Exercises Sheet 3

This set of exercises follows on from the topics discussed in the 'Python Introduction Part 3'. The focus is on using the control statements (e.g. for, if) and functions to build code that can complete tasks.

These exercises will need to be completed in script. You should create a script for each exercise. At the start of each script you will need to import numpy *and* matplotlib*.*

Remember - for any function you haven't met before, you can use ? to find out information, e.g. ?print

1. If else elif

a. Write an if else statement to check whether a variable, *invar*, equals 5. If it equals 5 then set a variable, *outvar,* equal to 9. If it does not equal 5 set *outvar* equal to 6. Try the statement for *invar=5* and print the result of *outvar*.
b. Write an if else statement that takes the value *outvar* and checks if it is greater than 8. If this is true then multiply it by 6 and set result to *outvar2*. If it is not true, then divide by 3 and set result to *outvar2*. Set the 6 and 3 to named variables inside the statement. Print the result
c. Use an if statement to test whether string 'hi' is EQUAL to string 'HI'. If the statement is true then print string 'TRUE' and if it is false use ELSE to print string 'FALSE'
d. Do the same to test whether 0 us equal to null string ' ' (two single quotation marks)

2. Loops

a. Create a *for* loop that prints the statement 'Hi there' ten times.
b. Create a *for* loop that increases a variable by 1 each time it loops. Execute the loop 20 times for a variable initially set to 0. Print out the final result.
c. Write a *for* loop that iterates over the elements in following list, printing out the elements separately.

```
star_temp = [38000,30000,16400,10800,8260,7240,6540]
```

d. Modify the *for* loop such that it prints the index of the element as well.
e. Create the following dictionary for star's temperatures:

```
star_temp = {'Theta1 Orionis C':38000,
        'Phi1 Orionis':30000,
        'Pi Andromedae A':16400,
            'Alpha Coronae Borealis A':10800,
```

'Beta Pictoris':8260,

'Gamma Virginis':7240,

'Eta Arietis':6540}.

Code a *for* loop to iterate over the dictionary keys, printing out the results.

f. Using the previous dictionary, code a for loop to iterate over the keys and values, printing out the results. The printout should read, 'The temperature of x is y K'.

g. Set a variable to value 0, then write a one line WHILE loop to add 1 to the variable until it reaches 100.

h. We want to write a basic control system for pointing a satellite. We know the offset from the target and want to correct it. Create an initial variable called offset and set it to 8. Code a while loop to correct the offset by 1 and print out the word correcting and the current value of offset. Stop the loop when the offset equals 0.

3. <u>Functions</u>

a. Write a function that takes any number in, multiplies it by 2 and returns the new value.

b. Re-write the above function so that the multiplicative value is a default argument. Test the function by giving it two arguments.

c. Write a function that plots a labelled sine curve. The function should be able to take in values of amplitude, period and phase. Make the amplitude and phase default parameters with values of 2 and 0. The plot title should show the values for the period, amplitude and phase.

d. Write a function that takes in an array and returns multiple outputs, namely the minimum value, maximum value and mean value.

e. Create a function that can take in any number of floats and return the sum of them all. You will have to use flexible arguments.

Harder exercises

You will have to use the help function or web-searches to find out how to use any new functions.

4. The control system you created in exercise 2h. only works for positive values of the offset. Incorporate *if else* statements so that it works with negative offset values as well. Test this with an initial offset value of -6.

5. We are interested in trying to mimic the outcomes of a coin toss. We want to create a script that can help us do this.

   In order to mimic the result of a toss, we can use a random number generator. I have used these in various examples through the course. The one we will use here is called np.random.randint() – and as the name suggests will generate a random integer. You should read the docs to see how it works.

Using randint, create a function that takes in the number of flips as an argument. The script should then use a *for* loop to call the random number generator. The value of the number generator should then be used to build a list of outcomes, with entries 'heads' or 'tails'. The function should return the list.

Note that the random number generators in Python (and other languages) are psudeo-random number generators. This means that you can set a seed value to get repeat performance. Try setting a seed value in the function, with np.random.seed().

6. Let's now modify the coin flipping script. Instead of returning the string values of the coin flip, we are interested in the number of heads you get after 10 coin flips (we will call this our experiment). Modify your function to return this value. We now want to calculate the probability of getting X number of heads from the 10 coin flips. To do this, re-write the function to perform the experiment 10,000 times and save the results. Plot a histogram of the results.

7. Now we will try to model a random walk process. We will do this by rolling a dice with 6 faces. If you score below a 4 you move forward, 4 or above you move backward. The steps you move is determined by another role of the dice. Create a function that uses randint to model the dice throwing. The inputs to the function should be the number of initial throws. You should start at position 0. The function should return the list (or array) of positions you occupy. Plot the outcome.