# Big Data & Predictive Analytics
# Lab 1 - Introduction to Python Programming
# Programming Basics

## Objectives

At the end, of this practical, you should be able to

- familiarise with the Python syntax and semantics
- use values and variables, form expressions and statements.
- use iteration and conditional statements
- perform simple IO and make use of Python libraries
- write a Python function and make use of Python standard data structures
- use Packages and basic plotting

## Resources and Reading

Though the practical explain the basics of Python before the practice questions, it is recommended that you consult indicated resources to further your Python programming skills. These exercises are ordered in increasing level of difficulty from the very basic to the challenging ones. See a recommended list of references as follows.

[PT ] http://www.pythontutor.com/

[RA ] Runestone Academy *How to Think Like a Computer Scientist: Interactive Edition*. See https://runestone.academy/runestone/default/user/login?next=/runestone/default/index

[ZS ] Zed Shaw. *Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code*. You will find exercises and Videos. See https://learnpythonthehardway.org/book/.

To write a program, we need to set an objective, understand and use the syntax and semantics of Python so as to write the necessary statements that compose the program. Statements usually manipulate values and variables by using well-defined operators.

N.B. We are going to use Python 3.5 or greater, which you can install by using Anaconda3, See https://www.anaconda.com/.

# Python as a calculator

1. Install Python 3.5 or greater with the jupyter notebook into your laptop, see Anaconda (www.continuum.io/). You may use a different IDE depending on your liking. Here are the instructions on how to install Anaconda.

***Installing Anaconda***

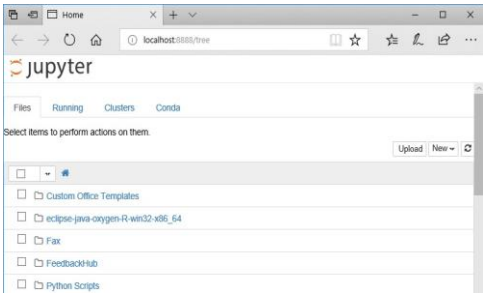
*A. Go to Anaconda website [https://www.anaconda.com/products/individual](https://www.anaconda.com/products/individual) and click on downloads; this will take to the various download options. Currently you can get Python 3.9 but any version above 3.5 will do for this module.*

*B. If you are using Windows search for name to find out if you have a 64-bit machine; most machines are currently 64-bit.*

*C. Select the appropriate executable to download the **.exe** installer. This may take few minutes.*

*D. Open and run the **.exe** installer to install Anaconda. This will install the Jupyter Notebook, which we will use for the Labs. Naturally, you are free to use another IDE (Integrated Development Environment such as PyCharm or Spyder) that you may like most.*

2. Launch jupyter. This will take you to the web browser on the following address: http://localhost:8888.




- The "Files" tab is where all your files are kept
- The "Running" tab keeps track of all your processes

- The "Clusters" tab is a parallel computing framework enabling to control individual engines.

Perform the following tasks:

(a) create a folder CO3093Labs: **click on the "New" button in the "Files" tab and select "Folder"**. Then rename the folder as it will appear as a folder with name 'Untitled Folder' as 'CO3093Labs'.

(b) Create a new notebook as follows: **click on the "New" button in the "Files" tab and select "Python3"**. Then rename the file 'Untitled' as 'Lab1'.

(c) Close the notebook Lab1.ipynb by going to the 'File' tab and selecting 'Close and Halt'; then move the note book to the folder CO3093Labs.

Go to your folder CO3093Labs; click on the 'Lab1' note book and evaluate the following expressions one by cell:

(a) `2//6`

(b) `2/6`

(c) `56%3`

(d) `12. * (2 - 6)**2`

(e) `1 == 1.0`

(f) `3 >= 2 and 3 <= 4`

(g) `3 >= 5 or 3 <= 4`

More on using jupyter-notebook can be found in http://jupyter.readthedocs.io/en/latest/running.html#how-do-i-start-the-notebook-using-a-custom-ip-or-port

# In Python, everything is an object

Python is an object-oriented programming language, and in Python everything is an object. This means every entity has some metadata (called attributes) and associated functionality (called methods). These attributes and methods are accessed via the dot syntax. We will come back to this later on. Run the following code fragment:

```
x = 8.5
type(x)
```

Python has **types**; however, the types are linked not to the variable names but to the objects themselves.

