command",
  required=True
)
⟨add subparsers to subp 149c⟩
argcomplete.autocomplete(argp)
args = argp.parse_args()
⟨create or restore the LadokSession ls 147b⟩
⟨execute subcommand 147a⟩
⟨save LadokSession ls 147c⟩
```

For each subcommand, we will add a subparser (`subp.add_parser`) that will set the `func` attribute. Then we can execute the correct function and let that function check the remaining arguments. We must also pass on a LADOK session object `ls`.

147a ⟨*execute subcommand* 147a⟩≡ (146c)

```
  if "func" in args:
    args.func(ls, args)
```

16.3 Create the LadokSession `ls`

We have several options for creating the LadokSession object `ls` that must be passed to the subcommand:

(1) The first one is to read the user's credentials (from the keyring, environment or configuration) and create the `ls` object by instantiating the `LadokSession` class (technically the `ladok3.kth.LadokSession` class).

(2) Read a pickled `LadokSession` object from file.

The second option allows us to save time by not having to reestablish an authenticated session to LADOK. The authentication step takes much more time that individual LADOK requests.

This leads to the following approach.

147b ⟨*create or restore the LadokSession ls* 147b⟩≡ (146c)

```
  LADOK_USER, LADOK_PASS = load_credentials(args.config_file)
  ls = restore_ladok_session([LADOK_USER, LADOK_PASS])
  if not ls:
    ls = ladok3.kth.LadokSession(LADOK_USER, LADOK_PASS)
```

147c ⟨*save LadokSession ls* 147c⟩≡ (146c)

```
  store_ladok_session(ls, [LADOK_USER, LADOK_PASS])
```

16.3.1 Saving and restoring the `LadokSession` object

Now we need to implement the `store_ladok_session` and `restore_ladok_session` functions. We will use `pickle` for storing and restoring the object. But we also want to encrypt the stored object.

148a ⟨*functions* 146b⟩+≡ (145) ◁146b 150a▷

```
def store_ladok_session(ls, credentials):
  if not os.path.isdir(dirs.user_cache_dir):
    os.mkdir(dirs.user_cache_dir)

  file_path = dirs.user_cache_dir + "/LadokSession"

  pickled_ls = pickle.dumps(ls)
  ⟨encrypt pickled ls 148b⟩

  with open(file_path, "wb") as file:
    file.write(encrypted_ls)

def restore_ladok_session(credentials):
  file_path = dirs.user_cache_dir + "/LadokSession"

  if os.path.isfile(file_path):
    with open(file_path, "rb") as file:
      encrypted_ls = file.read()
      ⟨decrypt encrypted ls 148c⟩
      return pickle.loads(pickled_ls)

  return None
```

We don't want to only encrypt the object, we also want to provide integrity for the object. This is to avoid any vulnerabilities with `pickle`. We use the `cryptography` module to handle the encryption. In particular, we use the `cryptography.fernet` protocol which encrypts using AES and signs using HMAC.

We will use the user's LADOK password to generate a cryptographic key. Then we do the encryption/decryption.

148b ⟨*encrypt pickled ls* 148b⟩≡ (148a)

```
⟨set up kdf and derive key from credentials 149a⟩

fernet_protocol = Fernet(key)
encrypted_ls = fernet_protocol.encrypt(pickled_ls)
```

148c ⟨*decrypt encrypted ls* 148c⟩≡ (148a)

```
⟨set up kdf and derive key from credentials 149a⟩

fernet_protocol = Fernet(key)
try:
  pickled_ls = fernet_protocol.decrypt(encrypted_ls)
except Exception as err:
  warn(f"cache was corrupted, cannot decrypt: {err}")
  pickled_ls = None
```

To actually do the key derivation, we use the PBKDF2 based on HMAC. We use the LADOK username as salt and the password as password.

149a ⟨*set up kdf and derive key from credentials* 149a⟩≡ (148)
```
kdf = PBKDF2HMAC(
  algorithm=hashes.SHA256(),
  length=32,
  salt=credentials[0].encode("utf-8"),
  iterations=100000
)
key = base64.urlsafe_b64encode(kdf.derive(credentials[1].encode("utf-8")))
```

16.4 Configuration

We provide a global configuration file. We want to allow the user to specify different locations for the configuration.

Location of configuration files We want the user to be able to specify the location of the configuration file.

149b ⟨*add global configuration options* 149b⟩≡ (146c)
```
argp.add_argument("-f", "-config-file",
  default=f"{dirs.user_config_dir}/config.json",
  help="Path to configuration file "
    f"(default: {dirs.user_config_dir}/config.json) "
    "or set LADOK_USER and LADOK_PASS environment variables.")
```
Since we provide the default value here, we can always rely on it to be available, so we can use `args.config_file` directly when we want to access the configuration file.

16.5 Managing credentials

We want a subcommand to handle the user's credentials for accessing LADOK. In particular, we need the user to be able to change the credentials in the system keyring, e.g., in case the user wrote the wrong password. The rest we don't need to do much about, merely point out the possibilities to the user.

149c ⟨*add subparsers to subp* 149c⟩≡ (146c) 151a▷
```
login_parser = subp.add_parser("login",
  help="Manage login credentials",
  description="""
Manages the user's LADOK login credentials (only credentials at KTH supported
right now). There are three ways to supply the login credentials, in order of
priority:

1) Through the system keyring: Just run `ladok login` and you'll be asked to
   enter the credentials and they will be stored in the keyring.

