# Python Programming

## Data Analysis with Pandas and Matplotlib

*A. AZYAT*

***This training is inspired from the following site:***

***https://ourcodingclub.github.io/2018/04/18/pandas-python-intro.html***

***Objectives:***

- ✓ Knowing the basic of Pandas data structures,
- ✓ Find out how to access data from a Pandas DataFrame, and how to filter data in a Pandas DataFrame,
- ✓ Learn how to read and sort data from a file;
- ✓ Discover the basics of the Matplotlib for plotting
- ✓ Acquire how to bring together other packages to enhance plots

*Let's learn by example! Before starting I want to inform that this work has been realized with PyCharm where I have installed Pandas, MatPlotLib, Scipy, IPython libraries. For more detail you can have a look on this site: https://www.jetbrains.com/help/pycharm/2016.1/tutorial-using-ipython-jupyter-notebook-with-pycharm.html#d392072e63*

## Knowing the basic Pandas data structures

The **Series** and the **DataFrame** are two core data structures used to store data with Pandas.

### 1. Series

The series is a **one-dimensional** array-like structure designed to hold a single array (or 'column') of data and an associated array of data labels, called an index. We can create a series to experiment with by simply passing a list of data, let's use numbers in this example:

After importing Pandas

Type this code:

```
import pandas as pd
pd.__version__
```

```
Out[4]: '0.23.4'
```

```
my_series = pd.Series([4.5, 3.1, -4.0, 9.0])
print(my_series)
```

The output should be:

```
0    4.5
1    3.1
2   -4.0
3    9.0
dtype: float64
```

If we wanted the values, we can add to our script the following line:

```
print(my_series.values)
```

```
[ 4.5  3.1 -4.   9. ]
```

## 2. DataFrames

The DataFrame represents tabular data. They are organised into colums (each of which is a Series), and each column can store a single data-type, such as floating point numbers, strings, boolean values etc. DataFrames can be indexed by either their row or column names.

We can create a DataFrame in Pandas from a Python dictionary, or by loading in a text file containing tabular data. First we are going to look at how to create one from a dictionary.

Let's first create a dictionary:

```
region_popul={
    'Region':['Oued EdDahab Lagouira','Laayoune Boujdour Sakia 
ElHamra','Guelmim Es Semara','Souss Massa Draa'],
    'Population' :[142955,301744,501921,3601917],
    'Ménages':[29385,67140,98867,747476]
}
dataframe=pd.DataFrame.from_dict(region_popul)
print(dataframe)
```

The output:

```
                     Region                    Population    Ménages
0          Oued EdDahab Lagouira               142955         29385
1  Laayoune Boujdour Sakia ElHamra             301744        67140
2             Guelmim Es Semara                501921        98867
3              Souss Massa Draa                3601917        747476
```

Note how the dictionary keys have become column headers running along the top, and as with the Series, an index number has been automatically generated. The columns are also in the order we specified.
We want to note that Pandas works best with dictionaries when the dictionary keys refer to column names or headers.

## How to access data

Pandas *DataFrames* have many useful methods that can be used to inspect the data and manipulate it. We are going to have a look at just a few of them.

If our *DataFrame* contening a lot of data , we would not want to print all of them to screen, instead we could have a look at the first n items with the head method, which takes the number of rows one want to view as its argument. For example:

```
print(dataframe.head(2))
```

The output should be:

```
        Region                          Population    Ménages
0        Oued EdDahab Lagouira          142955        29385
1  Laayoune Boujdour Sakia ElHamra          301744        67140
```

Using the `tail` method to look at the last `n` rows:

```
print(dataframe.tail(3))
```

Which gives as a result :

```
        Region                                    Population  Ménages
1        Laayoune Boujdour Sakia ElHamra          301744    67140
2            Guelmim Es Semara                    501921    98867
3             Souss Massa Draa                    3601917   747476
```

Our columns in the `dataframe` object are individual Series of data. We can access them by referring to the column name e.g. `dataframe['column-name']`. For instance:

#accessing to a column by its name

```
print("\n",dataframe['Region'])
```

```
The result is:
```

```
0           Oued EdDahab Lagouira
1    Laayoune Boujdour Sakia ElHamra
2              Guelmim Es Semara
3               Souss Massa Draa
Name: Region, dtype: object
```

✓ Note that Pandas DataFrames are accessed *primarily by columns*. In a sense the row is less important to a DataFrame.

