

# A LADOK3 Python API

Alexander Baltatzis, Daniel Bosk, Gerald Q. ‘Chip’ Maguire Jr.

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

25th October 2022

MIT License

Copyright (c) 2020 Alexander Baltatzis  
Copyright (c) 2020 Gerald Q. Maguire Jr.  
Copyright (c) 2021 Daniel Bosk

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## **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

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>I</b>	<b>The library</b>	<b>2</b>
<b>2</b>	<b>Overview</b>	<b>3</b>
<b>3</b>	<b>Accessing LADOK</b>	<b>4</b>
3.1	The <code>LadokSession</code> class . . . . .	4
3.1.1	Data about the session . . . . .	6
3.1.2	Signing in to LADOK . . . . .	7
3.1.3	Signing out of LADOK . . . . .	7
3.2	The <code>LadokSession</code> data methods . . . . .	8
3.2.1	LADOK data base class: <code>LadokData</code> . . . . .	8
3.2.2	To and from JSON format . . . . .	9
3.2.3	LADOK remote data base class: <code>LadokRemoteData</code> . . . .	10
3.2.4	Cached methods . . . . .	10
<b>4</b>	<b>Helper functions</b>	<b>11</b>
4.1	Filter on keys . . . . .	11
4.2	Extracting translations . . . . .	14
<b>5</b>	<b>Grading scales</b>	<b>15</b>
5.1	The <code>get_grade_scales</code> method . . . . .	15
5.2	The <code>GradeScale</code> and <code>Grade</code> classes . . . . .	15
<b>6</b>	<b>Students</b>	<b>18</b>
6.1	Getting students from LADOK . . . . .	19
6.2	The <code>Student</code> class . . . . .	20
6.3	Students' personal attributes . . . . .	21
6.4	Students' study-related attributes . . . . .	22
<b>7</b>	<b>Courses</b>	<b>24</b>
7.1	Course rounds . . . . .	24
7.2	Course instances . . . . .	27
7.3	Course components . . . . .	32
7.4	Results for a course round . . . . .	34
7.5	Participants for a course round . . . . .	34

<b>8</b>	<b>Course objects related to students</b>	<b>35</b>
8.1	Course registrations and a student's results . . . . .	35
8.2	Course results . . . . .	37
8.2.1	Setting the grade . . . . .	40
8.2.2	Working with existing results . . . . .	40
8.2.3	Adding new results . . . . .	40
8.2.4	Finalizing a result . . . . .	41
<b>II</b>	<b>Logging in at different universities</b>	<b>42</b>
<b>9</b>	<b>Logging in at KTH</b>	<b>43</b>
9.1	Logging in . . . . .	44
9.2	Running the request through Shibboleth . . . . .	46
9.3	Posting username and password . . . . .	46
9.4	Redirecting back to LADOK . . . . .	47
<b>III</b>	<b>API calls</b>	<b>49</b>
<b>10</b>	<b>Overview of helper functions and methods</b>	<b>50</b>
10.1	HTTP queries to LADOK . . . . .	50
10.2	Cleaning data for printing . . . . .	51
10.3	Example code and test code for the API . . . . .	52
<b>11</b>	<b>Grade-related API calls</b>	<b>53</b>
11.1	grade_scales_JSON . . . . .	53
<b>12</b>	<b>Student-related API calls</b>	<b>54</b>
12.1	get_student_data_JSON and get_student_data_by_uid_JSON .	54
12.2	registrations_JSON . . . . .	55
12.3	registrations_on_course_JSON . . . . .	56
<b>13</b>	<b>Course-related API calls</b>	<b>58</b>
13.1	course_rounds_JSON . . . . .	58
13.2	course_instance_JSON . . . . .	58
13.3	Course components . . . . .	59
13.3.1	course_round_components_JSON . . . . .	59
13.3.2	course_instance_components_JSON . . . . .	60
<b>14</b>	<b>Results-related API calls</b>	<b>61</b>
14.1	Reported results . . . . .	61
14.1.1	search_reported_results_JSON . . . . .	61
14.1.2	search_course_results_JSON . . . . .	62
14.2	Results for a student: student_results_JSON . . . . .	63
14.3	Modifying results . . . . .	64
14.3.1	create_result_JSON . . . . .	64
14.3.2	update_result_JSON . . . . .	64
14.4	Finalizing a result . . . . .	65
14.4.1	result_attestants_JSON and result_reporters_JSON .	65
14.4.2	user_info_JSON . . . . .	66

14.4.3	finalize_result_JSON	67
14.5	participants_JSON	67
<b>15</b>	<b>Baltatzis' and Maguire's original LadokSession methods</b>	<b>70</b>
15.1	get_results, save_result	70
15.2	get_student_data, get_student_name	74
15.3	all_grading_scale	75
15.4	grading_rights	75
15.5	change_locale	76
15.6	course_instances_JSON	76
15.7	organization_info_JSON	77
15.8	period_info_JSON	77
15.9	instance_info	77
15.10	instance_info_uid	78
15.11	studystucture_student_JSON	78
15.12	larosatesinformation_JSON	79
15.13	undervisningssprak_JSON	79
15.14	i18n_translation_JSON	82
15.15	svenskorter_JSON	83
15.16	kommuner_JSON	83
15.17	lander_JSON	84
15.18	undervisningstid_JSON	85
15.19	successivfordjupning_JSON	86
15.20	undervisningsform_JSON	87
15.21	LokalaPerioder_JSON	89
15.22	nivainomstudieordning_JSON	90
15.23	amnesgrupp_JSON	91
15.24	studietakt_JSON	92
15.25	finansieringsform_JSON	113
15.26	utbildningsomrade_JSON	113
15.27	kravpatidigarestudier_JSON	115
15.28	studieordning_JSON	116
15.29	enhet_JSON	122
15.30	studielokalisering_JSON	124
15.31	antagningsomgang_JSON	132
15.32	organisation_by_uid_JSON	133
15.33	utbildningstyp_JSON	134
15.34	aktivitetstillfallestyp_JSON	136
15.35	catalog_service_index__JSON	137
15.36	omradesbehorighet_JSON	138
15.37	hamtaStudieResultatForStudent_JSON	138
15.38	examen_student_uid_JSON	138
15.39	Helper methods	139

## IV A command-line interface 143

<b>16</b>	<b>The base interface</b>	<b>144</b>
16.1	Overview of the source code and dependencies	144
16.2	A command-line interface	145