2) Through the environment: Just set the environment variables LADOK_USER and
   LADOK_PASS.

3) Through the configuration file: Just write
```

```
      {
        "username": "the actual username",
        "password": "the actual password"
      }

    to the file """ + dirs.user_config_dir + """/config.json (default, or use
    the -f option).
"""
)
login_parser.set_defaults(func=update_credentials_in_keyring)
```

16.5.1 Updating the credentials in the keyring

As stated, if the subcommand is run, we should update the credentials in the keyring. If we run this subcommand, also want to clear the cache; otherwise, the cache will keep the outdated credentials.

150a ⟨*functions* 146b⟩+≡ (145) ◁148a 150b▷

```
def update_credentials_in_keyring(ls, args):
  user = input("LADOK username: ")
  passwd = getpass.getpass("LADOK password: [input is hidden]")

  keyring.set_password("ladok3", "username", user)
  keyring.set_password("ladok3", "password", passwd)

  clear_cache(ls, args)
```

16.5.2 Loading user credentials

The `load_credentials` function will try to get the user's LADOK credentials. There are three locations:

(1) the system keyring,

(2) the environment variables `LADOK_USER` and `LADOK_PASS`,

(3) the configuration file.

They are given the priority they are listed in above. It tries to fetch the credentials in the given order; if it succeeds, it returns those credentials, otherwise it tries the next. If all fail, the function will return `None` for both. (This is due to how we handle the `login` command.)

150b ⟨*functions* 146b⟩+≡ (145) ◁150a 152a▷

```
def load_credentials(filename="config.json"):
  """Load credentials from environment or file named filename"""
  try:
    user = keyring.get_password("ladok3", "username")
    passwd = keyring.get_password("ladok3", "password")
    if user and passwd:
      return user, passwd
  except:
    pass

  try:
    user = os.environ["LADOK_USER"]
    passwd = os.environ["LADOK_PASS"]
    return user, passwd
  except:
    pass

  try:
    with open(filename) as conf_file:
      config = json.load(conf_file)
    return config["username"], config["password"]
  except:
    pass

  return None, None
```

16.6 Managing the cache

We need a command to control the cache. For this purpose we provide the `cache` subcommand. This command will in turn have subcommands.

151a ⟨*add subparsers to subp* 149c⟩+≡ (146c) ◁149c 152c▷

```
cache_parser = subp.add_parser("cache",
  help="Manage cache",
  description="Manages the cache of LADOK data"
)
cache_subp = cache_parser.add_subparsers(
  title="subcommands",
  dest="subcommand",
  required=True
)
⟨add subcommands to cache command 151b⟩
```

16.6.1 Clear the cache

Since the cache stores sensitive data (student data) and the login credentials, we want to be able to clear the cache.

151b ⟨*add subcommands to cache command* 151b⟩≡ (151a)

```
cache_clear = cache_subp.add_parser("clear",
  help="Clear the cache",
  description="Clears everything from the cache"
)
cache_clear.set_defaults(func=clear_cache)
```

The `clear_cache` function will clear the cache. We simply remove the existing cache file and then exit. If we don't exit using `sys.exit`, the main program will write the cache back again on its exit.

152a ⟨*functions* 146b⟩+≡ (145) ◁150b

```
def clear_cache(ls, args):
  try:
    os.remove(dirs.user_cache_dir + "/LadokSession")
  except FileNotFoundError as err:
    pass

  sys.exit(0)
```

16.7 Other subcommands

To add a subcommand we must add a subparser to the `subp` parser object from above, we do this in ⟨*add subparsers to subp* 149c⟩. Every subcommand exists in a separate module and that module provides the function `add_command_options` that takes `subp` as an argument.

We add the `data` subcommand.

152b ⟨*modules* 152b⟩≡ (145) 152d▷

```
import ladok3.data
```

152c ⟨*add subparsers to subp* 149c⟩+≡ (146c) ◁151a 152e▷

```
ladok3.data.add_command_options(subp)
```

We add the `report` command.

152d ⟨*modules* 152b⟩+≡ (145) ◁152b 152f▷

```
import ladok3.report
```

152e ⟨*add subparsers to subp* 149c⟩+≡ (146c) ◁152c 152g▷

```
ladok3.report.add_command_options(subp)
```

We add the `student` command.

152f ⟨*modules* 152b⟩+≡ (145) ◁152d

```
import ladok3.student
```

152g ⟨*add subparsers to subp* 149c⟩+≡ (146c) ◁152e

```
ladok3.student.add_command_options(subp)
```

Chapter 17

The `data` command

We want to compute statistics based on LADOK data.We want to answer the following questions:

(1) What are the grade distribution of a course round?

(2) How do grade distributions differ between years (i.e., rounds) for the same course?

(3) How do grade distributions differ between (similar) courses?

(4) How is the throughput distribution of a course?

(5) How do throughput distributions vary between years?

We provide the module `data` (⟨*data.py* 155a⟩) to extract the necessary data from LADOK (Section 17.1). This is a subcommand for the `ladok` command-line interface. It can be used like this:

```
ladok data DD1315 > DD1315.csv
```

This program will produce CSV-formated data to answer the questions above. The data is formated like this:

- course,
- round,
- component,
- student (pseudonym),
- grade,
- normalized time.

Or more concretely, we can use it like this:

```python
import numpy as np
import pandas as pd

prgi = pd.read_csv("DD1315.csv")
```

```

rounds_years = {
  51386: "2020",
  50869: "2019",
  50662: "2018",
  50650: "2017",
  50523: "2016"
}
prgi_LAB3 = prgi[
  prgi.Component.eq("LAB3") & prgi.Round.isin(rounds_years.keys())
].replace(rounds_years)

stats = pd.crosstab(prgi_LAB3.Grade, prgi_LAB3.Round)

for column in stats:
  stats[column] /= stats[column].sum()

print(stats.round(3).to_latex(
  label="GradeDistribution",
  caption="Grade distribution for the DD1315 course."
))
```

This yields ??.

?? PythonTeX ??

If we convert the grades to numbers, then we can also compute the average grade with standard deviation etc.

```
grade_map = {
  "A": 5, "B": 4, "C": 3, "D": 2, "E": 1, "-": 0
}
grades_num = prgi_LAB3[ ["Round", "Grade"] ].replace(grade_map)

print(grades_num.groupby("Round")["Grade"].describe().round(3).to_latex(
  label="GradeStats",
  caption="Statistics for grades of the DD1315 course, grade map: " +
    ", ".join([f"{g} $\mapsto$ {s}" for g, s in grade_map.items()]) + "."
))
```

This yields ??.

?? PythonTeX ??

Finally, we'd like to merge the old rounds to compare with the latest.

```
rounds_avg = {
  "2019": "2019--2016",
  "2018": "2019--2016",
  "2017": "2019--2016",
  "2016": "2019--2016"
}
grades_avg = grades_num.replace(rounds_avg)

print(grades_avg.groupby("Round")["Grade"].describe().round(3).to_latex(
  label="GradeStatsAvg",
  caption="Statistics for grades of the DD1315 course, grade map: " +
    ", ".join([f"{g} $\mapsto$ {s}" for g, s in grade_map.items()]) + "."
))
```

This yields ??.
?? PythonTeX ??

17.1 The `data` subcommand

This is a subcommand run as part of the `ladok3.cli` module. We provide a function `add_command_options` that adds the subcommand options to a given parser. We need access to LADOK through the `ladok3` module. We will also write data to disk in CSV form, so we need the `csv` module.

155a ⟨*data.py* 155a⟩≡

```
import csv
import datetime
import ladok3
import os
import sys

⟨functions 156b⟩

def add_command_options(parser):
  ⟨add data parser to parser 155b⟩
  ⟨add data command arguments to data parser 156a⟩

def command(ladok, args):
  ⟨produce data about course specified in args 155c⟩
```

17.1.1 Setting up the data command options

We add a subparser. We set it up to use the function `command`.

155b ⟨*add data parser to parser* 155b⟩≡ (155a)

```
data_parser = parser.add_parser("data",
  help="Returns course results data in CSV form",
  description="""
Returns the results in CSV form for all first-time registered students.
""".strip())
data_parser.set_defaults(func=command)
```

17.2 Producing data

We fetch the data from LADOK and print it in CSV format to standard out (`sys.stdout`). This way the user can deal with how to store the data.

155c ⟨*produce data about course specified in args* 155c⟩≡ (155a)

```
data_writer = csv.writer(sys.stdout, delimiter=args.delimiter)
```

```
course_rounds = filter_rounds(
  ladok.search_course_rounds(code=args.course_code),
  args.rounds)

data_writer.writerow([
  "Course", "Round", "Component", "Student", "Grade", "Time"
])
for course_round in course_rounds:
  data = extract_data_for_round(ladok, course_round, args)

  for student, component, grade, time in data:
    data_writer.writerow(
      [course_round.code, course_round.round_code, component,
        student, grade, time]
    )
```

We must take a course code and a delimiter as arguments.

156a ⟨*add data command arguments to data parser* 156a⟩≡ (155a) 156c ▷

```
data_parser.add_argument("course_code",
  help="The course code of the course for which to export data")

data_parser.add_argument("-d", "-delimiter",
  default="\t",
  help="The delimiter for the CSV output; "
    "default is a tab character to be compatible with POSIX commands, "
    "use ‘-d,‘ or ‘-d ,‘ to get comma-separated values.")
```

We filter the rounds.

156b ⟨*functions* 156b⟩≡ (155a) 157a ▷

```
def filter_rounds(all_rounds, desired_rounds):
  """Returns only the round objects with round code in desired_rounds."""
  if not desired_rounds:
    return all_rounds
  return filter(
    lambda x: x.round_code in desired_rounds,
    all_rounds
  )
```

We need a list of desired rounds.

156c ⟨*add data command arguments to data parser* 156a⟩+≡ (155a) ◁ 156a 159a ▷

```
data_parser.add_argument("-r", "-rounds", nargs="+",
  help="The round codes for the rounds to include, "
    "otherwise all rounds will be included.")
```

17.3 Extracting data for a round

Now we want to extract data for a given course round. This is done by the function `extract_data_for_round`. We need access to LADOK through the `ladok` object of type `LadokSession`. We also need the course round through the `course_round` object of type `CourseRound`.

157a ⟨*functions* 156b⟩+≡ (155a) ◁156b 158a▷

