CS917 Foundations of Computing
Lab 1: Introduction to Python

Good afternoon,

In today's session you'll be writing your first python programs. There are some starter files on the website https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs917/programming/

When I get stuck with a problem my general plan of attack would be:
1. Think about the problem some more, try writing down a set of steps that would solve the problem in English
2. Translate this to python as best I can. Fix any bugs.
3. Call a tutor to ask for help

Good luck,
Adam

Preamble - Getting setup

We'll need a terminal so that we can compile and run the programs that we write. To open up a terminal, click on the small red hat in the bottom left of the screen and type terminal in the search box. The best one to use is named konsole.

In order to keep all of your programs neat and tidy we'll create a directory for our CS917 code to do this we'll use the command below:

```
> mkdir cs917
```

Now you should change into the directory using the cd command.

```
> cd cs917
```

You can choose to use any text editor that you are familiar with, but if you do not have a strong preference atom is available on all of the machines and offers some nice features like syntax highlighting. Type the command below (remember to add the & symbol).
> kwrite &

Section 1 - Writing Hello, World!

As we discussed in the lecture Hello world is the first program that everyone begins with. Type the program below into the editor.

```
print("Hello, World!")
```

Save the file as hello.py in your cs917 directory.

Now run the program by typing:

```
> python3 hello.py
```

You should now see you message printed on the screen.

Now change the message that your program prints and run out to be sure that it works.

Congratulations! You are now a python programmer.

Section 2 - Seeing Stars

You will now attempt to print out a pattern of stars. Create a files called stars.py that will print out the following pattern:

```
*
**
***
****
*****
```

Now try something a little harder:

```
*****
****
***
**
*
```

There are lots of was to do either of these. It might be a good idea to try producing both patterns using a for-loop and a while-loop so that you are comfortable with both.

Section 3 - Reading in some data

Python makes it easy to read in data from a user. Create a program named ask_stars.py that asks a user for a number of rows of stars to make. A sample execution might be:

```
> python3 ask_stars.py
How many rows of stars should I print? 7
*
**
***
****
*****
******
*******
```

Now that you have some user input, the next step is to ask whether the user would want to have the pattern increasing or decreasing.

You could define a function draw_stars which takes two parameters, the number of stars to draw and the order. This would make your code neater too.

Section 4 - A Simple Guessing Game

We're going to write a simple computer game where we have to guess a number between 1 and 10 that the computer has randomly generated. A sample play through of the game is shown below.

```
> python3 guess_game.py
I'm thinking of a number, can you guess what it is? 6
Too Low! Guess again. 10
Too High! Guess again. 8
Well done. The number is 8
```

Python has a random number module that you can import to generate random numbers. The documentation for this module can be found ahttps://docs.python.org/3.6/library/random.html

In our case we need the **randint** function which can be imported as follows:

```
from random import randint
```

This can now be used to generate random numbers:

```
#This generates a number between a and b inclusive.
random_number = randint(a,b)
```

For bonus points add in three levels of difficulty which the user can select from:
Level 1 (Easy): A number between 1-5
Level 2 (Medium): A number between 1-10
Level 3 (Hard): a number between 1-100

Section 5 - Get started with lists

Given the list in the lists.py file on the website. See if you can:

1. Print the third item in the list
2. Create a list containing the first three items of the list
3. Create a list containing every other item in the list.
4. Test if the list contains January.

Section 6 - A simple stats calculator

The grand finale of today's lab is to build a program called stats.py which takes in numbers from a user and generates some simple statistics about them.

The user will enter numbers (one per line) and enter "stop" when they have finished. After getting the numbers, you should calculate and print:

1. The number of numbers entered
2. The minimum number entered
3. The maximum number entered
4. The mean of the numbers entered

A sample execution might be:

```
>python stats.py
Enter numbers. When finished type 'stop' to calculate statistics.
6
12
34
11
2
stop
=== Results ===
You entered 5 numbers.
Minimumm number:  2
Maximum number: 34
Mean: 13.0
```

There are many ways to solve this problem, I have placed a couple of different solutions on the module webpage.