PRACTICE 02 : STEP 1. DATA COLLECTION

1. Aim of the practice

- Discover the principal libraries and functions for data mining and data analysis
- Discover the different data structures with pandas.
- Data manipulation on DataFrame.

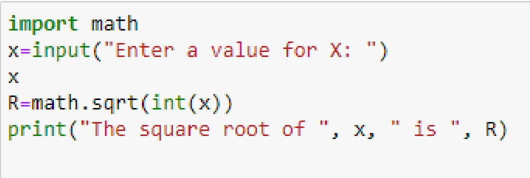
2. Libraries

Python libraries are collections of modules that contain useful codes and functions, eliminating the need to write them from scratch.

To import library, we use the following syntax:

Import name[.subpackage] [**as** variable]

Example:



```
import math
x=input("Enter a value for X: ")
x
R=math.sqrt(int(x))
print("The square root of ", x, " is ", R)
```

- To import some functions, we use the following syntax:

  From library name **Import** function name [**as** variable] or

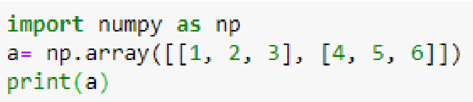
  Import library name. function name [**as** variable]

2.1. NumPy

NumPy is a Python library that provides a multidimensional array object. It facilitates advanced mathematical and other types of operations on large numbers of data. It manipulates the matrix to easily improve machine learning performance.

import numpy as np

Exercise 01: Write the following code



```
import numpy as np
a= np.array([[1, 2, 3], [4, 5, 6]])
print(a)
```

Try with the following expressions: type(a), type([0, 1, 2]), np.shape(a), a[0], a[0,0], a[:, 0], np.amax(a), len(a), np.zeros([5, 3]), b=a[:, 1,2)], c=np.sin(a).

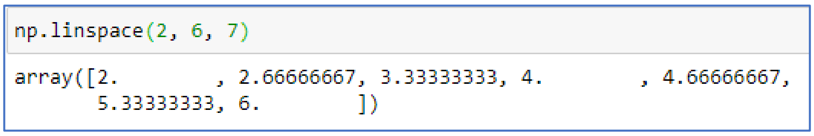
- Show the number of columns of the array “a”

Exercise 02: Linespace () Return evenly spaced numbers over a specified interval.

numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)

Example:

Array with 7 equally spaced samples in the closed interval [2, 6]



```
np.linspace(2, 6, 7)
```

```
array([2.        , 2.66666667, 3.33333333, 4.        , 4.66666667,
       5.33333333, 6.        ])
```

Exercise 1:

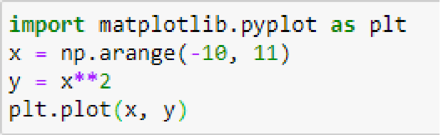
During the session

2.2. Matplotlib

Matplotlib is a Python library focused on data visualization and primarily used for creating beautiful graphs, plots, histograms, and bar charts. It is compatible for plotting data from SciPy, NumPy, and Pandas.

from matplotlib import pyplot as plt or **import matplotlib.pyplot as plt**

Example:



```
import matplotlib.pyplot as plt
x = np.arange(-10, 11)
y = x**2
plt.plot(x, y)
```

Exercise 2:

Use the previous function y=x**2 to show it using:

- The scatter plot “plt.scatter”.
- The bar chart “plt.bar’.

Generate line plot of cosinus and sinus function where the range is between -5 and 5 with 100 elements equally spaced. Use plt.xlabel, plt.ylabel, plt.title and plt.legend(loc='lower right') to show a complete plot.

2.3. Seaborn

Seaborn is another open-source Python library, one that is based on Matplotlib (which focuses on plotting and data visualization) but features Pandas’ data structures.

import seaborn as sns

2.4. Pandas

Pandas (Panel data) is the responsible for preparing high-level data sets for machine learning and training. It relies on two types of data structures, one-dimensional (series) and two-dimensional (DataFrame). **import pandas as pd**

3. Data structures with Pandas

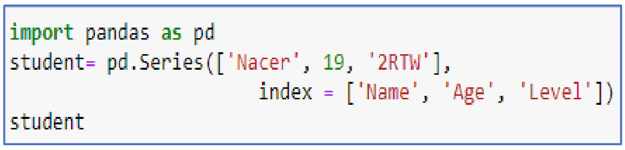
Data Structures are a way of organizing data so that it can be accessed more efficiently depending upon the situation.

3.1. Series

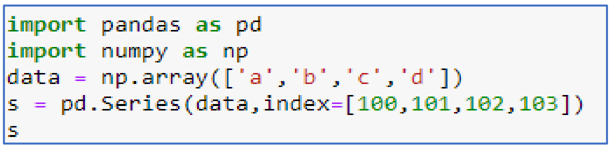
Series can be created by:

- Directly Via S=Pd.Series(Data, Index, Dtype, Name, Copy)
- Passing in a List of values
- Passing in a Dictionary

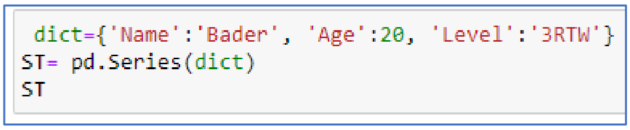
Examples:



```
import pandas as pd
student= pd.Series(['Nacer', 19, '2RTW'],
                   index = ['Name', 'Age', 'Level'])
student
```



```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data,index=[100,101,102,103])
s
```



```
 dict={'Name':'Bader', 'Age':20, 'Level':'3RTW'}
ST= pd.Series(dict)
ST
```

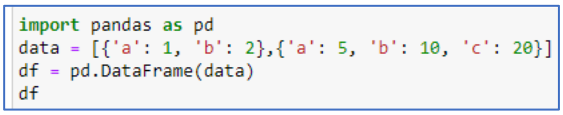
Retrieve a single element using index label value, for example s[101].
Retrieve the first two elements of student: student[:2].
Retrieve the last element of ST: ST[-1:].