```
def extract_data_for_round(ladok, course_round, args):
  ⟨compute start and length of the course 157b⟩
  ⟨get the results for the course round 157c⟩

  students = filter_students(course_round.participants(), args.students)

  for student in students:
    student_results = filter_student_results(student, results)

    ⟨determine if student should be included 160a⟩

    components = filter_components(course_round.components(), args.components)

    for component in components:
      if len(student_results) < 1:
        result_data = None
      else:
        result_data = filter_component_result(
          component, student_results[0]["ResultatPaUtbildningar"])

      if not result_data:
        grade = "-"
        normalized_date = None
      else:
        ⟨extract grade and normalized date from result data 158e⟩

      yield student, component, grade, normalized_date
```

17.3.1 Get round data

We need the start of the course and the length to be able to normalize the dates for the grades.

157b ⟨*compute start and length of the course* 157b⟩≡ (157a)

```
course_start = course_round.start
course_length = course_round.end - course_start
```

We must get the results for the course round from LADOK. For this we must use an instance ID of a component. However, LADOK returns the results for all components, not just the one requested for.

157c ⟨*get the results for the course round* 157c⟩≡ (157a)

```
component = course_round.components()[0]
results = ladok.search_reported_results_JSON(
  course_round.round_id, component.instance_id)
```

Now, we don't iterate over these results. We iterate over the students and the components of a course round. LADOK doesn't report 'none results'. But we want to have a result showing that a student hasn't done anything, that should affect the statistics. Then we must search for a student's result in the batch of results we received from LADOK.

158a ⟨*functions* 156b⟩+≡ (155a) ◁157a 158b▷

```
def filter_student_results(student, results):
  return list(filter(
    lambda x: x["Student"]["Uid"] == student.ladok_id,
    results))
```

Similarly, we want to find the result for a particular component.

158b ⟨*functions* 156b⟩+≡ (155a) ◁158a 159b▷

```
def filter_component_result(component, results):
  for component_result in results:
    ⟨get the component result data 158c⟩
    ⟨check component code in result data 158d⟩
    return result_data

  return None
```

Depending on whether the data is attested or not, we can get the actual grade and date from two different substructures: 'Arbetsunderlag' are results in LADOK that have been entered, but not attested; 'SenastAttesteradeResultat' are results that have been attested. They both have the same structure.

158c ⟨*get the component result data* 158c⟩≡ (158b)

```
if "Arbetsunderlag" in component_result:
  result_data = component_result["Arbetsunderlag"]
elif "SenastAttesteradeResultat" in component_result:
  result_data = component_result["SenastAttesteradeResultat"]
else:
  continue
```

The results only refer to the component's instance ID, so we must match the component on that ID. The `course_round` object allows us to do exactly that with the `components` method. We note that we can ignore the grade on the whole course, since that one is determined by the other components.

158d ⟨*check component code in result data* 158d⟩≡ (158b)

```
if component.instance_id != result_data["UtbildningsinstansUID"]:
  continue
```

Finally, if there is a grade, we can extract the grade and compute the normalized date. However, if there are no results, we set `None`. We also only consider grades finished during the course. I.e., if the student has finished after the course ended, we set it as not finished. Otherwise, we cannot compare with earlier courses, since then students have had the chance to finish, so we want to see how many didn't finish on time.

158e ⟨*extract grade and normalized date from result data* 158e⟩≡ (157a)

```
if "Betygsgradsobjekt" in result_data:
  grade = result_data["Betygsgradsobjekt"]["Kod"]
  date = datetime.date.fromisoformat(
    result_data["Examinationsdatum"])
  normalized_date = (date - course_start) / course_length
  if args.time_limit and normalized_date > args.time_limit:
    grade = "-"
    normalized_date = None
else:
  grade = "-"
  normalized_date = None
```

We must add the command line argument for the limit for the normalized date.

159a ⟨*add data command arguments to data parser* 156a⟩+≡ (155a) ◁156c 159c▷

```
  data_parser.add_argument("-t", "-time-limit", type=float,
    help="The time (normalized) for cutting off results, "
      "use ‘-t 1.0‘ to cut off at course end.")
```

17.3.2 Filtering data

We might not want all data, but specify on the command-line which data to keep. We want to filter on students and components.

159b ⟨*functions* 156b⟩+≡ (155a) ◁158b 160b▷

```
  def filter_students(all_students, desired_students):
    """Returns only the students with personnummer in desired_students."""
    if not desired_students:
      return all_students
    return filter(
      lambda x: x.personnummer in desired_students,
      all_students
    )

  def filter_components(all_components, desired_components):
    """Returns only the components with a code in the desired_components."""
    if not desired_components:
      return all_components
    return filter(
      lambda x: x.code in desired_components,
      all_components
    )
```

Now, we must also add the suitable command-line options.

159c ⟨*add data command arguments to data parser* 156a⟩+≡ (155a) ◁159a

```
  data_parser.add_argument("-s", "-students", nargs="+",
    help="List of personnummer for students to include, "
      "otherwise all students will be included.")