16.2.1	Process command-line options . . . . .	145
16.3	Create the <code>LadokSession</code> <code>ls</code> . . . . .	146
16.3.1	Saving and restoring the <code>LadokSession</code> object . . . . .	146
16.4	Configuration . . . . .	148
16.5	Managing credentials . . . . .	148
16.5.1	Updating the credentials in the keyring . . . . .	149
16.5.2	Loading user credentials . . . . .	149
16.6	Managing the cache . . . . .	150
16.6.1	Clear the cache . . . . .	150
16.7	Other subcommands . . . . .	151
<b>17</b>	<b>The data command</b>	<b>152</b>
17.1	The <code>data</code> subcommand . . . . .	154
17.1.1	Setting up the data command options . . . . .	154
17.2	Producing data . . . . .	154
17.3	Extracting data for a round . . . . .	155
17.3.1	Get round data . . . . .	156
17.3.2	Filtering data . . . . .	158
17.4	Which students to exclude . . . . .	158
<b>18</b>	<b>The report command</b>	<b>160</b>
18.1	The <code>report</code> subcommand . . . . .	160
18.2	Report the results . . . . .	161
18.3	Report a result given on command line . . . . .	161
18.4	Report many results given in standard input . . . . .	163
18.5	Determine which function to run . . . . .	164
<b>V</b>	<b>Other example applications</b>	<b>165</b>
<b>19</b>	<b>Transfer results from KTH Canvas to LADOK</b>	<b>166</b>
19.1	Overview . . . . .	166
19.1.1	Building files . . . . .	167
19.2	<code>canvas2ladok.py</code> : configuration and dependencies . . . . .	168
19.3	Getting the results out of Canvas . . . . .	169
19.4	The main body . . . . .	169
19.5	Putting the results into LADOK . . . . .	169
19.6	Producing the list of results . . . . .	170
19.6.1	Produce gradebook . . . . .	170
19.6.2	Assignment grades to component grade . . . . .	171
19.6.3	Convert to LADOK result form . . . . .	172

# 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 seen 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

```
1 pip3 install ladok3
```

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

```
2 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 use 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.

```
1 import ladok3
2
3 ladok = ladok3.kth.LadokSession(
4     "dbosk", "my secret password",
5     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 15a⟩*

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`<sup>1</sup>

```
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' }
```

<sup>1</sup>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<sup>2</sup>.

```
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<sup>3</sup>.

```
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.")
```

<sup>2</sup>This neat trick is due to Alexander Baltatzis.

<sup>3</sup>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 15b>
      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.

```

1  @cachetools.cachedmethod(
2      operator.attrgetter("cache"),
3      key=functools.partial(cachetools.keys.hashkey, "method"))
4  def method(self, *args):
5      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
          attributes.
        - 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], first_name="Student")
        gives [student]
```

```

filter_on_keys([student], first_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     $\langle$ test functions 6a $\rangle + \equiv$     (3b)  $\langle$ 6a 13a $\rangle$

```

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     $\langle$ functions 11 $\rangle + \equiv$     (3a)  $\langle$ 11 13b $\rangle$

```

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
      attributes.
    - 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], first_name="Student")
    gives [student]
filter_on_keys([student], first_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 13c>
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.

```

13b  <functions 11>+≡ (3a) <12b 14>
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 type(val1) != str or type(val2) != str:
        return val1 == val2

    return re.search(val2, val1)

```

We test this function with the following tests.

```

13c  <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:

```

1  {
2    "Benamning": [
3      {
4        "Sprakkod": "sv",
5        "Text": "Laborationer",
6        "link": []
7      },
8      {
9        "Sprakkod": "en",
10       "Text": "Programming
11              Assignments",
12       "link": []
13     }
14   ]
15 }

```

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

```

14  <functions 11>+≡ (3a) <13b 20b>
    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.

```
15a  <LadokSession data methods 15a>≡ (5) 19▷
      @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:

This makes the follow class design suitable.

```
15b  <classes 5>+≡ (3a) <10a 16▷
      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 17a>

def __iter__(self):
    <iterate over grades in grading scale 17b>

```

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.

```

16  <classes 5>+= (3a) <15b 20a>
    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.

17a  $\langle$ test if grade is in grading scale 17a $\rangle \equiv$  (15b)  
`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.

17b  $\langle$ iterate over grades in grading scale 17b $\rangle \equiv$  (15b)  
`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.

```
1 import ladok3
2 import os
3
4 ladok = ladok3.kth.LadokSession(
5     os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
6     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.

```
8 me = ladok.get_student("8506097891")
9 me2 = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")
10
11 print(f"{me.personnummer} {me.last_name}, {me.first_name}")
12 print(f"{me2.personnummer} {me2.last_name}, {me2.first_name}")
13 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).

```
16 for course in me.courses():
17     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.

```
19 course = me.courses(code="DD2395")[0]
20
21 print(f"{course.code} results:")
22 for result in course.results():
23     s = f"{course.code}"
```

```

24     if result.component:
25         s += f" {result.component}"
26     s += f" {result.grade}"
27     if result.attested:
28         s += f" ({result.date})"
29     print(s)

```

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

```

32 student = ladok.get_student("1234561234")
33 prgi = student.courses(code="DD1315")[0]
34
35 print(f"{student.personnummer} {student.first_name} {student.last_name}")
36
37 for result in prgi.results():
38     print(f"{result.component} {result.grade} ({result.date})", end="")
39     if not result.attested:
40         print("*")
41     else:
42         print()
43
44 print("Changing grades")
45
46 try:
47     lab2 = prgi.results(component="LAB1")[0]
48     lab2.set_grade("P", "2021-02-18")
49     lab2.finalize()
50 except Exception as err:
51     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 can delegate this job to the `Student` class.

```

19 <LadokSession data methods 15a>+= (5) <15a 24>
    @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`.

```
20a  <classes 5>+≡ (3a) <16 26>
      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 22a>

              <student attribute methods 21>
```

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.

```
20b  <functions 11>+≡ (3a) <14
      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.

```

21  <student attribute methods 21>≡ (20a) 22c>
    def __get_personal_attributes(self):
        """Helper method that fetches personal attributes"""
        <pull student attributes from LADOK 22b>

    @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.

22a *⟨pull all attributes from LADOK 22a⟩*≡ (20a)  
`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.

22b *⟨pull student attributes from LADOK 22b⟩*≡ (21)  

```

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.

22c *⟨student attribute methods 21⟩*+≡ (20a) <21  

```

def __get_study_attributes(self):
    """Helper method to fetch study related attributes"""
    ⟨create a list of the student's course objects 23b⟩

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).

```

23a  <pull all student attributes 23a>≡
      self.__get_study_attributes()

23b  <create a list of the student's course objects 23b>≡ (22c)
      # detta är egentligen kurstillfällen, inte kurser (ID-numret är alltså ett
      # ID-nummer för ett kurstillfälle)
      response = self.ladok.session.get(
          url=self.ladok.base_gui_proxy_url+
              '/studiedeltagande/tillfallesdeltagande/kurstillfallesdeltagande/student/'+
              self.ladok_id,
          headers=self.ladok.headers).json()

      self.__courses = []

      for course in response['Tillfallesdeltaganden']:
          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.

```
1  #!/usr/bin/python3
2  # -*- coding: utf-8 -*-
3  # -*- mode: python; python-indent-offset: 4 -*-
4  import ladok3
5  import os
6
7  ladok = ladok3.kth.LadokSession(
8      os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
9      test_environment=True) # for experiments
10
11  prgiX = ladok.search_course_rounds(code="DD1315")
12
13  for prgi in prgiX:
14      print(f"{prgi.code} {prgi.start}--{prgi.end}")
15  print()
16
17  prgi = prgiX[0]
18  print(f"{prgi.code} {prgi.start}--{prgi.end}")
19  print(f"round: {prgi.round_id}")
20  print(f"round code: {prgi.round_code}")
21  print(f"instance: {prgi.instance_id}")
22  print(f"education: {prgi.education_id}")
23  print(f"{prgi.code} components:")
24  for component in prgi.components():
25      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.

