Python Quickstart and Conversion Exercises

Python is already installed on the University systems. This comes with a graphical interface very like the one available for MatLab. To start this up you should:

1. Go to the search box available bottom left corner of the windows screen and type in “python”
2. Select **(UoN) Spyder (Python 3.4)**

Be careful to avoid the Python 2.7 version of Spyder which is also available.

This should bring up a 3-panel interface similar to the one in MatLab. The lefthand panel is the editor, the top-right panel allows you to examine objects (for example the values of your variables) and the bottom right panel is the console/command window. Across the top are action buttons that will save, run, debug your code.

To execute a simple python script, type **print(‘Hello’)** into the editor and execute it using the **run** action button (the 5th from the left at the top). You should see some output in your console panel, bottom right.

Within Python the numerical elements are packaged into something called “numpy” and the Matlab-like plotting routines into “matplotlib.pyplot”. To get these to work you will need to import them, so many of your routines will contain the lines:

```
import numpy as np
import matplotlib.pyplot as plt
```

near the top. Commands from these packages are prefixed by **np** or **plt** but will otherwise be familiar from MatLab, e.g.

```
plt.plot(x,y)
t=np.linspace(0,10,100)
```

One major issue: python is ‘zero indexing’ like C. This means the array index starts at zero, i.e. it runs from [0,n-1] rather than [1,n] as in Matlab.

There are a lot of python examples on the web. Searching the web for the problem you have usually throws up similar sample source code that you can then adapt. The same holds for error messages.

Common Commands and Structures

Usage	Command
Command line input	str1=input('Enter something')
Command line output	print('Hello')
Range of numbers	range(n)
Array of numbers	h=np.array([2.2,2.1,1.8,2.1])
Array of zeros	A=np.zeros(n)
Length of array	L=len(a)
Linspace	t=np.linspace(-10,10,51)
Meshgrid	X,Y=np.meshgrid(x,y)
Random integers	R=np.random.randint(1,13,n)
New figure window	plt.figure(1)
Basic plot	plt.plot(x,y)
Axis labels	plt.xlabel('Time')

For loop syntax. Note the final colon and the spacing at the start of the second and third lines. These are the lines that will be looped over. Also notice the limits of the loop are set by the 'range' command and that this **does not** include the upper bound (range(n) runs from 0 to n-1, a range with n elements).

```
for a in range(bottom limit, top limit+1):
   <do this>
   <and this>
<but not this>
```

While loop syntax. The Python while loop syntax is very similar.

```
while <condition is true>:
   <do this>
<but not this>
```

Conditionals. Are also similar:

```
if <condition is true>:
  <do this>
else:
   <do this>
   <and this>
<continue from here>
```

Functions. Ditto.

```
def functionname(varin1,varin2,):
   <do this>
   return(varout)
```

Example problems

1. Request a user to supply a positive integer. Once you have obtained a positive integer, request this many further inputs using a loop.
   Get your program to display, using a print statement with appropriate text, the sum, mean and median of the values entered.

2. Generate 1200 uniformly distributed random integers between 1 and 12.

(a) Count how many times each number is obtained.

(b) Produce a labelled histogram of your results.

3. The Leibniz approximation for π is

$$\pi = 4\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

(a) Display the result of the Leibniz sum as a function of n for the first 20 terms of the series. Label your axes.

(b) Extend your routine to evaluate π to an accuracy of 0.01. Note: you should avoid using the known value of π as part of your calculation.

4. It can be shown that the sum of the squares of the integers k from 1 to n is:

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

(a) Write a program to request a positive integer n. You may assume that the value supplied is a positive integer without testing for this.

(b) Create a vector containing the integer values from 1 to n and use it to calculate a vector containing the values of k^2. Sum this vector.

(c) Use a print statement with appropriate text to display the value of your sum and the result of the equation above for the value of n supplied.

(d) Calculate the value of $\sum_{k=1}^{n} k^2$ using a for loop and get your program to print out the result.

5. When particles fall on to a surface (such as snow flakes on a window pane or grains of sand on a sand dune), the shape of the surface onto which the deposit falls will vary with time. The roughness of the surface is characterised by its width w, defined to be

$$w = \sqrt{\frac{1}{L}\sum_{i=1}^{L}\left[h(i) - \bar{h}\right]^2}$$

where the h(i) are i measurements of the height of the surface at L positions, and $\bar{h}$ is the mean of the measurements h(i). The height at various points along the surface at a given time are

2.2, 2.1, 1.8, 2.1, 2.0, 1.9

(a) Enter these values into a 1D array and use them to calculate $\bar{h}$ using the appropriate inbuilt function.

(b) Calculate the width w from the equation above.

(c) Identify those heights which are greater than $\bar{h} + w$, and those which are less than $\bar{h} - w$, printing out the results with appropriate text.

6. The distribution of energies of molecules in a gas follows the Maxwell-Boltzmann distribution function, F(E), given by

$$F(E) = \frac{2}{\sqrt{\pi}}\left(\frac{1}{2kT}\right)^{3/2} E^{\frac{1}{2}}\, exp\left(-\frac{E}{kT}\right)$$

where E is the energy, T is the temperature in Kelvin and k = 1.38×10^{-23}JK^{-1}.

(a) Calculate F(E) as a function of E for the three values of T = 10^7K, 2×10^7K and 3×10^7K. For each temperature, use 100 values of energy between 0 and 1.5×10^{-15}J. You should only need one line containing the formula.

(b) Plot the three curves in different colours on the same graph. Include axis labels and a legend.

7. The solution to the diffusion equation in 1D may be written as

$$n(x,t) = \frac{N}{\sqrt{4\pi Dt}} e^{-x^2/4Dt}$$

where n(x,t) is the concentration of particles at position x at time t. Take N = 10^4 as the total number of particles, and D = 1 as the diffusion coefficient.

(a) Calculate a 1D array of 30 values of t between 1 and 10, and another 1D array of 51 values of x between -10 and 10.

(b) With the help of the function np.meshgrid, set up a 2D array of the values of n(x,t) at the defined values of x and t.

(c) Plot n(x,t) as a surface plot with properly scaled and labelled axes.