  data_parser.add_argument("-c", "-components", nargs="+",
    help="List of component codes for components to include, "
      "otherwise all components will be included.")
```

17.4 Which students to exclude

However, we don’t want to include all students. We check if a student should be included or not.

160a ⟨*determine if student should be included* 160a⟩≡ (157a)
```
  if not should_include(ladok, student, course_round, student_results):
    continue
```

We want to filter out some values from the data. We only want to keep students who are registered on the course the first time and who doesn't have any credit transfer on the course.

160b ⟨*functions* 156b⟩+≡ (155a) ◁159b 160c▷
```
  def should_include(ladok, student, course_round, result):
    """Returns True if student should be included, False if to be excluded"""
    if is_reregistered(ladok, student.ladok_id, course_round):
      return False

    if has_credit_transfer(result):
      return False

    return True
```

A student should be counted on the first round they were registered on. We check if a student is reregistered by checking if the course round is the first round the student was registered on.

160c ⟨*functions* 156b⟩+≡ (155a) ◁160b 160d▷
```
  def is_reregistered(ladok, student_id, course):
    """Check if the student is reregistered on the course round course."""
    registrations = ladok.registrations_on_course_JSON(
      course.education_id, student_id)
    registrations.sort(
      key=lambda x: x["Utbildningsinformation"]["Studieperiod"]["Startdatum"])
    first_reg = registrations[0]
    return first_reg["Utbildningsinformation"]["Utbildningstillfalleskod"] != \
      course.round_code
```

If the student has a credit transfer for any part of the course, we should exclude the student.

160d ⟨*functions* 156b⟩+≡ (155a) ◁160c
```
  def has_credit_transfer(results):
    """Returns True if there exists a credit tranfer among the results."""
    for result in results:
      for component_result in result["ResultatPaUtbildningar"]:
        if component_result["HarTillgodoraknande"]:
          return True

    return False
```

Chapter 18

The `report` command

We want to report results from the command line.We need the following data for each result:

- course identifier,
- course component,
- student identifier,
- grade, and
- date[1].

We provide two ways to do this: The first is to provide positional arguments on the command line. The second is to not provide any positinoal argument, and in this case we read CSV data from standard input.

18.1 The `report` subcommand

This is a subcommand is run as part of the `ladok3.cli` module. We provide a function `add_command_options` that adds the subcommand options to a given parser. We need access to LADOK through the `ladok3` module.

161 ⟨*report.py* 161⟩≡
```
import csv
import datetime
import ladok3
import sys

⟨functions 162b⟩

def add_command_options(parser):
  report_parser = parser.add_parser("report",
    help="Reports course results to LADOK",
    description="Reports course results to LADOK"
  )
```

[1]Optional, we set today's date if not present.

```
    report_parser.set_defaults(func=command)
    ⟨add report command arguments to report parser 162a⟩

  def command(ladok, args):
    ⟨report results depending on args 165⟩
```

As mentioned above, we provide two ways of doing this.

162a ⟨*add report command arguments to report parser* 162a⟩≡ (161) 162c ▷

```
  one_parser = report_parser.add_argument_group(
    "one result as positional args, only date is optional")
  ⟨add one result group arguments 162d⟩
  many_parser = report_parser.add_argument_group(
    "many results read from standard input as CSV, columns: "
    "course, component, student, grade, date.")
  ⟨add many results group arguments 164b⟩
```

18.2 Report the results

We provide two functions, one for each way of reporting the results.

162b ⟨*functions* 162b⟩≡ (161)

```
  def report_one_result(ladok, args):
    ⟨report results given in args 163c⟩

  def report_many_results(ladok, args):
    ⟨report results given in stdin 164a⟩
```

In both cases, we want to specify whether or not the results should be finalized. We add an option whether we want to finalize the grade for attestation by the examiner. We add this argument outside the two groups, since it's valid for both.

162c ⟨*add report command arguments to report parser* 162a⟩+≡ (161) ◁ 162a

```
  report_parser.add_argument("-f", "-finalize",
    help="Finalize the grade (mark as ready/klarmarkera) for examiner to attest",
    action="store_true",
    default=False
  )
```

18.3 Report a result given on command line

If we've chosen to give one result on the command line, then we'll need the following arguments.

We start with the course, component code, the student's ID and grade.

162d ⟨*add one result group arguments* 162d⟩≡ (162a) 163b ▷

```
  one_parser.add_argument("course_code", nargs="?",
    help="The course code (e.g. DD1315) for which the grade is for"
  )

  one_parser.add_argument("component_code", nargs="?",
    help="The component code (e.g. LAB1) for which the grade is for"
  )

  one_parser.add_argument("student_id", nargs="?",
    help="Student identifier (personnummer or LADOK ID)"
  )

  one_parser.add_argument("grade", nargs="?",
    help="The grade (e.g. A or P)"
  )
```

We must make them optional like this to make it work with out second alternative, so we must check ourselves that we got the arguments.

163a ⟨*check that we got all positional arguments* 163a⟩≡ (163c)

```
  if not (args.course_code and args.component_code and
    args.student_id and args.grade):
    print(f"{sys.argv[0]} report: "
      "not all positional args given: course_code, component, student, grade",
      file=sys.stderr)
    sys.exit(1)
```

Next, we have the date. To ensure it's a valid date we'll make `argparse` convert it to `datetime` format. If it's not provided, we let `argparse` set it to today's date.

163b ⟨*add one result group arguments* 162d⟩+≡ (162a) ◁162d

```
  one_parser.add_argument("date", nargs="?",
    help="Date on ISO format (e.g. 2021-03-18), "
      f"defaults to today's date ({datetime.date.today()})",
    type=datetime.date.fromisoformat,
    default=datetime.date.today()
  )
```

Now that we have the arguments, we can just execute the following code using them.

163c ⟨*report results given in args* 163c⟩≡ (162b)