```
24  <LadokSession data methods 15a>+≡ (5) <19
    @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:

```

1  {
2    "Resultat": [
3      "0": {
4        "Slutdatum": "2021-01-15",
5        "Startdatum": "2020-08-24",
6        "StudielokaliseringID": 135195,
7        "StudietaktID": 6,
8        "TillfallesKod": "51386",
9        "Uid": "4e94fe55-1cef-11ea-a622-3565135944de",
10       "UndervisningsformID": 1,
11       "Utbildningsinstans": {
12         "Avvecklad": false,
13         "Benamning": [
14           {
15             "Sprakkod": "sv",
16             "Text": "Programmeringsteknik och Matlab",
17             "link": []
18           },
19           {
20             "Sprakkod": "en",
21             "Text": "Programming Techniques and Matlab",
22             "link": []
23           }
24         ],
25         "Enhhet": "HP",

```



```

26         "GallerUtbildningUtanAngivenOmfattning": false,
27         "Omfattning": 7.5,
28         "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
29         "ResultatPaKursVidAttesteringsAvModul": false,
30         "Uid": "9f30cc02-d6b5-11e8-8fd5-cf9d2c5c41ba",
31         "UtbildningUID": "38ab2393-73d8-11e8-afa7-8e408e694e54",
32         "Utbildningskod": "DD1315",
33         "Versionsnummer": 2,
34         "link": []
35     },
36     "link": [ ... ]
37 },
38 "1": { ... },
39 ...
40 ],
41 TotaltAntalPoster: 32
42 }

```

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).

```

26 <classes 5>+= (3a) <20a 32b>
    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 27b>

    class CourseRound(CourseInstance):
        """Represents a course round"""
        def __init__(self, /, **kwargs):
            """Must be constructed from at least:
            Uid, TillfallKod, Startdatum, Slutdatum"""
            <prepare kwargs and call CourseInstance constructor 27a>

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

            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 34a⟩*

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.

27a *⟨prepare kwargs and call CourseInstance constructor 27a⟩*≡ (26)

```

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.

27b *⟨CourseInstance methods 27b⟩*≡ (26)

```

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 32a⟩

def __pull_attributes(self):
    ⟨fetch CourseInstance data object from LADOK 28⟩
    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.

```

28  <fetch CourseInstance data object from LADOK 28>≡ (27b)
    headers = self.ladok.headers.copy()
    headers["Content-Type"] = "application/vnd.ladok-resultat+json"
    headers["X-XSRF-TOKEN"] = self.ladok.get_xsrftoken()
    headers["Referer"] = self.ladok.base_gui_url

    put_data = {"Identitet": [self.__instance_id]}

    response = self.ladok.session.put(
        url=self.ladok.base_gui_proxy_url + '/resultat/utbildningsinstans/moduler',
        json=put_data,
        headers=headers)
    response = response.json()

```

```
data = response["Utbildningsinstans"][0]
```

Then data will be populated with the following values:

```

1  {
2    "Avvecklad": false,
3    "Benamning": [
4      {
5        "Sprakkod": "sv",
6        "Text": "Programmeringsteknik och Matlab",
7        "link": []
8      },
9      {
10       "Sprakkod": "en",
11       "Text": "Programming Techniques and Matlab",
12       "link": []
13     }
14   ],
15   "BetygsskalaID": 131657,
16   "Enhet": "HP",
17   "GallerUtbildningUtanAngivenOmfattning": false,
18   "KravPaHanvisningTillBeslutshandling": false,
19   "KravPaProjekttitel": false,
20   "Moduler": [
21     {
22       "Avvecklad": false,
23       "Benamning": [
24         {
25           "Sprakkod": "sv",
26           "Text": "Laborationer",
27           "link": []
28         },
29         {
30           "Sprakkod": "en",
31           "Text": "Programming Assignments",
32           "link": []
33         }
34       ],
35       "BetygsskalaID": 131656,
36       "Enhet": "HP",
37       "GallerUtbildningUtanAngivenOmfattning": false,
38       "KravPaHanvisningTillBeslutshandling": false,
39       "KravPaProjekttitel": false,
40       "Moduler": [],
41       "Omfattning": 1.5,
42       "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
43       "ResultatPaKursVidAttesteringAvModul": false,
44       "Uid": "387c7248-73d8-11e8-b4e0-063f9afb40e3",
45       "UtbildningUID": "387c99a9-73d8-11e8-afa7-8e408e694e54",
46       "Utbildningskod": "MAT1",

```

```

47     "Versionsnummer": 1,
48     "link": [ ... ]
49 },
50 {
51     "Avvecklad": false,
52     "Benamning": [
53         {
54             "Sprakkod": "sv",
55             "Text": "Laborationer",
56             "link": []
57         },
58         {
59             "Sprakkod": "en",
60             "Text": "Programming Assignments",
61             "link": []
62         }
63     ],
64     "BetygsskalaID": 131657,
65     "Enhet": "HP",
66     "GallerUtbildningUtanAngivenOmfattning": false,
67     "KravPaHanvisningTillBeslutshandling": false,
68     "KravPaProjekttitel": false,
69     "Moduler": [],
70     "Omfattning": 3,
71     "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
72     "ResultatPaKursVidAttesteringAvModul": false,
73     "Uid": "389b1dda-73d8-11e8-b4e0-063f9afb40e3",
74     "UtbildningUID": "389b1e62-73d8-11e8-afa7-8e408e694e54",
75     "Utbildningskod": "LAB3",
76     "Versionsnummer": 1,
77     "link": [ ... ]
78 },
79 {
80     "Avvecklad": false,
81     "Benamning": [
82         {
83             "Sprakkod": "sv",
84             "Text": "Laborationer",
85             "link": []
86         },
87         {
88             "Sprakkod": "en",
89             "Text": "Programming Assignments",
90             "link": []
91         }
92     ],
93     "BetygsskalaID": 131656,
94     "Enhet": "HP",
95     "GallerUtbildningUtanAngivenOmfattning": false,
96     "KravPaHanvisningTillBeslutshandling": false,

```

