Python Refresher

Here we provided a basic refresher on Python concepts.

We recommend you run this Jupyter notebook on your own computer to ensure that all packages have been installed correctly. Please refer to the Juypter document on Canvas for a guide on how to install Jupiter Notebook and how to install new packages in the Python environment with Anaconda.

Other resources:

-- For a more detailed review of Python, please refer to other guides, e.g. https://developers.google.com/edu/python?hl=en.

-- Complete OLET1603 - Analysing and Plotting data: Python - see the document on Canvas about how to enroll

-- Recommended books for the Python part of this course:

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

Introduction to Machine Learning with Python

Example 1: Python

Data structures

Python list, dictionary, tuple and set are the most used data structures in data science applications.

In [15]:

```
dictionary = {1: "One", 2: "Two", 3: "Three"}
tuple = (1, 2, 3)
set = {1, 2, 3}

print("list:", list)
print("dictionary:", dictionary)
print("tuple:", tuple)
print("set:", set)
```

```
list: [1, 3, 4]
dictionary: {1: 'One', 2: 'Two', 3: 'Three'}
tuple: (1, 2, 3)
set: {1, 2, 3}
```

Useful Functions

In [16]:

```
#Length of list
list = [1, 2, 3]
print(len(list))
#String concatenation and int to string function
print("hi" + str(len(list)))
#Append/remove in list
list.append(4)
print(list)
list.remove(2)
print(list)
#Count number in list
print(list.count(4))

# The help function can be used on any other function to display its documentation
help(len)
```

```
3
hi3
[1, 2, 3, 4]
[1, 3, 4]
```

```
1
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

Loops and conditions in Python

There are two types of loops in Python, for and while. Python uses indents to indicate blocks of code

In [17]:

```
for i in range(2):
    print(i)
    print(2-i)
```

```
0
2
1
1
```

In [18]:

```
# Conditions if and else are employed to nest conditions
x = 5;
if x%2==0 :
    print ("This is an even number.")
else :
    print ("This is an odd number.")
```

```
This is an odd number.
```

Functions

Python code could be made modular as block of reuseable code, called function. A function can accept parameters and can return values.

In [19]:

```
def hello(name):
    print("Hello " + name)
    return True

hello("World")
```

```
Hello World
```

Out[19]:

```
True
```

Example 2: Numpy Arrays

Numpy arrays are a very common form of arrays to use in python. If Numpy is not found, refer to the Jupyter notebook guide to installing new packages into the anaconda environment.

For more details and examples on Numpy functions refer to https://github.com/ageron/handson-ml/blob/master/tools_numpy.ipynb

In [20]:

```
## Import Numpy library - "as" keyword allows a simple reference to the package
import numpy as np

#1D Zeros array
print(np.zeros(5))
```

```
#2D Zeros array
a = np.zeros([3,4])
print("a:")
print(a)

print("Size:" + str(a.size))
print("Shape:" + str(a.shape))
print("Num Dimensions:" + str(a.ndim))

# Arrays of ones can be made, and multipled by integers
b = 2*np.ones([3,4])
print("b:")
print(b)
# Arrays of the same size can be added together
print("a+b")
print(a+b)


# Numpy range of numbers
print(np.arange(5,10))
# Can also be used with a step parameter
print("c:")
c = np.arange(5,10,0.5)
print(c)



# Elements from an array can be accessed using "[]". Note: Python arrays index from 0
print(c[0])
print(c[1:2])


# Numpy arrays can be reshaped
print("d:")
d = b.reshape(2,6)
print(d)
print("Shape", d.shape)
# ":" can be used to access all elements for a row/column
print(d[:, 1:4])
```

```
[0. 0. 0. 0. 0.]
a:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Size:12
Shape:(3, 4)
Num Dimensions:2
b:
[[2. 2. 2. 2.]
 [2. 2. 2. 2.]
 [2. 2. 2. 2.]]
a+b
[[2. 2. 2. 2.]
 [2. 2. 2. 2.]
 [2. 2. 2. 2.]]
[5 6 7 8 9]
c:
[5.  5.5 6.  6.5 7.  7.5 8.  8.5 9.  9.5]
5.0
[5.5]
d:
[[2. 2. 2. 2. 2. 2.]
 [2. 2. 2. 2. 2. 2.]]
Shape (2, 6)
[[2. 2. 2.]
 [2. 2. 2.]]
```

Example 3: Pandas Dataframe

We will be using pandas dataframe in this course. Pandas dataframe is a two dimensional, potentially heterogeneous tabular data structure. It consists of three main components: data, row and columns.

In [21]:

```
import pandas as pd

# create a simple dictionary of people
data = {'Name': ["John", "Anna", "Peter", "Linda"],
          'Location' : ["New York", "Paris", "Berlin", "London"],
          'Age' : [24, 13, 53, 33]
        }
# Convert dictionary to dataframe
data_pandas = pd.DataFrame(data)
# IPython.display allows "pretty printing" of dataframes
# in the Jupyter notebook
display(data_pandas)

# Specific columns of data can also be selected using "[]"
display(data_pandas[["Name", "Age"]])

# Rows can also be selected by either an integer of the row using the iloc function of pandas
display(data_pandas.iloc[[1]])


# You can also select a row with loc.
#Let's first change the index of this dataset to "Name", and then select the row for Peter.
data_pandas = data_pandas.set_index("Name")
display(data_pandas)
display(data_pandas.loc[["Peter"]])
```

	Name	Location	Age
0	John	New York	24
1	Anna	Paris	13
2	Peter	Berlin	53
3	Linda	London	33

	Name	Age
0	John	24
1	Anna	13
2	Peter	53
3	Linda	33

	Name	Location	Age
1	Anna	Paris	13

Name	Location	Age
John	New York	24
Anna	Paris	13
Peter	Berlin	53
Linda	London	33

Name	Location	Age
Peter	Berlin	53

In [ ]:

```

```