# Variables, Expressions, and Statements

Conceptually, *a value* is a basic element of the Python language such as a number or a character. The value has usually a **type** such as **integer, floating point, Boolean, etc.**.

```
print(type(8))
print(type(True))
print(type('Hallo'))
```

*A variable* is a placeholder for a value. Note that unlike in Java, variables need not be declared with a type. Assignments are simply done as follows:

```
n = 3
n = 5.3
print(n)
print(type(n))
```

Variable names can be arbitrarily long containing letters and numbers. They should not start with a number neither contains certain characters such as space. If you choose an illegal name, you will get a syntax error. Also, Python **keywords** cannot be used as variables, see the [RA] book for more details. **As always, do use meaningful variable names in your programming and document what the variable is**.

3. Run and fix the following errors

   (a) `print ('We will have a feast tonight unless you say)`

   (b) `speed = 'This is the current speed' int(speed) # bad call`

   (c) `bad name = 5`

4. Use the following formula to convert temperatures from degrees Celsius to Fahrenheit.

$$T(^oF) = T(^oC)\frac{9}{5} + 32$$

   Use meaningful variable names and add comments to your code. Test your formula if for the temperatures $20^oC$ and $40^oC$.

   **Hint:** You may use the command input to read input data from the keyboard.

5. It is handy to use Python's interactive help. Carry out the following tasks:

a. Run the command `help('modules')` and observe the output; this may take a while.
b. You can also use `help('input')` to see the documentation for that specific function.
c. You can also use `help('min')` to see the documentation for that function.

You have now learnt how to quickly find the documentation of a function that you know of from within Python.

# Conditional execution

Conditional statements are usually written with If statements:

```
if x%2 == 0:
    print('x is even')
else:
    print('x is odd')
```

After the colon character (:), the following statements block must be **indented**. Python Boolean values are True, False. We have also the logical operators and, or, not. As in Java, conditional statements can be *nested*.

6. Write a code snippet, which, for a given number of worked hours and an hourly rate computes the pay of an employee if the employee earns 1.5 times the hourly rate for hours worked above 40 hours.

# Iteration

As in Java, we can write While and For loops as follows:

- A typical While loop

```
n = 5
while (n > 0):
    print(n)
    n = n-1
print('Done!')
```

- A For loop

```
#range(5) gives the integers 0 to 4.
for i in range(5):
    print(i)
print('Done!')
```

7. Write a program which repeatedly reads numbers until the user enters ”done”. Once ”done” is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake and print an error message and skip to the next number.

```
Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
16 3 5.33333333333
```

**Hint:** Use the command input to read from the keyboard and the try ... except to deal with invalid input values. We will learn a bit more about exception handling in the tutorial.

# Packages

Python comes with lots of packages that ease up programming. Anaconda http://continuum.io/downloads is a popular Python distribution consisting of more than 195 popular Python packages. Python has specialised packages for Predictive analytics, which we will rely upon for the rest of the module:

- The most versatile package widely used in data science is **pandas** http://pandas.pydata.org/index.html. It enables us to read data from files of different formats, perform some basic statistics, and manipulates data such as dealing with missing data points or merging two data sets, etc.
- The package **Numpy** http://www.NumPy.org/ enabling us to perform array operations and various linear algebra operations.
- The package **Matplotlib** http://matplotlib.org enables us to generate 2D plots of high quality.
- The package **Scikit-learn** http://scikit-learn.org/stable/index.html is formed by algorithms and methods to carry out predictive modelling in Python.

To use a package, we need to import it as follows:

```
import math import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
print(math.pi) # pi value
x = np.random.randn(100,100)
y = np.mean(x,0)
plt.plot(y)
plt.show()
```

We can also import specific functions from a package as follows:

```
from numpy import linspace
x = linspace(0, 1, 100) # make 100 points between 0 and 1.
```

8. Plot the function $x^3 - 4$ in the interval $[-4, 4]$.

   **Hint:** Use `linspace` from `numpy` and scatter to plot the given function. Please add legend to your graph.

# Functions

Python has various built-in functions such as max, min, len, type, str, .... Functions are also accessed via libraries. For example, we need to import the Maths library to use functions such as sqrt, log, .... Run the following code fragment:

```
import math
print(max('Hello world'))
print(len('Hello world'))
print(float(32))
print(math.sqrt(2))
```

We can define our own function as well as follows:

```
#defining the function
def print_twice(bruce):
   print(bruce)
   print(bruce) #calling the function

print_twice("Bye")
```

We can then call the function print twice with an argument such as 'Bye' or 10, etc.

9. Write a function fahrenheit (celsius) that converts temperatures from degrees Celsius to Fahrenheit.

$$T(^oF) = T(^oC)\frac{5}{9} + 32$$

Run your function with the input 20.

