CLASS 3 PYTHON (NUMPY)

In [ ]:
```
# NUMPY IS A LINEAR ALGEBRA LIBRARY FOR PYTHON
# It is used to generate data
# it is used for demo
# install numpy and is already exist in our juypiter notebook
# pip is used to install application
# numpy has an  allaise called np
```

In [3]:
```
# numpy is also works with Multi dimensional array
import numpy as np
mylist =[1,2,3,4,5]
type(mylist)
```

Out[3]:
```
list
```

In [4]:
```
# We can convert the list into numpy
np.array(mylist)
```

Out[4]:
```
array([1, 2, 3, 4, 5])
```

In [5]:
```
# save the same RESULT inside "X"
# numpy array is similar to list and very fast than python list
x = np.array(mylist)
```

In [6]:
```
x
```

Out[6]:
```
array([1, 2, 3, 4, 5])
```

In [8]:
```
type(x)
```

Out[8]:
```
numpy.ndarray
```

In [9]:
```
# numpy array is similar to list and very fast than python list
data = [[1,2,3], [4,5,6], [7,8,9]]
```

In [10]:
```
data
```

Out[10]:
```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

In [11]:
```
np.array(data)
```

Out[11]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [ ]:
```
# store in variable
```

In [14]:
```
record = np.array(data)
```

In [15]:
```
record
```

Out[15]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [20]:
```
# generating data with numpy
# arange( - generate a range of positive numbers) as long as negative is not specify
# last number not included
np.arange(2, 10)
```

Out[20]:
```
array([2, 3, 4, 5, 6, 7, 8, 9])
```

In [22]:
```
# Generate even number
np.arange(2, 11, 2)
```

Out[22]:
```
array([ 2,  4,  6,  8, 10])
```

In [23]:
```
# How to generate evenly space numbers
# generate 0-10 into six equal part
# last number included
np.linspace(0, 10, 6)
```

Out[23]:
```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

In [24]:
```
# How to generate random numbers
# rand generate floating random number greater than 0 & less than 1
np.random.rand(10)
```

Out[24]:
```
array([0.89950816, 0.94348857, 0.68512268, 0.74987541, 0.75033828,
       0.49151034, 0.47508   , 0.80207813, 0.77836136, 0.27966889])
```

In [28]:
```
np.random.rand(5, 4)
```

Out[28]:
```
array([[0.63227938, 0.6440576 , 0.13199298, 0.54286763],
       [0.95359495, 0.9597785 , 0.51680764, 0.30222487],
       [0.29171494, 0.41332153, 0.23411809, 0.6773691 ],
       [0.51336757, 0.90332818, 0.29602451, 0.31398164],
       [0.54829184, 0.5691589 , 0.61471144, 0.37414261]])
```

In [29]:
```
# randn will generate  random unmbers that can be less than 0 and greater than 1
np.random.randn()
```

Out[29]:
```
0.4014411684761965
```

In [30]:
```
np.random.randn(2, 5)
```

Out[30]:
```
array([[-0.00554147, -0.36803659,  1.14698148,  0.12853885,  1.31999583],
       [ 1.24214605, -1.38861575,  0.9515113 ,  0.74053704, -0.32519788]])
```

In [31]:
```
np.random.randint(4)
```

Out[31]:
```
2
```

In [32]:
```
# start from 1 & END WITH4
np.random.randint(1, 4, 7)
```

Out[32]:
```
array([2, 2, 3, 2, 2, 1, 2])
```

In [34]:
```
np.random.randint(1, 4, 24).reshape(6, 4)
```

Out[34]:
```
array([[2, 2, 2, 2],
       [1, 2, 3, 2],
       [2, 3, 2, 2],
       [3, 3, 2, 1],
       [3, 1, 3, 1],
       [1, 2, 3, 2]])
```

In [36]:
```
# indexing numpy array
# row = 4, while column = 4 according to index number
# alway remeber to add 1 when resolve the row/colum
x = np.arange(1, 26).reshape(5, 5)
```

In [37]:
```
x
```

Out[37]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [39]:
```
# bring out 8 from array
# start with row which is row 1 according to index
# end with column 2
x[1][2]
```

Out[39]:
```
8
```

In [42]:
```
# bring out 24 & 25 from array
# silicing numpy arrany

x[4][3:5]
```