For example, the following code *dataframe[0],* generate an error!!!

- ✓ We have to use a different method to get the row, try this: dataframe.iloc[0]
- ✓ The `iloc` method gives us access to the DataFrame in more traditional 'matrix' style notation, i.e. `[row, column]` notation.

| | |
|---|---|
| Write this code | `dataframe.iloc[0,0]`<br>`dataframe.iloc[2,1]`<br>`dataframe. ['Population'][1]` |

- ✓ Give comments on each code line.
- ✓ Try these methods:
    - o dataframe.columns, list(dataframe.columns), dataframe[['Population', 'Ménages']], dataframe.mean(),dataframe.sum(),dataframe.median(), dataframe.describe(),

## How to filter data in a Pandas DataFrame

It possible to apply conditions over the data we are inspecting in order to filter our data. Taking this example :

```
dataframe.Population > 300000
print(dataframe[(dataframe['Population']>50000) &
(dataframe['Ménages']>10000)]
```

Would give:

```
0     False
1      True
2      True
3      True
Name: Population, dtype: bool
```

## How to read data from a file using Pandas

So far we have only created data in Python itself, however Pandas has built in tools for reading data from a variety of external data formats, including Excel spreadsheets, raw text and .csv files. It can even interfacing with databases such as MySQL.
We are going working with the **`population_regions.csv`** file in https://azyat-abdelilah.blogspot.com/p/enseignements.html . We can load this easily into a DataFrame with the `read_csv` function. Create a new file and enter:

```
#working with files using pandas, example: population_regions.csv
import pandas as pd

dataframe=pd.read_csv("population_regions.csv", sep=";", encoding = "ISO-
8859-1")
print(dataframe)
```

Run the script, and you should get the following output:

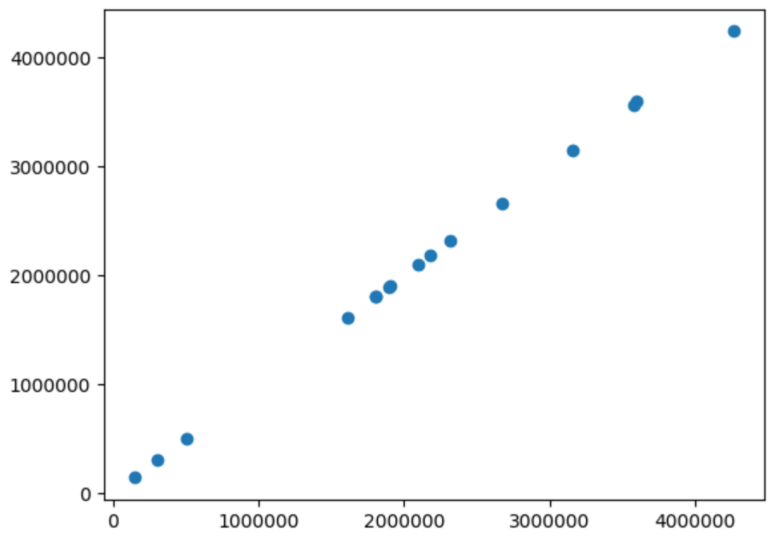
```
                 Region                     Menages   ...   Etrangers  Marocains
0          Oued Ed-Dahab-Lagouira           29385    ...     888         142067
1   Laayoune-Boujdour-Sakia El Hamra        67140    ...     743         301001
```

```
2        Guelmim Es Semara             98867    ...    311      501610
3        Souss-Massa-Draa              747476   ...    5498     3596419
4        Gharb-Chrarda-Beni-Hssen      374753   ...    2002     1902110
5        Chaouia-Ouardigha             388320   ...    2003     1891947
6        Marrakech-Tensift-Al-Haouz    738600   ...    8167     3568476
7        Oriental                      453864   ...    3879     2093750
8        Grand Casablanca              1032576  ...    28161    4242589
9        Rabat-Salé-Zemmour-Zaer       640354   ...    18210    2658544
10       Doukkala-Abda                 444154   ...    1961     2181129
11       Tadla-Azilal                  316475   ...    729      1606777
12       Meknes-Tafilalet              504298   ...    2066     2314799
13       Fès-Boulemane                 401627   ...    3841     1804454
14       Taza-Al Hoceima-Taounate      356119   ...    530      1806506
15       Tanger-Tetouan                719798   ...    7217     3149858

[16 rows x 5 columns]
```