```
  ⟨check that we got all positional arguments 163a⟩
  try:
    student = ladok.get_student(args.student_id)
    course = student.courses(code=args.course_code)[0]
    result = course.results(component=args.component_code)[0]
    result.set_grade(args.grade, args.date)
    if args.finalize:
      result.finalize()
  except Exception as err:
    try:
      print(f"{student}: {err}")
    except ValueError as verr:
      print(f"{verr}: {args.student_id}: {err}")
```

This means that we need the following command-line arguments. The option `args.finalize` is already set above, so we don't need to add that one here.

18.4 Report many results given in standard input

We want to read CSV data from standard input.

164a ⟨*report results given in stdin* 164a⟩≡ (162b)
```
  data_reader = csv.reader(sys.stdin, delimiter=args.delimiter)
  for course_code, component_code, student_id, grade, date in data_reader:
    ⟨report a result read from stdin 164d⟩
```

We need to add an argument for the delimiter.

164b ⟨*add many results group arguments* 164b⟩≡ (162a) 164c ▷
```
  many_parser.add_argument("-d", "-delimiter",
    default="\t",
    help="The delimiter for the CSV input; "
      "default is a tab character to be compatible with POSIX commands, "
      "use ‘-d,‘ or ‘-d ,‘ to get comma-separated values.")
```

We also want to handle errors and confirmations differently. When reporting in bulk, we don't want unnecessary errors. We also want to have a summary of the changes.

164c ⟨*add many results group arguments* 164b⟩+≡ (162a) ◁ 164b
```
  many_parser.add_argument("-v", "-verbose",
    action="count", default=0,
    help="Increases the verbosity of the output: -v will print results that "
      "were reported to standard out. Otherwise only errors are printed.")
```

When we actually report a result, this is similar to how we did it above. The difference is that we've read the variables from the CSV data in `stdin` instead of from `args`.

164d ⟨*report a result read from stdin* 164d⟩≡ (164a)
```
  try:
    student = ladok.get_student(student_id)

    try:
      course = student.courses(code=course_code)[0]
    except IndexError:
      raise Exception(f"{course_code}: no such course.")

    try:
      component = course.results(component=component_code)[0]
    except IndexError:
      raise Exception(f"{course_code} has no component {component_code}.")

    if not component.attested and component.grade != grade:
      component.set_grade(grade, date)
      component.finalize()
      if args.verbose:
        print(f"{course_code} {student}: reported "
          f"{component.component} = {component.grade} ({date}).")
    elif component.grade != grade:
      print(f"{course_code} {student}: attested {component.component} "
        f"result {component.grade} ({component.date}) "
      f"is different from {grade} ({date}).")
  except Exception as err:
    try:
      print(f"{course_code} {component_code}={grade} ({date}) {student}: {err}",
        file=sys.stderr)
    except ValueError as verr:
      print(f"{verr}: "
        f"{course_code} {component_code}={grade} ({date}) {student_id}: {err}",
        file=sys.stderr)
```

18.5 Determine which function to run

We know that if a course code is given as positional argument, then we know that we should report one result from the command line.

165 ⟨*report results depending on args* 165⟩≡ (161)

```
  if args.course_code:
    report_one_result(ladok, args)
  else:
    report_many_results(ladok, args)
```

Chapter 19

The student command

We want to show data for a student.For this, we need a student identifier. We want to use it like this:

```
ladok student 700101-1234
```

We also want to optionally print courses.

```
ladok student 700101-1234 -c "DD131[0-9]"
```

19.1 The student subcommand

This is a subcommand is run as part of the `ladok3.cli` module. We provide a function `add_command_options` that adds the subcommand options to a given parser. We need access to LADOK through the `ladok3` module.

166 ⟨*student.py* 166⟩≡

```
  import csv
  import ladok3.cli

  ⟨functions 167d⟩

  def add_command_options(parser):
    student_parser = parser.add_parser("student",
      help="Shows a student's information in LADOK",
      description="""
      Show a student's information in LADOK.
      Shows information like name, personnummer, contact information.
      """
    )
    student_parser.set_defaults(func=command)
    ⟨add student command arguments to student parser 167a⟩

  def command(ladok, args):
    ⟨print info depending on args 167c⟩
```

19.2 The student command options

19.2.1 The student identifier

As mentioned above, we want to take one positional argument, the student identifier. Since our API (Section 6.1) can handle both personnummer and LADOK ID transparently, we don't have to care about that either.

167a ⟨*add student command arguments to student parser* 167a⟩≡ (166) 167b ▷

```
student_parser.add_argument("id",
  help="The student's ID, either personnummer or LADOK ID"
)
```

19.2.2 An optional course identifier

We also want an option for the course code.

167b ⟨*add student command arguments to student parser* 167a⟩+≡ (166) ◁ 167a

```
student_parser.add_argument("-c", "-course",
  nargs="?", const=".*",
  help="A regular expression for which course codes to list, " \
    "use no value for listing all courses."
)
```

19.3 Print the student data

Now that we have the student identifier, we can simply use that to fetch the student from LADOK. Note that we access the `student.alive` attribute just to force pulling the personal data from LADOK to trigger any error in the try-statement.

167c ⟨*print info depending on args* 167c⟩≡ (166)

