University of Stirling
Computing Science and Mathematics

CSCU9YE - Artificial Intelligence

Lab 1: Introduction to Python

Python[1] is considered an interpreted language because Python programs are executed by an interpreter. There are two ways to use the interpreter: **interactive** mode and **script** mode. In interactive mode, you type Python programs and the interpreter displays the result:

```
>>> 2 + 2
4
```

The chevron, `>>>`, is the prompt the interpreter uses to indicate that it is ready. If you type `2 + 2`, the interpreter replies `4`.

Alternatively, you can store code in a file and use the interpreter to execute the contents of the file, which is called a script. By convention, Python scripts have names that end with `.py`.

To run Python programs you can use the integrated development environment (IDE) for Python called IDLE, which is bundled with the default implementation of the language. IDLE has two main window types, the **Shell** window and the **Editor** window. We will use both in this lab.

Alternative, you can use this online IDE for completing the lab. Repl. it is a powerful and simple online interpreter for Python (and other languages). REPL stands for Read-Eval-Print-Loop. It also provides two window types, the **Shell** window and the **Editor** window.

https://repl.it/languages/python3

Warm Up Using the Interpreter

We will conduct first some computations using the interactive mode. For this we will use the IDLE's shell window. You can start by using the Python interpreter as a calculator and try arithmetic expressions such as:

```
>>> 2*50
```

Let us now type the "Hello World" Program in Python:

```
>>> print 'Hello World'
```

This is an example of the `print` statement, where the quotation marks indicate the beginning and end of the text to be displayed. Notice that in Python 3, the syntax for printing is slightly different: `print('Hello World')` where the parenthesis indicate that `print` is a function.

[1] There are two versions of Python 2 and 3. We will be using version 2 in the lab sheets as this is the version installed in the lab. We will indicate sometimes the difference between the two versions.

Lists are an important data structure in Python. A list is a sequence of values. The values in a list are called elements or sometimes items. There are several ways to create a new list; the simplest is to enclose the elements in square brackets `[` and `]`:

```
[10, 20, 30, 40]

['Jane Smith', 'Andy Clark', 'Mary Taylor']
```

As you might expect, you can assign list values to variables, you can also create an empty list and later dynamically append to it:

```
>>>  numbers = [10, 20, 30, 40, 50]

>>>  names = ['Jane Smith', 'Andy Clark', 'Mary Taylor']

>>>  mylist = []        # Empty list
```

Experiment with printing the values of a list using the `print` statement, eg. `print numbers` , and accessing and printing particular elements of a list, eg. `print numbers[3], print numbers[0:2],` this last syntax express a range of values. Also you can append elements to a list with the list function `append,` for example: `mylist.append(50)` or `names.append('Tim Williams')`

Lists are mutable, which means that their values can be changed after the list has been created. When the bracket operator appears on the left side of an assignment, it identifies the element of the list that will be assigned. For example, you can modify the 2nd element in the list numbers with:

```
>>> numbers[1] = 200
```

If you print the variable, you can see that it has changed.

A useful function in Python is `range`, which generates a list of consecutive integer numbers (notice that in Python 3 the function `range` does not produce a list but its own different data type). For example type

```
>>> range(20)
```

Range is a useful function to use together with loops. For example, test the effect of the following lines of code in your interpreter. Notice the the function `len()` returns the length of the object passed as argument.

```
>>> for i in range(len(numbers)):
        numbers[i] = numbers[i] * 2
```

Exercise 1: Practice using the interpreter as a calculator

The volume of a sphere with radius r is $4/3\ \pi\ r^3$. What is the volume of a sphere with radius 5?
Hint: math.pi (math module) is the mathematical constant "π", to available precision on your computer.

Exercise 2: Loops, Lists and Functions

Now let us turn to write and execute a script. It is important to write modular code, so let us define and use a function. Below there is an example of a simple function that receives two parameters and returns two values. Specifically, it receives two numbers and returns their sum and difference. You can test the following code in your script.

```
def test_function(a, b):
    val_sum = a+b
    val_dif = a-b
    return val_sum, val_dif

#call the test function. Variables val1 and val2 will contain the returned
values
val1, val2 = test_function(45, 20)
```

It is acceptable and commonplace to use Internet search to learn the syntax for something in an unfamiliar language. Python in particular has a lot of online documentation.

Your task is to create a list of numbers with the hypothetical ages of a group of people. The ages will be randomly-generated, and we assume that the minimum age is 1 and the maximum 120.

> *Hint*: use the function from the `random` Python library, which you can import with the following line (`import random as rnd`). Specifically, the call: `rnd.randint(0,100)` generates a random integer number between 0 and 99.

You should create a function that assigns the values to the list, and at the same time keeps the sum of ages to be able to compute the average age in the group, after all values have been included. Your function should receive as a parameter the size of the group, and should return two values: a list of ages, and its average. Also your function should print the numbers out as they are added to the list. After calling your function, you should receive the two returning values on two variables. You should then also print out the two variables. Finally to test that your average computation is correct, you should compare your value with computing the average using the predefined function function `sum()` which computes the sum of the elements in a list. Whenever you print a variable, please include text to describe what the number represents.

Show:

- Print out ages as they are added to the list
- Print out of the two ways of computing the average

Exercise 3: Plots

Visualisation is an important part of Data Science. Let us practice with one the most widely used plotting libraries in Python *Matplotlib*[2], a 2D plotting library which produces high quality figures. To import the library you should add the line:

```
import matplotlib.pyplot as plt
```

[2] https://matplotlib.org/index.html

Continuing with the script you have created for Exercise 1, let us add the necessary lines of code to visualise the ages data. First as a standard line plot using the function `plt.plot()` and then using a so called *boxplot* wit the function `plt.boxplot()`
We encourage you to find out using Internet what a box-plot indicates and represents. Also you can improve your plots by adding some visual features to them. You can find examples in:

- https://matplotlib.org/index.html
- https://matplotlib.org/examples/pylab_examples/boxplot_demo.html

Hint: Before plotting each figure, you should call the function `plt.figure()` to generate a new window. Otherwise you will overlap the new plot in the last active window.

- If you are using IDLE or the command line, you should you should call `plt.show()` (after call the `plt.plot` call) to display the plot. Also bear in mind that the window plot pops out and can be hidden behind your other windows.
- If you are using Repl. it, the `plt.show()` is not supported, so you should use the function `plt.savefig()` instead. This function saves the plot to an image file: `plt.savefig('plot.png')`
-

Show: The two plots for the ages data (simple line plot and box plot).