3.2. Data frame

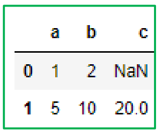
The fundamental Pandas object is called a DataFrame. It is a 2-dimensional size-mutable, potentially heterogeneous, tabular data structure.
A DataFrame can be created by:

- Directly Via: Pandas.Dataframe(Data, Index, Columns, Dtype, Copy)
- Passing in a List of Lists
- Passing in a Dictionary
- Passing in a List of Series
- Reading Data from File, such as CSV, Excel, Json, etc.

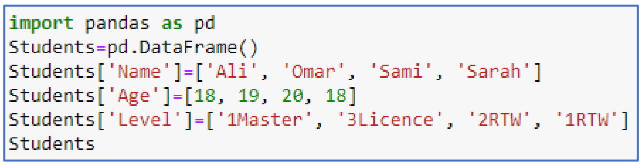
Examples:



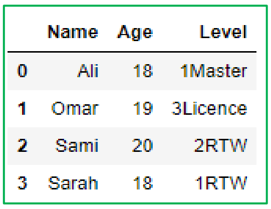
```
import pandas as pd
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
df
```



	a	b	c
0	1	2	NaN
1	5	10	20.0



```
import pandas as pd
Students=pd.DataFrame()
Students['Name']=['Ali', 'Omar', 'Sami', 'Sarah']
Students['Age']=[18, 19, 20, 18]
Students['Level']=['1Master', '3Licence', '2RTW', '1RTW']
Students
```



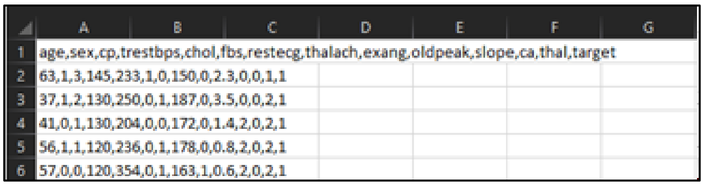
	Name	Age	Level
0	Ali	18	1Master
1	Omar	19	3Licence
2	Sami	20	2RTW
3	Sarah	18	1RTW

Exercise 03:

During the session

Reading Data from a CSV File:

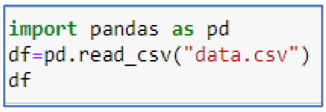
CSV files (Comma Separated Values) store tabular data (numbers and text) in plain text, where each line of the file typically represents one data record.



	A	B	C	D	E	F	G
1	age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,target						
2	63,1,3,145,233,1,0,150,0,2.3,0,0,1,1						
3	37,1,2,130,250,0,1,187,0,3.5,0,0,2,1						
4	41,0,1,130,204,0,0,172,0,1.4,2,0,2,1						
5	56,1,1,120,236,0,1,178,0,0.8,2,0,2,1						
6	57,0,0,120,354,0,1,163,1,0.6,2,0,2,1						

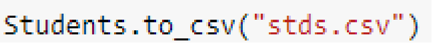
To access data from the CSV file, we require a function read_csv() from Pandas that retrieves data in the form of the data frame.

Examples:



```
import pandas as pd
df=pd.read_csv("data.csv")
df
```

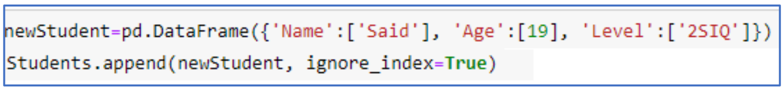
Save dataframe to CSV file:



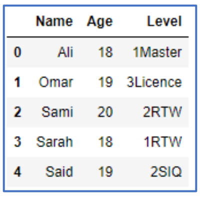
```
Students.to_csv("stds.csv")
```

4. Data Manipulation on Dataframe

- **Adding data in DataFrame using Append Function**



```
newStudent=pd.DataFrame({'Name':['Said'], 'Age':[19], 'Level':['2SIQ']})
Students.append(newStudent, ignore_index=True)
```



	Name	Age	Level
0	Ali	18	1Master
1	Omar	19	3Licence
2	Sami	20	2RTW
3	Sarah	18	1RTW
4	Said	19	2SIQ

- **Columns selection**

Students[['Name', 'Level']]

- **Columns Deletion**

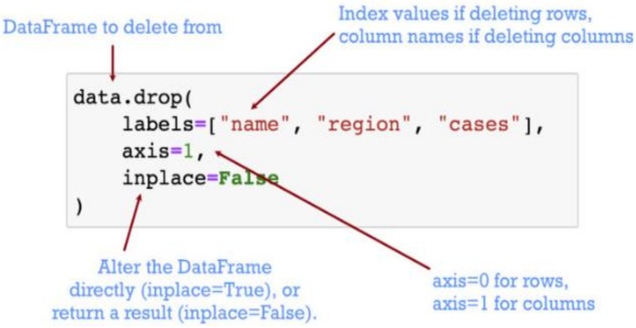
s.drop("Name", axis=1, inplace=True)

- **Rows selection**

lab=s.loc[:,"Age"]

- **Rows Deletion**

df = df.drop(labels=4, axis=0)



DataFrame to delete from

Index values if deleting rows, column names if deleting columns

```
data.drop(
    labels=["name", "region", "cases"],
    axis=1,
    inplace=False
)
```

Alter the DataFrame directly (inplace=True), or return a result (inplace=False).

axis=0 for rows, axis=1 for columns

Exercise 04:

During the session