Out[42]:
```
array([24, 25])
```

In [45]:
```
# bring out 12, 13 17 & 18 from array
x[2:4, 1:3 ]
```

Out[45]:
```
array([[12, 13],
       [17, 18]])
```

In [46]:
```
# bring out 6, 7 11 & 12 from array
x[1:3, 0:2]
```

Out[46]:
```
array([[ 6,  7],
       [11, 12]])
```

In [47]:
```
# bring out 16,18,20  & 21,23,25 from array
x[3:5, 0:5:2]
```

Out[47]:
```
array([[16, 18, 20],
       [21, 23, 25]])
```

In [48]:
```
# Arithematic with python list
# this will concatenate
mylist + mylist
```

Out[48]:
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

In [50]:
```
# When sum array together it will pick the number in list and add the same to other li
np.array(mylist) + np.array(mylist)
```

Out[50]:
```
array([ 2,  4,  6,  8, 10])
```

In [51]:
```
#when you have multidimension array
#  Broadcasting duplicate itself
# it must have either same number of row/column
#it must the same dimension either row wise or column wise
np.arange(1, 10).reshape(3, 3) + np.arange(1, 4)
```

Out[51]:
```
array([[ 2,  4,  6],
       [ 5,  7,  9],
       [ 8, 10, 12]])
```

In [52]:
```
np.arange(1, 4)
```

Out[52]:
```
array([1, 2, 3])
```

In [55]:
```
np.arange(1, 10).reshape(3, 3)
```

Out[55]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [56]:
```
np.arange(1, 10).reshape(3, 3) + np.arange(1, 4)
```

Out[56]:
```
array([[ 2,  4,  6],
       [ 5,  7,  9],
       [ 8, 10, 12]])
```

In [ ]:
```

```

In [54]:
```
# add 7 to array
np.arange(1, 10).reshape(3, 3) + 10
```

Out[54]:
```
array([[11, 12, 13],
       [14, 15, 16],
       [17, 18, 19]])
```

In [57]:
```
# use sum
np.arange(1,26).reshape(5,5)
```

Out[57]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [58]:
```
x.sum()
```

Out[58]:
```
325
```

In [60]:
```
# Sum items by column
# AXIS CAN BE EITHER 0 OR 1
# COLUMN = 0
x.sum(axis=0)
```

Out[60]:
```
array([55, 60, 65, 70, 75])
```

In [61]:
```
# sum items by row
# AXIS CAN BE EITHER 0 OR 1
# row = 1
x.sum(axis=1)
```

Out[61]:
```
array([ 15,  40,  65,  90, 115])
```

In [62]:
```
# genereate data 1
np.ones((5, 4))
```

Out[62]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [65]:
```
np.zeros((3, 3))
```

Out[65]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [71]:
```
#If you do not want the random # to be changing
#we can use seed to free the random state of your value
# the seed must has value before we pass will be effect
np.random.randint(1, 4, 24).reshape(6, 4)
```

Out[71]:
```
array([[3, 2, 2, 3],
       [2, 2, 3, 2],
       [3, 1, 3, 2],
       [3, 1, 1, 1],
       [3, 1, 2, 3],
       [1, 3, 2, 3]])
```

In [68]:
```
np.random.seed(5)
y = np.random.randint(1, 4, 24).reshape(6, 4)
```

In [69]:
```
y
```

Out[69]:
```
array([[3, 2, 3, 3],
       [1, 2, 1, 1],
       [3, 1, 3, 1],
       [1, 2, 2, 1],
       [1, 2, 2, 3],
       [1, 3, 2, 3]])
```

In [70]:
```
y
```

Out[70]:
```
array([[3, 2, 3, 3],
       [1, 2, 1, 1],
       [3, 1, 3, 1],
       [1, 2, 2, 1],
       [1, 2, 2, 3],
       [1, 3, 2, 3]])
```

In [80]:
```
# comparison operator in Numpy
x > 10
```

Out[80]:
```
array([[False, False, False, False, False],
       [False, False, False, False, False],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

In [ ]:
```
# Pandas - python library for data analysis