## Plotting data using Matplotlib with Pandas

***Role***: *Matplotlib* is a Python package used for data visualization and plotting. It is a useful complement to Pandas, and like Pandas, is a very feature-rich library which can produce a large variety of plots, charts, maps, and other visualizations.
**Example**: Test the following script; first to work with matplotlib we should import it like this:

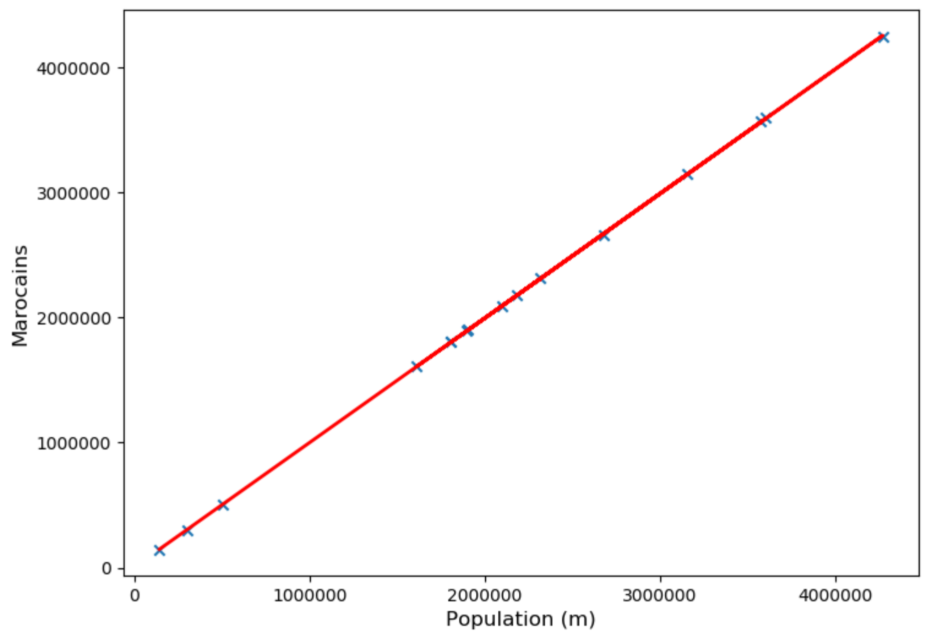
```
import pandas as pd
import matplotlib.pyplot as plt

dataframe=pd.read_csv("population_regions.csv", sep=";", encoding = "ISO-8859-1")

x=dataframe.Population
y=dataframe.Marocains
plt.scatter(x,y)
plt.show()
```



## Bringing together other Python libraries with Matplotlib: scipy

**scipy** is a library that contains a lot of statistics routines. We can import it by adding to the top of our script, and we are going to use it in order to deal with linear regression:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as ss
import numpy

dataframe=pd.read_csv("population_regions.csv", sep=";", encoding = "ISO-
8859-1")

x=dataframe.Population
y=dataframe.Marocains

stats=ss.linregress(x,y)
m=stats.slope
b=stats.intercept

plt.figure(figsize=(6,6))

plt.scatter(x, y, marker='x')

# Set the linewidth on the regression line to 3px
plt.plot(x, m * x + b, color="red", linewidth=2)

# Add x and y lables, and set their font size
plt.xlabel("Population (m)", fontsize=12)
plt.ylabel("Marocains", fontsize=12)

# Set the font size of the number lables on the axes
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()

You will see as a result:
```



**Exercise:**

1- Using data for education 2003 file (or use this link for another example, http://geossc.ma/open-data-partage-de-donnees-geospatiales-en-telechargement-format-geojson/) , read the SCV format,
2- By Pandas, filter all the effective of students more than 40 000,
3- Manipulate data with the following methods: sum, mean, median, describe…
4- Using Matplotlib to build scatter charts of Al Hoceima= f(year), Tetuan = f(year), and for all provinces with different colors,
5- Trace a linear of regression showing labels,