```

97     "KravPaProjekttitel": false,
98     "Moduler": [],
99     "Omfattning": 1.5,
100    "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
101    "ResultatPaKursVidAttesteringAvModul": false,
102    "Uid": "38a1867d-73d8-11e8-b4e0-063f9afb40e3",
103    "UtbildningUID": "38a1d44d-73d8-11e8-afa7-8e408e694e54",
104    "Utbildningskod": "LAB2",
105    "Versionsnummer": 1,
106    "link": [ ... ]
107  },
108  {
109    "Avvecklad": false,
110    "Benamning": [
111      {
112        "Sprakkod": "sv",
113        "Text": "Laborationer",
114        "link": []
115      },
116      {
117        "Sprakkod": "en",
118        "Text": "Programming Assignments",
119        "link": []
120      }
121    ],
122    "BetygsskalaID": 131656,
123    "Enhet": "HP",
124    "GallerUtbildningUtanAngivenOmfattning": false,
125    "KravPaHanvisningTillBeslutshandling": false,
126    "KravPaProjekttitel": false,
127    "Moduler": [],
128    "Omfattning": 1.5,
129    "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
130    "ResultatPaKursVidAttesteringAvModul": false,
131    "Uid": "38a6416f-73d8-11e8-b4e0-063f9afb40e3",
132    "UtbildningUID": "38a66848-73d8-11e8-afa7-8e408e694e54",
133    "Utbildningskod": "LAB1",
134    "Versionsnummer": 1,
135    "link": [ ... ]
136  }
137 ],
138 "Omfattning": 7.5,
139 "OrganisationUID": "2474f616-dc41-11e8-8cc1-eaeeb71b497f",
140 "ResultatPaKursVidAttesteringAvModul": false,
141 "Uid": "9f30cc02-d6b5-11e8-8fd5-cf9d2c5c41ba",
142 "UtbildningUID": "38ab2393-73d8-11e8-afa7-8e408e694e54",
143 "Utbildningskod": "DD1315",
144 "Versionsnummer": 2,
145 "link": [ ... ]
146 }

```

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

32a  $\langle \text{assign CourseInstance data to private attributes 32a} \rangle \equiv$  (27b)

```

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.

32b  $\langle \text{classes 5} \rangle + \equiv$  (3a)  $\langle 26 \ 35 \rangle$

```

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.

```
34a  <CourseRound data methods 34a>≡ (26) 34b>
      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.

```
34b  <CourseRound data methods 34a>+≡ (26) <34a 34c>
      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.

```
34c  <CourseRound data methods 34a>+≡ (26) <34b 34d>
      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.

```
34d  <CourseRound data methods 34a>+≡ (26) <34c
      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.

```
35 <classes 5>+≡ (3a) <32b 37b>
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 36>
    <add new CourseResult objects for missing components 37a>

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.

```

36  <pull existing CourseResult objects from LADOK 36> ≡ (35)
    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.

```
37a  <add new CourseResult objects for missing components 37a>≡ (35)
    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.

```
37b  <classes 5>+≡ (3a) <35
    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 39b⟩
    else:
        ⟨populate CourseResult attributes from data 39a⟩

@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 40a⟩

    self.__modified = True
    self.push()

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

    ⟨finalize the grade for CourseResult 41b⟩

@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 40b)
    else:
        (push the new CourseResult grade data to LADOK 41a)
        self.__modified = False

```

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

```

39a  (populate CourseResult attributes from data 39a)≡                                     (37b)
    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.

```

39b  (populate CourseResult attributes for empty result 39b)≡                         (37b)
    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.

```

40a  <set the grade for CourseResult 40a>≡ (37b)
    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.

```

40b  <push the existing CourseResult grade data to LADOK 40b>≡ (37b)
    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.

41a  $\langle \text{push the new CourseResult grade data to LADOK 41a} \rangle \equiv$  (37b)

```

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.

41b  $\langle \text{finalize the grade for CourseResult 41b} \rangle \equiv$  (37b)

```

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.

```
43a  <kth.py 43a>≡
      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 46a>

          def saml_login(self, url):
              """Do the SSO login"""
              <KTH login procedure 44>
```

We also test this login functionality.

```
43b  <test kth.py 43b>≡
      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

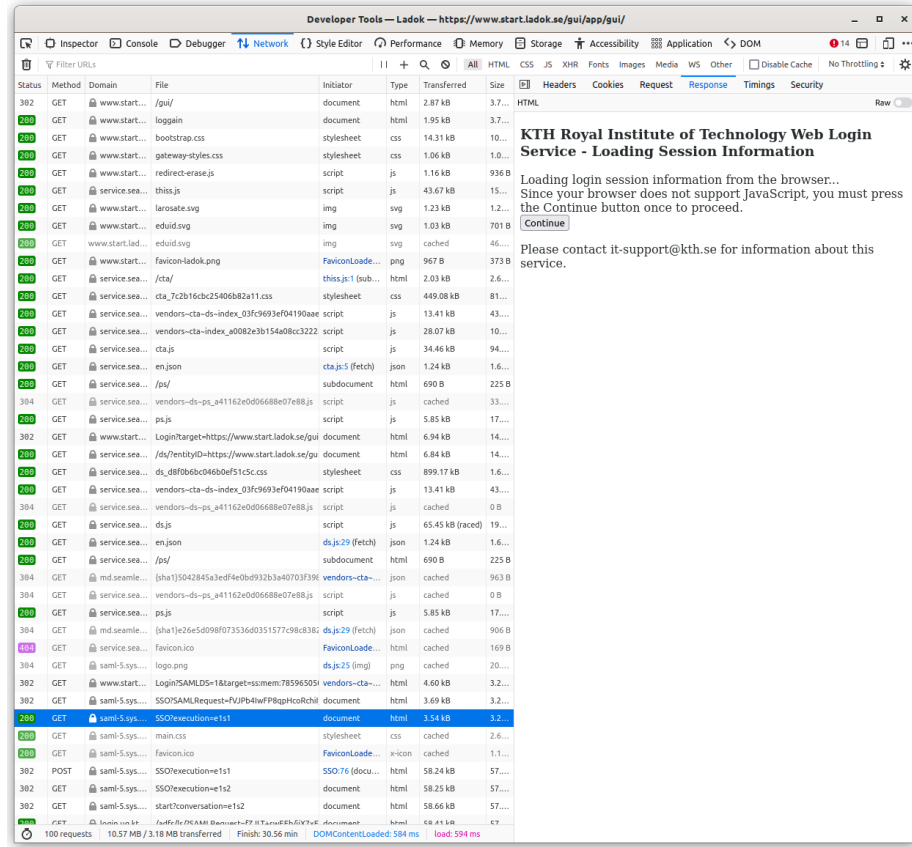


Figure 9.1: Firefox Developer Tools showing the requests when login in to LADOK through KTH.

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.

44  $\langle KTH \text{ login procedure } 44 \rangle \equiv$  (43a)

