# CM1103 Lab Worksheet Week 9

## matplotlib

`matplotlib` is a commonly used Python module for plotting figures[1]. Read the first two examples in the tutorial at `http://matplotlib.org/users/pyplot_tutorial.html`.

1. Plot the set of student marks given by `examMarks = [25, 72, 83, 91, 61]` with red circles and appropriately scaled and labelled axes.

2. Given sets of student marks `examMarks = [25, 72, 83, 91, 61]` and `cwkMarks = [56, 90, 45, 62, 60]`, plot the examMarks (on the x-axis) against the coursework marks (on the y-axis), with blue crosses and appropriately scaled and labelled axes.

3. Type the code:

```
fig = plt.figure()
ax = fig.add_subplot(111)
```

   You can now plot multiple data series on the same axes using the function call `ax.plot(...)`. Plot the exam marks and the coursework marks as separate series in this way. Use the example at `http://stackoverflow.com/a/19125863` to add an appropriate legend.

## Random Number Generation *Think Python! 13.2*

Download the file `MyRandom.py` from Learning Central. This file contains a function called `random` which implements a random number generator. Test out this random number generator by using the following code:

```
import MyRandom
for i in range(5):
   print(MyRandom.random())
```

4. Using the template below, write a function that will call the random function multiple times, putting each random number into a list.

```
def RandExperiment(n):
   randomNumbers = []
   for i in range(n):
      currentNumber = ??
      ??? Add the item to the list
   return randomNumbers
```

[1]It is often used and bundled with `numpy` and `scipy` for scientific computing.

5. Run this RandExperiment function with different values (such as those shown below) and comment on how "random" this random number generator actually is.

```
RandExperiment(10)
RandExperiment(50)
RandExperiment(100)
RandExperiment(200)
```

6. From these experiments determine what the period of this random number generator is. The period is the number of results that are produced before the numbers start repeating again from the start.

7. This random number generator uses three variables to control the numbers it generates. Modify these values of *c*, *m* and *a* to try to improve the performance of the random number generator.

8. Use the values shown below and work out what the period of this random number generator is.

```
m = 83
a = 84
c = 53
```

## Classes *Think Python! Chapter 15*

**Before starting this section you should read Chapter 15 of Think Python.**

9. Define a class with a single function by entering the following code:

```
class MyName:
   def printMe(self):
      print("{:s}␣{:s}".format(self.firstname,self.surname))
```

   (a) This class represents a person's firstname and surname.

   (b) Test the class out by entering the following code

   ```
   me=MyName()
   me.firstname="Stuart"
   me.surname="Allen"
   me.printMe()
   ```

   (c) It is important to note that the actual variable names are only used in the `print` function. Enter the following code and see what happens when you try to use `printMe` before assigning firstname and surname.

   ```
   me=MyName()
   me.printMe()
   ```

   It is also important to note that every function in the class must be passed the `self` parameter, this is so that the function can see the class that it is part of.

(d) We can make this more elegant. Instead of defining a function `printMe`, define a function `__repr__(self):` that returns the string representation instead of printing it. You should now be able to print objects of the class using the built in `print` function (e.g. `print(me)`).

(e) Modify this class so that it copes with a single middle name in addition to firstname and surname

10. Download the template file `PhoneBook.py` from Learning Central. In this file we are going to define a class called `PhoneBook` which will hold contact details for a number of people.

    (a) First define an empty class called `PhoneBookEntry`.

    (b) The first function of the PhoneBook class has been done for you. This method is the init function, and it creates a dictionary called data when the PhoneBook is first created.

    (c) The second function adds entries to the phone book. Complete this function.

    (d) The third function *delEntry* will delete an entry from the phone book. Complete this function.

    (e) The fourth function should be called *exist* and should have the name of a person as a parameter. The function should return *True* if the person exists in the phone book and false if they do not.

    (f) The final function, which you must create, will be called *printBook* and it should print out the data for every item in the phonebook. You will need to use the code for looping through a dictionary that was given to you on Week 8's lab sheet.

    (g) Use the following code to test out the PhoneBook you have just created:

```
myPhoneBook=PhoneBook()
myPhoneBook.addEntry("Stuart␣Allen","02920222222","S.M.Allen@cs.cf.ac.uk")
myPhoneBook.addEntry("Tom␣Beach","02920111111","T.H.Beach@cs.cf.ac.uk")
myPhoneBook.exist("Stuart␣Allen")
myPhoneBook.exist("Tom")
myPhoneBook.printBook()
```

    (h) Add other entries to the phone book and write code to test out the other functions you created, such as deleting entries from the phone book.

## List Generators

11. A list generator is used to create a list of items so that a data structure can be iterated over. Create a list:

```
myList=[1,2,4,8,16,32,64,128,256,512,1024]
```

We can then make a *generator* to move through the list backwards. They key command here is that we now use *yield* instead of return:

```
def myReverse(data):
   for index in range(len(data)-1,-1,-1):
      yield data[index]
```

We can now iterate through the list in reverse. Enter the following code and check that it does what you expect.

```
for i in myReverse(myList):
   print(i)
```

12. Define a generator similar to that in Question 11 that will allow you to iterate over a list (starting at the begining), but only yielding every n$^{th}$ element. The function definition and an example is provided below to start you off:

```
def myNth(data,n):
   ??complete this method

myList=[1,2,4,8,16,32,64,128,256,512,1024]
for i in myNth(myList,2):
   print(i)
```

For reference the example given above will produce the following output:

```
1
4
16
64
256
1024
```

## Regular Expressions

The following are some example regular expressions:

- [0-9] - Will match any numbers from 0 to 9.
- [a-z] - Will match any lower case letter
- To match a specific letter simply enter that letter. I.e. T will match T .
- Adding a * after a letter means: match zero or more. I.e. [a-z]* will match zero or more lower case letters.
- Adding a + after a letter means: match one or more. I.e. [a-z]+ will match one or more lower case letters.
- Adding {N} after a letter means: match N of those letters. I.e. [a-z]{5} will match 5 lower case letters.
- Adding {N,M} after a letter means: match between N and M of those letters. I.e. [a-z]{5,6} will match between 5 and 6 letters.
- For further help see, http://www.regular-expressions.info/tutorial.html

13. Use the information above to write a regular expression which will enable you to validate the following data:

    (a) A current Cardiff University username: The letter 'c' , followed by seven numbers between 0 and 9. I.e.

    ```
    c1234567
    ```

    (b) An old style Cardiff University username: the letters 'scm' followed by a single number, followed by either two or three letters. I.e.

    ```
    scm7uii
    ```

    (c) A bank sort code: Two numbers followed by a - followed by a further two numbers and a - followed by a final two numbers. I.e.

    ```
    09-97-68
    ```

    (d) A current UK car registration number: Two capital letters, followed by two numbers, followed by three capitol letters. I.e.

    ```
    BD51SMR
    ```

## Challenge: *Random marking*

Suppose I have a group of $n$ students. To save me effort in marking their latest assessment, I collect in all of their scripts, shuffle them randomly, then return them to the class and ask them to mark the sheet they're given.

If I set the students 10,000 assignments, what proportion of these will result in at least one student marking their own work? Show how this varies for $1 \leq n \leq 10$.

Email AllenSM@cardiff.ac.uk with your answer and the code you wrote to solve the problem, with the subject line "CM1103 week 9 challenge" by midnight on Wednesday in Week 10 – a winner will be picked at random from the correct answers **but** you'll get one extra entry for particularly noteworthy, concise or interesting code.