# Series & dataframe
# series - single column
# dataframe - multiple columns
# S MUST BE CAPITAL LETTER IN SERIES
```

In [82]:
```
sample = np.arange(1, 11)
sample
```

Out[82]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [83]:
```
import pandas as pd
```

In [85]:
```
sample = pd.Series(sample)
```

Out[85]:
```
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
dtype: int32
```

In [86]:
```
sample[8]
```

Out[86]:
```
9
```

In [87]:
```
#we can use the data gneratae to  staff income
df = np.random.randint(100,500,12).reshape(4,3)
df
```

Out[87]:
```
array([[469, 345, 235],
       [456, 464, 209],
       [258, 290, 367],
       [423, 480, 283]])
```

In [88]:
```
# convert the data to dataframe
# data is in built, D & F
# store it variable
pd.DataFrame(data=df)
```

Out[88]:

	0	1	2
0	469	345	235
1	456	464	209
2	258	290	367
3	423	480	283

In [93]:
```
pd.DataFrame(data=df, columns=["Jan", "Feb", "Mar"], index=["RBC", "CIBC", "TD", "BMO"
```

Out[93]:

	Jan	Feb	Mar
RBC	469	345	235
CIBC	456	464	209
TD	258	290	367
BMO	423	480	283

In [95]:
```
months = ["Jan", "Feb", "Mar"]
companies = ["RBC", "CIBC", "TD", "BMO"]
pd.DataFrame(data=df, columns= months, index=companies)
```

Out[95]:

	Jan	Feb	Mar
RBC	469	345	235
CIBC	456	464	209
TD	258	290	367
BMO	423	480	283

In [96]:
```
#save the df in csv file
df = pd.DataFrame(data=df, columns= months, index=companies)
```

In [97]:
```
df.to_csv("dunni.csv")
```

In [98]:
```
# import data in pandas
# HTML IS WEB
# STORE IN ANY VARIABLE
df = pd.read_html("https://en.wikipedia.org/wiki/The_World%27s_Billionaires")
```

In [100…
```
df[2]
```

Out[100]:

	No.	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX
2	3	Jeff Bezos	$114 billion	59	United States	Amazon
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway
5	6	Bill Gates	$104 billion	67	United States	Microsoft
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft

In [102…

```
# EXPLONATORY DATA ANALYSIS EDA
# CHECK ROWs & COLUMNS
#check info and missing value
df[2].shape
```

Out[102]:

```
(10, 6)
```

In [104…

```
df = df[2]
```

In [105…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   No.                          10 non-null     int64 
 1   Name                         10 non-null     object
 2   Net worth (USD)              10 non-null     object
 3   Age                          10 non-null     int64 
 4   Nationality                  10 non-null     object
 5   Primary source(s) of wealth  10 non-null     object
dtypes: int64(2), object(4)
memory usage: 608.0+ bytes
```

In [106…

```
df.describe()
```

Out[106]:

	No.	Age
count	10.00000	10.000000
mean	5.50000	71.700000
std	3.02765	12.247902
min	1.00000	51.000000
25%	3.25000	65.500000
50%	5.50000	70.500000
75%	7.75000	80.250000
max	10.00000	92.000000

In [110…

```
# accessing single column
df["Name"]
```

Out[110]:

```
0    Bernard Arnault & family
1                   Elon Musk
2                  Jeff Bezos
3               Larry Ellison
4              Warren Buffett
5                  Bill Gates
6            Michael Bloomberg
7         Carlos Slim & family
8                Mukesh Ambani
9                Steve Ballmer
Name: Name, dtype: object
```

In [112…

```
# accessing multiple column
df[["Name", "Age"]]
```

Out[112]:

	Name	Age
0	Bernard Arnault & family	74
1	Elon Musk	51
2	Jeff Bezos	59
3	Larry Ellison	78
4	Warren Buffett	92
5	Bill Gates	67
6	Michael Bloomberg	81
7	Carlos Slim & family	83
8	Mukesh Ambani	65
9	Steve Ballmer	67

In [113…

```
# REMOVING A COLUMN
# pass command into variable to make it permanent
df.drop("Age", axis=1)
```

Out[113]:

	No.	Name	Net worth (USD)	Nationality	Primary source(s) of wealth
0	1	Bernard Arnault & family	$211 billion	France	LVMH
1	2	Elon Musk	$180 billion	United States	Tesla, SpaceX
2	3	Jeff Bezos	$114 billion	United States	Amazon
3	4	Larry Ellison	$107 billion	United States	Oracle Corporation
4	5	Warren Buffett	$106 billion	United States	Berkshire Hathaway
5	6	Bill Gates	$104 billion	United States	Microsoft
6	7	Michael Bloomberg	$94.5 billion	United States	Bloomberg L.P.
7	8	Carlos Slim & family	$93 billion	Mexico	Telmex, América Móvil, Grupo Carso
8	9	Mukesh Ambani	$83.4 billion	India	Reliance Industries
9	10	Steve Ballmer	$80.7 billion	United States	Microsoft