```

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

```

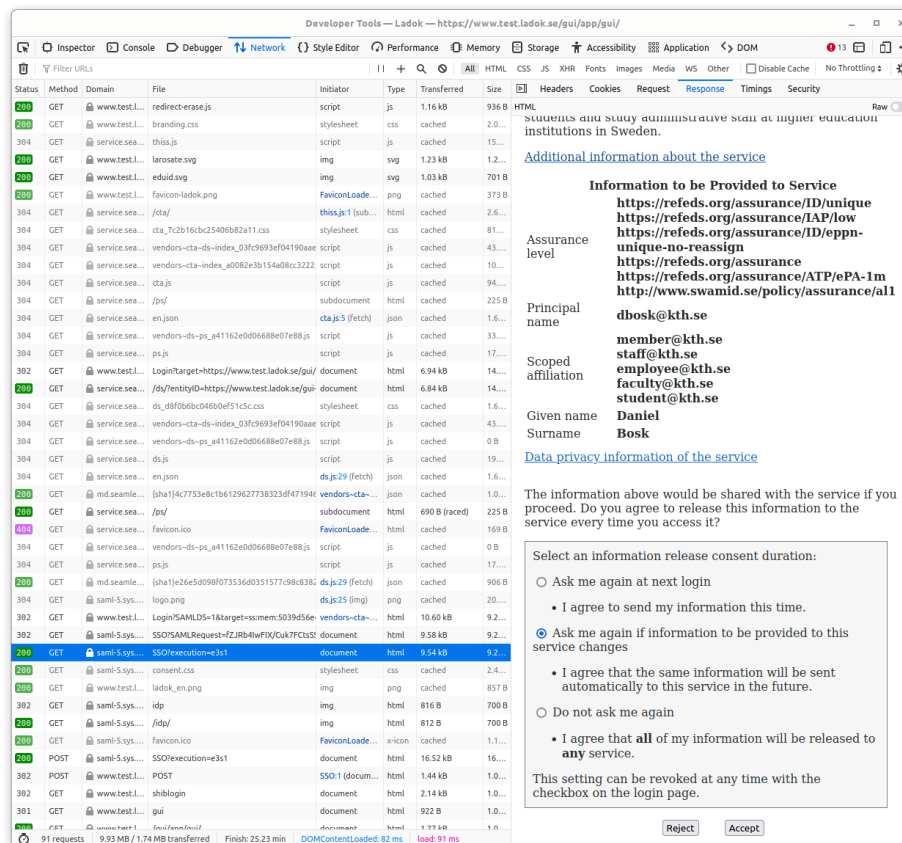


Figure 9.2: Firefox Developer Tools showing the requests when login in to LADOK through KTH for the first time. Pay attention to the page rendered on the right side, different from Fig. 9.1.

## 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.

```

46a  <KTH login methods 46a>≡ (43a) 46b>
    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.

```

46b  <KTH login methods 46a>+≡ (43a) <46a 47>
    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.

```

47  <KTH login methods 46a>+≡ (43a) <46b
    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.

```
50  <LadokSession data methods 50>≡ 53a>
    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.

```

51a  <functions 51a>≡ 51b>
      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.

```

51b  <functions 51a>+≡ <51a 51c>
      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.

```

51c  <functions 51a>+≡ <51b
      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.

```

1  import json
2  import ladok3
3  import os
4
5  ladok = ladok3.kth.LadokSession(
6      os.environ["KTH_LOGIN"], os.environ["KTH_PASSWD"],
7      test_environment=True) # for experiments

```

We will use the following to test the API methods.

```

52  <test api.py 52>≡
    import json
    import ladok3.kth
    import os

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

    <test functions 53b>

```

We will then add `test_X()` functions in the `<test functions 53b>` 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.

```
53a <LadokSession data methods 50>+≡ <50 54>
    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.)

```
53b <test functions 53b>≡ (52) 55a>
    def test_grade_scales_JSON():
        assert json.dumps(ladok.grade_scales_JSON()[0], indent=2, ensure_ascii=False)
```

This method is used as follows.

```
s 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

### 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.)

```
54  <LadokSession data methods 50>+≡ <53a 55b>
#####
#
# 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=FORNAMN_ASC'
        + 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.

```

55a  <test functions 53b>+≡ (52) <53b 55c>
    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.

```

55b  <LadokSession data methods 50>+≡ <54 55d>
    #####
    #
    # 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.

```

55c  <test functions 53b>+≡ (52) <55a 56a>
    def test_get_student_data_by_uid_JSON():
        assert ladok.get_student_data_by_uid_JSON("de709f81-a867-11e7-8dbf-78e86dc2470c")

```

## 12.2 registrations\_JSON

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

```

55d  <LadokSession data methods 50>+≡ <55b 56b>
    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.

```
56a <test functions 53b>+≡ (52) <55c 56c>
def test_registrations_JSON():
    me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")
    results = ladok.registrations_JSON(me.ladok_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
9 me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")
10
11 results = ladok.registrations_JSON(me.ladok_id)
12
13 ladok3.clean_data(results)
14 print(json.dumps(results, indent=2, ensure_ascii=False))
```

The output looks like this. ?? PythonTeX ??

### 12.3 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.

```
56b <LadokSession data methods 50>+≡ <55d 58a>
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.

```
56c <test functions 53b>+≡ (52) <56a 58b>
def test_registrations_on_course_JSON():
    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)

    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
15 me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")
16 dasak = me.courses(code="DD2395")[0]
17
18 results = ladok.registrations_on_course_JSON(dasak.education_id,
19     me.ladok_id)
20
21 ladok3.clean_data(results)
22 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.

```
58a <LadokSession data methods 50>+≡ <56b 58c>
    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.

```
58b <test functions 53b>+≡ (52) <56c 59a>
    def test_course_rounds_JSON():
        dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
        results = ladok.course_rounds_JSON(dasak10.instance_id)
        assert json.dumps(results[:1]+results[-1:], indent=2, ensure_ascii=False)
```

This method is used as follows.

```
23 dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
24 results = ladok.course_rounds_JSON(dasak10.instance_id)
25
26 ladok3.clean_data(results)
27 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.)

```
58c <LadokSession data methods 50>+≡ <58a 59b>
```

```

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.

```

59a  <test functions 53b>+≡ (52) <58b 59c>
def test_course_instance_JSON():
    dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
    results = ladok.course_instance_JSON(dasak10.instance_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)

```

This method is used as follows.

```

28 results = ladok.course_instance_JSON(dasak10.instance_id)
29
30 ladok3.clean_data(results)
31 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.

```

59b  <LadokSession data methods 50>+≡ <58c 60a>
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.

```

59c  <test functions 53b>+≡ (52) <59a 60b>
def test_course_round_components_JSON():
    dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
    components = ladok.course_round_components_JSON(dasak10.round_id)
    assert json.dumps(components, indent=2, ensure_ascii=False)

```

This method is used as follows.

```

32 try:
33     components = ladok.course_round_components_JSON(dasak10.round_id)
34 except Exception as err:
35     print(f"error: {err}")
36 else:
37     ladok3.clean_data(components)
38     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.

```

60a <LadokSession data methods 50>+≡ <59b 61>
    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.

```

60b <test functions 53b>+≡ (52) <59c 62a>
    def test_course_instance_components_JSON():
        dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
        components = ladok.course_instance_components_JSON(dasak10.instance_id)
        assert json.dumps(components, indent=2, ensure_ascii=False)

```

This method is used as follows.

```

39 try:
40     components = ladok.course_instance_components_JSON(dasak10.instance_id)
41 except Exception as err:
42     print(f"error: {err}")
43 else:
44     ladok3.clean_data(components)
45     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.

```
61  <LadokSession data methods 50>+≡ <60a 62b>
    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.

```

62a  <test functions 53b>+≡ (52) <60b 63a>
    def test_search_reported_results_JSON():
        dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
        LAB1 = dasak10.components(code="LAB1")[0]

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

        assert json.dumps(results, indent=2, ensure_ascii=False)

```

This method is used as follows.

```

46  LAB1 = dasak10.components(code="LAB1")[0]
47
48  results = ladok.search_reported_results_JSON(dasak10.round_id, LAB1.instance_id)
49
50  ladok3.clean_data(results)
51  results = list(filter(
52      lambda x: x["Student"]["Uid"] == me.ladok_id,
53      results))
54  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.

```

62b  <LadokSession data methods 50>+≡ <61 63b>
    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.

