Stoner Cheat Sheet

Loading a data file

```
>>> import Stoner
>>> d=Stoner.DataFile('my_data.txt')
>>> d=Stoner.DataFile(False)
      #brings up a file dialog box
```

Valid file types: DataFile, VSMFile, BigBlueFile, CSVFile, XRDFile, SPCFile, BNLFile, TDMSFile, QDSquidVSMFile, OpenGDAFile, RasorFile, FmokeFile

Looking at data

As a whole:

```
>>> d.data
>>> d.column_headers
>>> d.metadata
```

Columns:

```
>>> d.column(0)
>>> d.column('Temperature')
>>> d.column('Temp') #complete label unnecessary
>>> d.column(['Temperature',0])
>>> d.Temperature
```

Rows:

```
>>> d[1]
>>> d[1:4]
```

Specific:

```
>>> d[10,0]
>>> d[10,'Temp']
>>> d[0:10,['Voltage','Temp']]
```

Getting the index of a column:

```
>>> i=d.find_col('Temp')
>>> [i1,i2]=d.find_col(['Temperature','Resistance'])
```

Getting an iterable of the column/row:

```
>>> d.rows()
>>> d.columns()
>>> for row in d: ...
```

Searching:

```
>>> d.search('Temperature',4.2)
>>> d.search('Temperature',4.2,['Temp',
              'Resist']) #returns only 2 columns
>>> d.search('Temperature',
 lambda x,y: x>10 and x<100)
>>> d.unique('Temp')
>>> d.unique(column,return_index=False,
                            return_inverse=False)
```

Copying:

```
 >>> t=d.clone
```

Modifying data

Appending data

```
>>> a=Stoner.DataFile('some_new_data.txt')
>>> d=d+a   # + used to append rows of data
>>> d=d&a   # & used to append columns of data
>>> d.add_column(numpy.arange(100), 'NewCol')
>>> d.add_column(lambda x: x[0]-x[1], 'NewCol')
    #see also AnalyseFile.apply
```

Swap, reorder and rename columns:

```
>>> d.swap_column(('Resistance','Temperature'))
>>> d.swap_column(('Resistance','Temperature'),
                                 headers_too=False))
>>> d.reorder([1,3,'Volt','Temp'])
>>> d.rename('old_name','new_name')
>>> d.rename(0,'new_name')
```

Sort columns:

```
>>> d.sort('Temp',reverse=False)
```

Delete rows and columns:

```
>>> d.del_rows(10)
>>> d.del_rows('X Col',value)
>>> d.del_rows('X Col',lambda x,y:x>300)
  #x is value in 'X Col' y is complete row
>>> d.del_column('Temperature')
```

Saving data

Data saved in TDI format (tab delimited with first column reserved for metadata), or CSV formatted with no metadata.

```
>>> d.save()
#saves with the filename that it was loaded with
>>> d.save('edited_data.txt')
```

Multiple data files

Recursively import a folder structure:

```
>>> f=Stoner.DataFolder('C:\MyData\')
>>> f=Stoner.DataFolder(False) #dialog window
>>> f=Stoner.DataFolder(multifile=True)
 #select a few files from a folder to process
>>> f=Stoner.DataFolder(False, pattern='*.txt')
 #only .txt files in folder picked
```

Look at files and do something with them:

```
>>> f.files
>>> for fi in f: fi.save() #fi is a DataFile
>>> f[1].column_headers
```

Plotting data

2D:

```
>>> p=Stoner.PlotFile(d)  #where d is a DataFile
>>> p=Stoner.PlotFile('mydata.dat')
>>> p.plot_xy('Magnetic F', ['Moment', 'Suscepti'])
 #only partial column label required
>>> p.plot_xy(2,3) #plot column 2 against 3
>>> p.plot_xy(colx,coly,'ro') #use red circles
>>> p.plot_xy(x,[y1,y2],['ro','b-'],figure=2, \
      yerr='Moment err',plotter=errorbar )
```

and after - options for editing the plot:

```
>>> p.xlabel='new label'
>>> p.title='new title'
>>> p.xlim=(-10,10)
>>> import matplotlib.pyplot as plt
>>> plt.semilogy()
```

3D:

```
 >>> p.plot_xyz(xcol,ycol,zcol,
                    cmap=matplotlib.cm.jet)
```

Analysing data

Load the data:

```
>>> a=Stoner.AnalyseFile(d) #d is a DataFile
```

Do maths on the data:

```
>>> a.subtract('A','B', header="A-B",replace=True)
>>> a.subtract(0,1) #subtract col 1 from col 0
>>> a.subtract(0,3.141592654) #subtract pi from col 0
>>> a.subtract(0,a2.column(0))
  #also can use a.add, a.multiply, a.divide similarly
>>> a.apply(func, 'Momen', replace=True, header='data_edit')
    #func accepts a row of data and returns a float
>>> a.normalise('Signal_col', 'Reference_col')
>>> a.normalise('Moment', max(a.column('Moment'))
 #last example normalises the column maximum to 1
```

Other functions available are interpolate, threshold, integrate and peaks.

Split the data into a DataFolder object according to the value in a certain column:

```
>>> f=a.split('Temperature', lambda x,r: x>100)
 #x is the Temperature value, r is a list of all values in row
>>> max( f[0].Temperature ) #outputs 99.5
>>> max( f[1].Temperature ) #outputs 300.1
```

Fit the data:

```
>>> a.polyfit(xcol,ycol,order, result="New Column")
>>> a.curve_fit(func,  xcol, ycol, p0=None, sigma=None,
      bounds=lambda x, y: True, result="New column")
>>> (Stoner.PlotFile(a)).plot_xy(xcol, "New Col")
```

`polyfit` and `curve_fit` are the same as the scipy functions. Both accept bounds on fitting region. func should be `def f(xdata,p[0],p[1]...)`. `p0` is the initial parameter guess.
More sophisticated fitting using nlfit. In this case build a .ini file to define fit (see example in scripts)).

```
>>> a.nlfit("fit.ini", func)
```

`def func(xcolumn, params)` and returns a column of data. func can also be a str naming one of the functions in FittingFuncs.py eg 'BDR', 'Simmons', 'Arrhenious', 'WLfit'.