# Data structures: Strings

A *string* is a sequence of characters starting from the index 0. Strings are immutable objects and there are lots of built-in methods for parsing and manipulating strings in Python.

```
>>> s = 'Monty Python'
>>> s[0]
'M'
>>> s[0:5]
'Monty'
>>> 'o' in s
True
>>> s.lower()
'monty python'
>>> s.find('n')
2
>>> s[0] = 'P' # error since strings are immutable
```

10. Consider the following code snippet:

```
data = 'X-DSPAM-Confidence: 0.8475 '
```

Use the string methods (strip and find) and string slicing to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a floating point number.

# Data structures: Lists

A *list* is a sequence of items. Unlike strings, lists are *mutable*; we can add, remove items in the list. There are various built-in functions to manipulate lists.

```
>>> s = [ 2, 4, 8]
>>> s[0]=10
>>> print(s)
[10, 4, 8]
>>> s.sort()
>>> print(s)
```

```
[4, 8, 10]
>>> print(s.sort())
None
```

Note that most list methods modify the argument and return None. This means that if you assign l = s.sort(), then will be None. So be careful while using Python list methods.

11. Run step by step the following commands and print the results. Note how we can define lists in comprehension.

```
l1 = [1, 9, 7, 3, 5]
l2 = [e for e in range(10) if e%2==1]
print(l1 == l2)
l3 = sorted(l1)
print(l2 == l3) #values testing
print(l2 is l3) #references testing
```

# Variables and Objects

**Python variables are pointers**. If we have two variable names pointing to the same **mutable** object, then they are *aliases* meaning changing one will change the other as well. For example

```
>>> x = [2, 4, 8]
>>> y = x #y is an alias of x
>>> x.append(10)
>>> print(y)
[2, 4, 8, 10]
>>> print(x is y)
True
```

*Numbers, strings, and other simple types are* **immutable***: you can't change their value - you can only change what values the variables point to*.

12. Write a program that prompts the user for a list of numbers and prints out the maximum and minimum of the numbers at the end when the user enters

'done'. Store the numbers the user enters in a list and use the `max()` and `min()` functions to compute the maximum and minimum numbers after the loop completes. Return the sorted list of numbers read in. Handle exceptions appropriately.

```
Enter a number:2
Enter a number:5
Enter a number:7
Enter a number:3
Enter a number:9
Enter a number:4
Maximum: 9.0
Minimum: 2.0
```

# Data Structures: Tuples

A *tuple* is an ordered sequence of elements that may be of different types. Unlike lists, tuples are immutable, this means we cannot change its element values.

13. Run step by step the following commands and print their results.

```
x = 3.4
y = 6.7
y, x = x, y #swap the values of x and y
e = (1, 2, 3) > (0, 3, 10)
txt = 'Python is fun'
words = txt.split()
t = list()
for word in words:
    t.append((word, len(word)))
t.sort(reverse = True)
```

# Python arrays

NumPy (Numerical Python) provides an efficient way to store and manipulate multidimensional dense arrays in Python. To access it we need to import the package numpy.

- It provides an ndarray structure, which allows efficient storage and manipulation of vectors, matrices, andhigher-dimensional datasets.
- It provides a readable and efficient syntax for operating on this data, from simple element-wise arithmeticto more complicated linear algebraic operations.

14. Run step by step the following commands and print the results of the array operations carried out mainly how we can use a mask (logical expression) to select a subarray of a given array. For example we may want to select all the elements of an array that are greater than 2. Anyway, Try to understand all those array operations.

```
import numpy as np
l = [x**2 for x in range(1, 5)]
a = np.array(l)
b = a+2 #element wise operation
c = a*b #element wise multiplication
# arrays must have the same dimensions
d = c/b #element wise division
a2 = a.reshape(2, 2)
amask = a2[a2<5] #array subsetting
```

15. Consider the following string

data = ' From the farm, I saw the first sheep from afar while I was the first farmer in the place. '

Write a function word count that reads in a string text s and returns a list of tuples that stores each word and its number of occurrences in s. This is better implemented as a dictionary and you can return the output as a dictionary if you can.

**Hint:** Use the method str.maketrans(dict.fromkeys(string.punctuation)) to delete all punctuation characters that may be attached to a word. Then, use the string methods strip, translate, split in order to answer this question. As usual, you may

# Regular expressions

Python is a powerful language for text processing thanks to its package on **regular expressions** enabling us to *search* and *extract* patterns through files by using methods such as split and find. See the following links http://en.wikipedia.org/wiki/Regular_expression and https://docs.python.org/2/library/re.html for more details. Remember you do not need to know all the wild characters and you can always look at the documentation when required.

```
import re
```

```
hand = open('msg.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line) : #line beginning with From
        print(line)
```

16. Consider the following code snippet:

```
msg = '@msg: Hallo from cs@le.ac.uk to cse@unilim.fr about the conf.@2PM'
```

Write that extract all valid email addresses in the text msg by using regular expressions.