```
63a <test functions 53b>+≡ (52) <62a 63c>
def test_search_course_results_JSON():
    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)

    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```
55 dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
56 LAB1 = dasak10.components(code="LAB1")[0]
57
58 results = ladok.search_course_results_JSON(dasak10.round_id, LAB1.instance_id)
59
60 ladok3.clean_data(results)
61 results = list(filter(
62     lambda x: x["Student"]["Uid"] == me.ladok_id,
63     results))
64 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.

```
63b <LadokSession data methods 50>+≡ <62b 64a>
def student_results_JSON(self, student_id, course_round_id):
    """Returns the results for a student on a course round"""
    response = self.get_query(
        '/resultat/studierresultat/student/' + student_id +
        '/utbildningstillfalle/' + course_round_id
    )

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

We test this in the following way.

```
63c <test functions 53b>+≡ (52) <63a 65b>
def test_student_results_JSON():
    dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
    me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")

    results = ladok.student_results_JSON(me.ladok_id, dasak10.round_id)
    assert json.dumps(results, indent=2, ensure_ascii=False)
```

This method is used as follows.

```

65 results = ladok.student_results_JSON(me.ladok_id, dasak10.round_id)
66
67 ladok3.clean_data(results)
68 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.

```

64a <LadokSession data methods 50>+≡ <63b 64b>
    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/skapa",
            {"Resultat": [{
                "Betygsgrad": grade_id,
                "BetygsskalaID": grade_scale_id,
                "Examinationsdatum": date,
                "Noteringar": notes,
                "StudieresultatUID": study_result_id,
                "UtbildningsinstansUID": instance_id
            }]}
        )

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

```

### 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`.

```

64b <LadokSession data methods 50>+≡ <64a 65a>
    def update_result_JSON(self,

```

```

        grade_id, grade_scale_id, date,
        result_id, last_modified, notes=[]):
    response = self.put_query(
        '/resultat/studieresultat/updatera',
        {
            '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.

```

65a  <LadokSession data methods 50>+≡                                     <64b 66a>
    def result_attestants_JSON(self, result_id):
        """Returns a list of result attestants"""
        response = self.put_query(
            "/resultat/anvandare/resultatrattinghet/attestanter/kurstillfallesrapportering",
            {"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.

We can test this in the following way.

```

65b  <test functions 53b>+≡                                             (52) <63c 66b>
    def test_result_attestants_JSON():
        attestants = ladok.result_attestants_JSON(
            "d05c1e97-4c1e-11eb-8e41-bc743cd4482b")
        assert json.dumps(attestants[0], indent=2, ensure_ascii=False)

```



This method is used as follows.

```

69 attestants = ladok.result_attestants_JSON(
70     "d05c1e97-4c1e-11eb-8e41-bc743cd4482b")
71
72 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).

```

66a <LadokSession data methods 50>+≡ <65a 66c>
    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.

```

66b <test functions 53b>+≡ (52) <65b 67a>
    def result_reporters_JSON():
        components = ladok.course_round_components_JSON(dasak10.round_id)
        reporters = ladok.result_reporters_JSON(components["OrganisationUID"])
        assert json.dumps(reporters[8], indent=2, ensure_ascii=False)

```

This method is used as follows.

```

73 reporters = ladok.result_reporters_JSON(components["OrganisationUID"])
74 ladok3.remove_links(reporters)
75
76 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.

```

66c <LadokSession data methods 50>+≡ <66a 67b>
    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.

```
67a  <test functions 53b>+≡ (52) <66b 68>
      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.

77  me_teacher = ladok.user_info_JSON()
78  ladok3.remove_links(me_teacher)
79
80  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.

```
67b  <LadokSession data methods 50>+≡ <66c 67c>
      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/{result_id}/klarmarkera",
              {
                  "Beslutsfattare": [attestant_id] if attestant_id else [],
                  "RattadAv": [reporter_id],
                  "ResultatetsSenastSparad": last_modified
              }
          )

          if response.status_code == 200:
              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.)

```
67c  <LadokSession data methods 50>+≡ <67b>
      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'],
        'deltagaretilstand': 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.

```

68  <test functions 53b>+≡ (52) <67a
    def test_participants_JSON():
        dasak10 = ladok.search_course_rounds(code="DD2395", round_code="81099")[0]
        me = ladok.get_student("de709f81-a867-11e7-8dbf-78e86dc2470c")

        results = ladok.participants_JSON(dasak10.round_id)
        assert json.dumps(results, indent=2, ensure_ascii=False)

```

This method is used as follows.

```

s1  results = ladok.participants_JSON(dasak10.round_id)
s2
s3  ladok3.clean_data(results)
s4  results = list(filter(

```

```
85     lambda x: x["Student"]["Uid"] == me.ladok_id,  
86     results))  
87 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`

```
70  <LadokSession data methods 70>≡
#####
#
# 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 # tillgödöräkningen 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 tydliggen ...
    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 labmomentet är samma som course_code så sätts kursbetyg!
# result_date        - betygdatum, 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

```

74  <LadokSession data methods 70>+≡<70 75a>
    #####
    #
    # 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

```

75a  <LadokSession data methods 70>+≡                                     <74 75b>
      # 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

```

75b  <LadokSession data methods 70>+≡                                     <75a 76a>
      # 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

```
76a  <LadokSession data methods 70>+≡ <75b 76b>
    # 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

```
76b  <LadokSession data methods 70>+≡ <76a 77a>
    # 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

```

77a  <LadokSession data methods 70>+≡                                     <76b 77b>
      # 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

```

77b  <LadokSession data methods 70>+≡                                     <77a 77c>
      # 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

```

77c  <LadokSession data methods 70>+≡                                     <77b 78a>
      # 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

78a    *<LadokSession data methods 70>+≡* <77c 78b>

```

# 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

78b    *<LadokSession data methods 70>+≡* <78a 79a>

```

# 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).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

82 *<LadokSession data methods 70>+≡*

*<79b 83a>*

```

# 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': 'Kurs',
#         'Giltighetsperiod': { 'link': [
#         'Grundtyp': 'KURS',
#         'ID': 52,
#         'Kod': '2007KTF',
#         'RegelverkForUtbildningstyp': { 'en': 'Regelverk för utbildningstyp', 'sv': 'Regelverk för utbildningstyp' }
#     }
# }

```

```
# 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

83a <82 83b>

```
(LadokSession data methods 70)+≡
# 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

83b <83a 84>

```
(LadokSession data methods 70)+≡
# 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

```

84  <LadokSession data methods 70>+≡                                     <83b 85>
    # 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': 'B0',
    #               'LarosateID': -1,

```

```

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

```

## 15.18 undervisningstid\_JSON

85 *<LadokSession data methods 70>+≡*

*<84 86>*

```

# 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

```

86  <LadokSession data methods 70>+≡
# 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

```

87  <LadokSession data methods 70>+≡                                     <86 89>
# 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:

```



```

#                                     'link': []},
#                                     {  'Benamning': {  'en': 'Normal teaching',
#                                                         'sv': 'Normal'},
#                                     'Beskrivning': {},
#                                     'Giltighetsperiod': {'link': []},
#                                     'ID': '1',
#                                     'Kod': 'NML',
#                                     'LarosateID': -1,
#                                     'link': []}],
#   'link': []}

