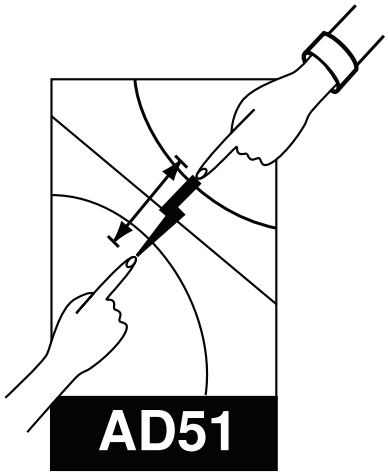
Python Cheat Sheet



JB after KLA August 2019

1 Introduction to Python

We suggest that demonstrators, and any students who want to, take the straightforward, quick on-line free introductory course 'Introduction to Python' from DataCamp[1] or the Learn Python course from learnpython.org [2].

2 Python hints and tips

We recommend using Python either via Jupyter notebook or using Spyder. Both can be accessed by looking for Anaconda on the start menu from any Windows PC in the Teaching Laboratories and finding the icon for Jupyter or Spyder.

In Jupyter:

- Remember to run each cell sequentially by pressing Shift and Enter together or using the 'Run' button at the top of the notebook

In Spyder:

- Pressing the up arrow repeatedly whilst you have clicked in the Console window will permit you to review and edit old commands so that you can reuse them. You can also see your command history in this pane.
- "Tab auto-completion" means that if you press the tab key after starting to type in a variable name, you will get a list of suggestions (use the up/down arrows to select one and then press Enter to add it to your command sequence).
- Running code from a script: there are several ways to do this but we recommend either (1) highlighting or selecting the parts of the code you want to run and clicking the Run selection button or (2) clicking the Run file button to run the whole script. *Note that this is different to MATLAB.*

3 Key Python functions useful for basic data analysis and plotting

See following page.

[1] `https://www.datacamp.com/courses/intro-to-python-for-data-science`
[2] `https://www.learnpython.org`

 Copyright © 2022 University of Oxford, except where indicated.

```
help(f)               # built in help for function f
import numpy as np    # read in the library numpy and call it np
import matplotlib.pyplot as plt   # read in the plotting library matplotlib and call it plt
import pandas as pd   # read in the library pandas and call it pd
df = pd.read_csv("datafile.csv")   # read CSV file into dataframe variable df
x = df[df.columns[0]].values    # assign first column to vector x
y = df[df.columns[1]].values    # assign second column to vector y
plt.scatter(x, y, marker='o', label='data')
                      # basic scatter plot of (x,y) data with label 'data'
plt.xlabel('Xlabel')          # Plot a label on x axis of Xlabel on graph
plt.legend()          # Plot a legend on graph
plt.hist(y)           # histogram of data in y
df.describe()         # general information about data in dataframe variable df
%matplotlib inline    # Use in Jupyter Notebooks so plots display nicely
```

Linear fit (using statsmodel library):

```
import statsmodels.api as sm     # import stats package
X = sm.add_constant(x)           # add a constant to fit
results = sm.OLS(y, X).fit()     # save results of fit
print(results.summary())         # print results out to screen
plt.plot(x, results.params[0]+results.params[1]*x, 'r' , label='fitted line')
                                 # plot fitted line on graph
```

Plotting data:

```
plt.plot(x,y ':', colour="black", label="y = x")    # plot x vs y as a dotted black line
plt.plot(x,y '--', colour="blue", label="y = 2x")   # plot x vs y(=2x) as a dashed blue line
plt.grid(True)       # plot gridlines
plt.xlim(0,100)      # set limit on x axis of 0 and 100
plt.xscale('log')    # set (base 10) log scale on x axis
plt.savefig('simpleplot.png')    # saves plot to a PNG file, must come before show command
plt.show()           # plots final graph
```

Plotting data with errors in y (x ± dx , y ± dy)**:**

```
yerr = dy       # error in y, can be a single value or an array
xerr = dx       # error in x - can be set to 0 if not needed
plt.errorbar(x, y,yerr,xerr, label='y=2x') # note order of data and error bar information,
                    # xerr can be removed from command if required
plt.show()      # plots final graph
```

Nonlinear fitting:

```
from scipy.optimise import curve_fit   # import curve fitting package from scipy
def exponential(x,a,b,c):              # define the fitted function name
     return a*np.exp(-b*x)+c           # define the form of the function
popt,pcov=curve_fit(exponential,x,y,p0=(A,B,C)) # use the function to fit data
                     # need to specify initial guessess A, B, C for fit
a_fit=popt[0]      # fitted slope is popt[0]
b_fit=popt[1]      # fitted exponential parameter is popt[1]
c_fit=popt[2]      # fitted intercept is popt[2]
uncertainties=np.sqrt(pcov.diagonal())   # uncertainties are square root of
                           # diagonal elements in covariance matrix
```

Other:

```
np.median(data)     # median value of the variable 'data'
np.mean(data)       # mean value of the variable 'data'
len(data)           # length/number of elements in the variable 'data'
```