```
try:
  student = ladok.get_student(args.id)
  student.alive
except Exception as err:
  ladok3.cli.err(-1, f"can't fetch student data for {args.id}: {err}")

print_student_data(student)

if args.course:
  print_course_data(student, args.course)
```

19.3.1 Printing personal student data

To print the student's personal data, we simply print the most interesting attributes.

167d ⟨*functions* 167d⟩≡ (166) 168 ▷

```
def print_student_data(student):
  """Prints the student data, all attributes, to stdout."""
```

```
    print(f"First name:   {student.first_name}")
    print(f"Last name:    {student.last_name}")
    print(f"Personnummer: {student.personnummer}")
    print(f"LADOK ID:     {student.ladok_id}")
    print(f"Alice:        {student.alive}")
```

19.3.2 Printing student's course data

To print the student's course data, we simply filter the courses on the option that the user supplies.

168 ⟨*functions* 167d⟩+≡ (166) ◁167d

```
  def print_course_data(student, course):
    """Prints the courses"""
    print("Courses:")
    for course in student.courses(courde_code=course):
      print(f"- {course}")
```

Part V

Other example applications

Chapter 20

Transfer results from KTH Canvas to LADOK

Here we provide an example program `canvas2ladok.py` which exports results from KTH Canvas to LADOK for the introductory programming course prgi (DD1315). You can find an up-to-date version of this chapter at

`https://github.com/dbosk/intropy/tree/master/adm/reporting`.

20.1 Overview

We want to get results out of Canvas and put them into LADOK. We want to take each assignment group in Canvas and map into a course component in LADOK. *Assumption*: we assume that all assignments in the Canvas course instance are grouped; and each group must map exactly to the names of the LADOK components.

The course components in LADOK (and assignment groups in Canvas) for the prgi (DD1315) course are

- LAB1,
- LAB2,
- LAB3,
- MAT1[1].

The LAB3 component corresponds to a project. In Canvas, it has P/F assignments (e.g., presentation) and one A–F assignment (the project itself). The component grade is based on the project itself, but the other assignments each must have a P too.

For now we will ignore setting the grade on the full course.

When running the program we must have the following environment variables set:

[1] URL: `https://www.kth.se/student/kurser/kurs/DD1315`

- `LADOK_LOGIN` and `LADOK_PASSWD` which hold the credentials for LADOK (or KTH CAS).
- `CANVAS_SERVER` and `CANVAS_TOKEN` to point to the Canvas server to use and a token to authenticate.

We provide a Docker image for running this program.

20.1.1 Building files

We provide the following files:

- ⟨*canvas2ladok.mk* 171a⟩.
- ⟨*Dockerfile* 171b⟩,
- ⟨*canvas2ladok.py* 172⟩,

We provide an include file for GNU make to build all the targets.

171a ⟨*canvas2ladok.mk* 171a⟩≡

```
.PHONY: all
all: canvas2ladok.py prgi2ladok

.PHONY: prgi2ladok
prgi2ladok: canvas2ladok.py Dockerfile
  docker build -t prgi2ladok .

Dockerfile: canvas2ladok.nw
  ${NOTANGLE}

.PHONY: clean
clean:
  ${RM} canvas2ladok.pdf canvas2ladok.py
  ${RM} Dockerfile canvas2ladok.mk

.PHONY: distclean
distclean:
  -docker image rm prgi2ladok
```

The Docker image will just contain the `canvas2ladok.py` file and run it by default.

171b ⟨*Dockerfile* 171b⟩≡

```
FROM dbosk/ladok3

RUN pip install -no-cache-dir canvasapi
COPY canvas2ladok.py ./

CMD ["python", "./canvas2ladok.py"]
```

20.2 `canvas2ladok.py`: configuration and dependencies

We use the standard structure for the script. We start with the imports and set up our `ladok` and `canvas` objects, these are used to access the two systems.

172 ⟨*canvas2ladok.py* 172⟩≡

```
import cachetools
from canvasapi import Canvas
import collections
import datetime
import ladok3
import os
import re

LOGIN = os.environ["LADOK_USER"]
PASSWD = os.environ["LADOK_PASS"]

ladok = ladok3.kth.LadokSession(
        LOGIN, PASSWD,
        test_environment=False) # for experiments

API_URL = os.environ["CANVAS_SERVER"]
API_KEY = os.environ["CANVAS_TOKEN"]

canvas = Canvas(API_URL, API_KEY)

course_code_ladok = "DD1315"
# We match all course round HT20 and onwards
course_code_canvas = "DD1315[HV]T[2-9][0-9]"
components = [
  "LAB1",
  "LAB2",
  "LAB3",
  "MAT1"
]

⟨classes 174a⟩
⟨functions 173a⟩

def main():
  ⟨main body 173d⟩

if __name__ == "__main__":
  main()
```

20.3 Getting the results out of Canvas

We must first select the course. We do this my matching against a regular expression.

173a ⟨*functions* 173a⟩≡ (172) 173b ▷

```
@cachetools.cached(cache={})
def get_courses(code):
  pattern = re.compile(code)

  for course in canvas.get_courses():
    if pattern.match(course.course_code):
      yield course

  return None
```

We must process each assignment group (which corresponds to a LADOK component). So we get all relevant objects from Canvas.

173b ⟨*functions* 173a⟩+≡ (172) ◁ 173a 173c ▷