```

## 15.21 LokalaPerioder\_JSON

```

89  <LadokSession data methods 70>+≡                                     <87 90>
# 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': []},
#

```





```

#                                     '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

```

91  <LadokSession data methods 70>+≡                                     <90 92>
# 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

```

92  (LadokSession data methods 70)+≡                                     <91 113a>
    # 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': 'Två tredjedelsfart'},
#         'Beskrivning': { 'sv': 'Ändras till '
#                           '"Två tredjedelsfart" 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,

```





```
#####
#
# 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': []},
```



```

#         'ID': '1',
#         'Kod': 'UH',
#         'LarosateID': -1,
#         'link': [],
#         { 'Benamning': { 'en': 'Upper secondary '
#                           'or equivalent',
#                           'sv': 'Inga tidigare '
#                                 'högskolestudier '
#                                 'krävs'},
#         'Beskrivning': {},
#         'Emilvarde': 'grundläggande',
#         '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

```

116  <LadokSession data methods 70>+≡                                     <115 122>
# 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 '

```



```

#                                     '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 '
#                       'förrordningen '
#                       '(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 '
#                           'förrordningen (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': []}

```





```

#                                     'samband med övergången till '
#                                     '2007 års studieordning'},
#                                     '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 '
#                   'Försvärshögskolan'},
#   '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

124 *<LadokSession data methods 70>+≡*

*<122 132>*

```

# 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, '
#                           '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

```

132  <LadokSession data methods 70>+≡                                     <124 133>
# 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

133 *<LadokSession data methods 70>+≡*

*<132 134>*

```

# 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

```







```

#           {  '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

137 *<LadokSession data methods 70>+≡*

*<136 138a>*

```

# 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

```

138a  <LadokSession data methods 70>+≡                                     <137 138b>
      # added by GQMJr
      #####
      #
      # omradesbehorighet_JSON
      #
      # RETURNERAR JSON of "omradesbehorighet"
      # for information see https://antagning.se/globalassets/omradesbehorigheter-hogskolan.pdf
      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

```

138b  <LadokSession data methods 70>+≡                                     <138a 138c>
      # 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

```

138c  <LadokSession data methods 70>+≡                                     <138b 139>
      # 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

```

139  <LadokSession data methods 70>+≡                                     <138c
#####
##
## private methods
##

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

def get_xsrftoken(self):
    return self.__get_xsrftoken()

# 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=FORNAMN_ASC'
        + 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']['Betygsskal'])
} 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 144⟩](#).

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.

```
144 ⟨cli.py 144⟩≡
    #!/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 151b⟩
```

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

*⟨functions 145b⟩*

*⟨command-line interface 145a⟩*

## 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.

145a *⟨command-line interface 145a⟩*≡ (144)

```
def main():
    """Run the command-line interface for the ladok command"""
    ⟨process command-line options 145c⟩

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.

145b *⟨functions 145b⟩*≡ (144) 147a▷

```
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.

145c *⟨process command-line options 145c⟩*≡ (145a)

```
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 148b⟩
subp = argp.add_subparsers(
    title="commands",
    dest="command",
```



```

        required=True
    )
    <add subparsers to subp 148c>
    argcomplete.autocomplete(argp)
    args = argp.parse_args()
    <create or restore the LadokSession ls 146b>
    <execute subcommand 146a>
    <save LadokSession ls 146c>

```

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`.

```

146a <execute subcommand 146a>≡ (145c)
    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 than individual LADOK requests.

This leads to the following approach.

```

146b <create or restore the LadokSession ls 146b>≡ (145c)
    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)

146c <save LadokSession ls 146c>≡ (145c)
    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.

```

147a  <functions 145b>+≡ (144) <145b 149a>
      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 147b>

          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 147c>
                  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.

```

147b  <encrypt pickled ls 147b>≡ (147a)
      <set up kdf and derive key from credentials 148a>

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

147c  <decrypt encrypted ls 147c>≡ (147a)
      <set up kdf and derive key from credentials 148a>

      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.

148a  $\langle$ set up kdf and derive key from credentials 148a $\rangle \equiv$  (147)

```

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.

148b  $\langle$ add global configuration options 148b $\rangle \equiv$  (145c)

```

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.

148c  $\langle$ add subparsers to subp 148c $\rangle \equiv$  (145c) 150a $\triangleright$

```

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.

```

149a  <functions 145b>+≡ (144) <147a 149b>
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.)

```

149b  <functions 145b>+≡ (144) <149a 151a>
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.

```

150a  <add subparsers to subp 148c>+≡ (145c) <148c 151c>
      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 150b>

```

### 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.

```

150b  <add subcommands to cache command 150b>≡ (150a)
      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.

```
151a  <functions 145b>+≡ (144) <149b
      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 148c>*. 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.

```
151b  <modules 151b>≡ (144) 151d>
      import ladok3.data
```

```
151c  <add subparsers to subp 148c>+≡ (145c) <150a 151e>
      ladok3.data.add_command_options(subp)
```

We add the `report` command.

```
151d  <modules 151b>+≡ (144) <151b
      import ladok3.report
```

```
151e  <add subparsers to subp 148c>+≡ (145c) <151c
      ladok3.report.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 154a](#)) 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:

```
1 ladok data DD1315 > DD1315.csv
```

This program will produce CSV-formatted data to answer the questions above. The data is formatted like this:

- course,
- round,
- component,
- student (pseudonym),
- grade,
- normalized time.

Or more concretely, we can use it like this:

```
1 import numpy as np
2 import pandas as pd
3
4 prgi = pd.read_csv("DD1315.csv")
```

```

5
6 rounds_years = {
7     51386: "2020",
8     50869: "2019",
9     50662: "2018",
10    50650: "2017",
11    50523: "2016"
12 }
13 prgi_LAB3 = prgi[
14     prgi.Component.eq("LAB3") & prgi.Round.isin(rounds_years.keys())
15 ].replace(rounds_years)
16
17 stats = pd.crosstab(prgi_LAB3.Grade, prgi_LAB3.Round)
18
19 for column in stats:
20     stats[column] /= stats[column].sum()
21
22 print(stats.round(3).to_latex(
23     label="GradeDistribution",
24     caption="Grade distribution for the DD1315 course."
25 ))

```

This yields ??.

**?? PythonTeX ??**

If we convert the grades to numbers, then we can also compute the average grade with standard deviation etc.

```

26 grade_map = {
27     "A": 5, "B": 4, "C": 3, "D": 2, "E": 1, "-": 0
28 }
29 grades_num = prgi_LAB3[ ["Round", "Grade"] ].replace(grade_map)
30
31 print(grades_num.groupby("Round")["Grade"].describe().round(3).to_latex(
32     label="GradeStats",
33     caption="Statistics for grades of the DD1315 course, grade map: " +
34     ", ".join([f"{g} $\mapsto$ {s}" for g, s in grade_map.items()]) + ". "
35 ))

```

This yields ??.

**?? PythonTeX ??**

Finally, we'd like to merge the old rounds to compare with the latest.

```

36 rounds_avg = {
37     "2019": "2019--2016",
38     "2018": "2019--2016",
39     "2017": "2019--2016",
40     "2016": "2019--2016"
41 }
42 grades_avg = grades_num.replace(rounds_avg)
43
44 print(grades_avg.groupby("Round")["Grade"].describe().round(3).to_latex(

```



```

45     label="GradeStatsAvg",
46     caption="Statistics for grades of the DD1315 course, grade map: " +
47         ", ".join([f"{g} $\mapsto$ {s}" for g, s in grade_map.items()]) + "."
48 ))

```

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.

```

154a  <data.py 154a>≡
      import csv
      import datetime
      import ladok3
      import os
      import sys

      <functions 155b>

      def add_command_options(parser):
          <add data parser to parser 154b>
          <add data command arguments to data parser 155a>

      def command(ladok, args):
          <produce data about course specified in args 154c>

```

### 17.1.1 Setting up the data command options

We add a subparser. We set it up to use the function `command`.

```

154b  <add data parser to parser 154b>≡ (154a)
      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.

```

154c  <produce data about course specified in args 154c>≡ (154a)
      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.

```

155a <add data command arguments to data parser 155a>≡ (154a) 155c>
    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.

```

155b <functions 155b>≡ (154a) 156a>
    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.

```

155c <add data command arguments to data parser 155a>+≡ (154a) <155a 158a>
    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`.

```

156a  <functions 155b>+≡ (154a) <155b 157a>
      def extract_data_for_round(ladok, course_round, args):
          <compute start and length of the course 156b>
          <get the results for the course round 156c>

          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 159a>

              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 157e>

              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.

```

156b  <compute start and length of the course 156b>≡ (156a)
      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.

```

156c  <get the results for the course round 156c>≡ (156a)
      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.

```
157a <functions 155b> += (154a) <156a 157b>
    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.

```
157b <functions 155b> += (154a) <157a 158b>
    def filter_component_result(component, results):
        for component_result in results:
            <get the component result data 157c>
            <check component code in result data 157d>
            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.

```
157c <get the component result data 157c> = (157b)
    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.

```
157d <check component code in result data 157d> = (157b)
    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.

```
157e <extract grade and normalized date from result data 157e> = (156a)
    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.

```

158a  <add data command arguments to data parser 155a>+≡      (154a) <155c 158c>
        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.

```

158b  <functions 155b>+≡      (154a) <157b 159b>
        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.

```

158c  <add data command arguments to data parser 155a>+≡      (154a) <158a
        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.



## 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<sup>1</sup>.

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 positional argument, and in this case we read CSV data from standard input.

### 18.1 The report subcommand

This 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.

```
160 <report.py 160>≡
    import csv
    import datetime
    import ladok3
    import sys

    <functions 161b>

    def add_command_options(parser):
        report_parser = parser.add_parser("report",
            help="Reports course results to LADOK",
            description="Reports course results to LADOK"
        )
```

---

<sup>1</sup>Optional, we set today's date if not present.

```

report_parser.set_defaults(func=command)
    <add report command arguments to report parser 161a>

def command(ladok, args):
    <report results depending on args 164>

    As mentioned above, we provide two ways of doing this.
161a <add report command arguments to report parser 161a>≡ (160) 161c>
    one_parser = report_parser.add_argument_group(
        "one result as positional args, only date is optional")
    <add one result group arguments 161d>
    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 163b>

```

## 18.2 Report the results

We provide two functions, one for each way of reporting the results.

```

161b <functions 161b>≡ (160)
    def report_one_result(ladok, args):
        <report results given in args 162c>

    def report_many_results(ladok, args):
        <report results given in stdin 163a>

```

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.

```

161c <add report command arguments to report parser 161a>+≡ (160) <161a
    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.

```

161d <add one result group arguments 161d>≡ (161a) 162b>
    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.

```

162a  <check that we got all positional arguments 162a>≡ (162c)
        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.

```

162b  <add one result group arguments 161d>+≡ (161a) <161d
        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.

```

162c  <report results given in args 162c>≡ (161b)
        <check that we got all positional arguments 162a>
        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.

```
163a <report results given in stdin 163a>≡ (161b)
      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 163d>
```

We need to add an argument for the delimiter.

```
163b <add many results group arguments 163b>≡ (161a) 163c>
      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.

```
163c <add many results group arguments 163b>+≡ (161a) <163b
      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`.

```
163d <report a result read from stdin 163d>≡ (163a)
      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.

```

164  <report results depending on args 164>≡ (160)
    if args.course_code:
        report_one_result(ladok, args)
    else:
        report_many_results(ladok, args)

```

## Part V

# Other example applications

## Chapter 19

# 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>.

### 19.1 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<sup>1</sup>.

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:

---

<sup>1</sup>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.

### 19.1.1 Building files

We provide the following files:

- `<canvas2ladok.mk 167a>`.
- `<Dockerfile 167b>`,
- `<canvas2ladok.py 168>`,

We provide an include file for GNU make to build all the targets.

```
167a <canvas2ladok.mk 167a>≡
.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.

```
167b <Dockerfile 167b>≡
FROM dbosk/ladok3

RUN pip install --no-cache-dir canvasapi
COPY canvas2ladok.py ./

CMD ["python", "./canvas2ladok.py"]
```

## 19.2 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.

```

168  <canvas2ladok.py 168>≡
    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 170a>
    <functions 169a>

    def main():
        <main body 169d>

    if __name__ == "__main__":
        main()

```

### 19.3 Getting the results out of Canvas

We must first select the course. We do this by matching against a regular expression.

```
169a <functions 169a>≡ (168) 169b>
    @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.

```
169b <functions 169a>+≡ (168) <169a 169c>
    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.

```
169c <functions 169a>+≡ (168) <169b 169e>
    def students_results(course, component):
        <produce results for a component 172>
```

We cover the details in Section 19.6, once we've covered what LADOK needs in Section 19.5.

### 19.4 The main body

Then we can report those results to LADOK. In summary, we do the following:

```
169d <main body 169d>≡ (168)
    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)
```

### 19.5 Putting the results into LADOK

We get a list of results for a component. We simply try to report them to LADOK.

```
169e <functions 169a>+≡ (168) <169c 170c>
    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}")

```

## 19.6 Producing the list of results

We need the result to have the following data:

```

170a <classes 170a>≡ (168) 170b>
    Result = collections.namedtuple(
        "Result", ["person_nr", "date", "grade", "grade_scale"])

```

Now, given a component, we should produce such `Result` tuples.

### 19.6.1 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.

```

170b <classes 170a>+≡ (168) <170a
    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.

```

170c <functions 169a>+≡ (168) <169e 171a>
    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.

```

171a  <functions 169a>+≡ (168) <170c 171b>
      def assignments_in_component(course, component):
          return filter(
              lambda assignment: assignment.assignment_group_id == component.id,
              course.get_assignments())

```

### 19.6.2 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.

```

171b  <functions 169a>+≡ (168) <171a>
      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)

```

### 19.6.3 Convert to LADOK result form

Lastly, we must put these things together and convert the result to the `Result` format that we need.

```

172  <produce results for a component 172>≡ (169c)
    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

```