```
def get_components(course):
  for component in course.get_assignment_groups():
    if component.name in components:
      yield component
```

For each component, we must check which students passed all assignments; we must report the student's LADOK identifier, what grade they should have and the time of submission of the assignment.

173c ⟨*functions* 173a⟩+≡ (172) ◁ 173b 173e ▷

```
def students_results(course, component):
  ⟨produce results for a component 176⟩
```

We cover the details in Section 20.6, once we've covered what LADOK needs in Section 20.5.

20.4 The main body

Then we can report those results to LADOK. In summary, we do the following:

173d ⟨*main body* 173d⟩≡ (172)

```
for course in get_courses(course_code_canvas):
  for component in get_components(course):
    print(f"{course.course_code} {component.name}")
    component_results = students_results(course, component)
    report_results(course_code_ladok, component.name, component_results)
```

20.5 Putting the results into LADOK

We get a list of results for a component. We simply try to report them to LADOK.

173e ⟨*functions* 173a⟩+≡ (172) ◁ 173c 174c ▷

```
def report_results(course_code, component_name, results):
  for result in results:
    try:
      student = ladok.get_student(result.person_nr)
      course = student.courses(code=course_code)[0]
      component = course.results(component=component_name)[0]
      if not component.attested:
        component.set_grade(result.grade, result.date)
        component.finalize()
      elif component.grade != result.grade:
        print(f"{student}: attested {component.component} "
          f"result {component.grade} is different from {result.grade}.")
    except Exception as err:
      try:
        print(f"{student}: {err}")
      except ValueError as verr:
        print(f"{verr}: {result.person_nr}: {err}")
```

20.6 Producing the list of results

We need the result to have the following data:

174a ⟨*classes* 174a⟩≡ (172) 174b▷

```
Result = collections.namedtuple(
  "Result", ["person_nr", "date", "grade", "grade_scale"])
```

Now, given a `component`, we should produce such `Result` tuples.

20.6.1 Produce gradebook

To do this, we must first produce a gradebook that we can filter. The gradebook will use the simpler `Grade` tuple. It will use the student's LADOK ID as the key.

174b ⟨*classes* 174a⟩+≡ (172) ◁174a

```
Grade = collections.namedtuple("Grade", ["grade", "date"])
```

To create the gradebook, we will first fetch all the users and then fetch all the submissions. Then we will map them together. This way we only need two requests to Canvas, instead of one per user.

174c ⟨*functions* 173a⟩+≡ (172) ◁173e 175a▷

```
def make_gradebook(course, component):
  gradebook = {}

  users_by_id = {}
  for user in course.get_users(enrollment_type=["student"]):
    users_by_id[user.id] = user
    gradebook[user.integration_id] = {}

  for assignment in assignments_in_component(course, component):
    for submission in assignment.get_submissions():
      try:
```

```
      user = users_by_id[submission.user_id].integration_id
    except KeyError:
      # This only happens for the Test Student
      continue

    if not submission.graded_at:
      gradebook[user][assignment.name] = None
      continue

    date = submission.submitted_at or submission.graded_at
    date = datetime.date.fromisoformat(date.split("T")[0]) or None
    graded_date = datetime.date.fromisoformat(
      submission.graded_at.split("T")[0])
    if date > graded_date:
      date = graded_date

    gradebook[user][assignment.name] = \
      Grade(grade=submission.grade, date=date)

return gradebook
```

Note that for some users, we won't get any submission. And for some other users, we get a submission with no grade and no date.

We the component object doesn't give the assignments it include. We must check each assignment if it belongs to the component or not.

175a ⟨*functions* 173a⟩+≡ (172) ◁174c 175b▷

```
def assignments_in_component(course, component):
  return filter(
    lambda assignment: assignment.assignment_group_id == component.id,
    course.get_assignments())
```

20.6.2 Assignment grades to component grade

Next, we must filter the gradebook so that only the students who passed all assignments are reported to LADOK with a pass. A student must pass all assignments, the latest date sets the date to use in LADOK. *Assumption:* If some assignments are P/F and some are A–F, then the A–F one should be the grade. There will not be more than one A–F assignment per component.

175b ⟨*functions* 173a⟩+≡ (172) ◁175a

```
def summarize(grades):
  max_grade = "P"
  grade_scale = "PF"
  max_date = datetime.date(year=1970, month=1, day=1)
  for _, grade in grades.items():
    if grade is None or grade.grade is None or grade.grade[0] == "F":
      return None

    if grade.date > max_date:
      max_date = grade.date
    if grade.grade in "ABCDE":
      max_grade = grade.grade
      grade_scale = "AF"

  return (Grade(grade=max_grade, date=max_date), grade_scale)
```

20.6.3 Convert to LADOK result form

Lastly, we must put these things together and convert the result to the `Result` format that we need.

176 ⟨*produce results for a component* 176⟩≡ (173c)

```
  to_report = []
  gradebook = make_gradebook(course, component)

  for ladok_id, grades in gradebook.items():
    try:
      grade, grade_scale = summarize(grades)
    except TypeError as err:
      # Only happens for students who are not done
      continue

    to_report.append(Result(
      person_nr=ladok.get_student(ladok_id).personnummer,
      grade=grade.grade,
      grade_scale=grade_scale,
      date=grade.date))

  return to_report
```