In [115…

```
# REMOVING multiple COLUMN
# pass command into variable to make it permanent
df.drop(["Age", "Nationality"], axis=1)
```

Out[115]:

	No.	Name	Net worth (USD)	Primary source(s) of wealth
0	1	Bernard Arnault & family	$211 billion	LVMH
1	2	Elon Musk	$180 billion	Tesla, SpaceX
2	3	Jeff Bezos	$114 billion	Amazon
3	4	Larry Ellison	$107 billion	Oracle Corporation
4	5	Warren Buffett	$106 billion	Berkshire Hathaway
5	6	Bill Gates	$104 billion	Microsoft
6	7	Michael Bloomberg	$94.5 billion	Bloomberg L.P.
7	8	Carlos Slim & family	$93 billion	Telmex, América Móvil, Grupo Carso
8	9	Mukesh Ambani	$83.4 billion	Reliance Industries
9	10	Steve Ballmer	$80.7 billion	Microsoft

In [116…

```
# add a new  COLUMN
# pass command into variable to make it permanent
df["double Age"] = df["Age"] * 2
```

In [117…

```
df
```

Out[117]:

	No.	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134

In [118…

```
#renaming coulmns
df.rename(columns={"No.": "S/N"})
df = df.rename(columns={"No.": "S/N"})
```

In [119…

```
df
```

Out[119]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134

In [122…

```
# Accessing rows of elons
# LOC is locate
# elon row is 1
# start from one to one
#last one included
df.loc[1:1]
```

Out[122]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102

In [123…

```
#last one would not be included which 2
df.iloc[1:2]
```

Out[123]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102

In [124…

```
# accessing more than a row
df.iloc[1:5:2]
```

Out[124]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156

In [125…
```
# remove a row
df.drop(1, axis=0)
```

Out[125]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134

In [126…
```
#add a new row
# if want the S/N to continue add
df.append(df)
```

```
C:\Users\samuel\AppData\Local\Temp\ipykernel_2104\3000466870.py:2: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  df.append(df)
```

Out[126]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134

In [127…

```
#add a new row
# if want the S/N to continue from 1
df.append(df, ignore_index=True)
```

```
C:\Users\samuel\AppData\Local\Temp\ipykernel_2104\4228313633.py:3: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  df.append(df, ignore_index=True)
```

Out[127]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134
10	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
11	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
12	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
13	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
14	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
15	6	Bill Gates	$104 billion	67	United States	Microsoft	134
16	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
17	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
18	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
19	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134

In [128…
```
# sorting data
df.sort_values(by="Age")
```

Out[128]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184

In [129…
```
# sorting data by ascending
df.sort_values(by="Age", ascending=False)
```

Out[129]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
4	5	Warren Buffett	$106 billion	92	United States	Berkshire Hathaway	184
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
6	7	Michael Bloomberg	$94.5 billion	81	United States	Bloomberg L.P.	162
3	4	Larry Ellison	$107 billion	78	United States	Oracle Corporation	156
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
5	6	Bill Gates	$104 billion	67	United States	Microsoft	134
9	10	Steve Ballmer	$80.7 billion	67	United States	Microsoft	134
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130
2	3	Jeff Bezos	$114 billion	59	United States	Amazon	118
1	2	Elon Musk	$180 billion	51	United States	Tesla, SpaceX	102

In [130…

```
# FILTERING OF DATA
df["Age"] > 60
```

Out[130]:

```
0     True
1    False
2    False
3     True
4     True
5     True
6     True
7     True
8     True
9     True
Name: Age, dtype: bool
```

In [133…

```
df["Nationality"] != "United States"
```

Out[133]:

```
0     True
1    False
2    False
3    False
4    False
5    False
6    False
7     True
8     True
9    False
Name: Nationality, dtype: bool
```

In [134…
```
# To see data not onlt true /false alone
# FIND NATIONALITY THAT IS OF USA EXTARCTION
df[df["Nationality"] != "United States"]
```

Out[134]:

	S/N	Name	Net worth (USD)	Age	Nationality	Primary source(s) of wealth	double Age
0	1	Bernard Arnault & family	$211 billion	74	France	LVMH	148
7	8	Carlos Slim & family	$93 billion	83	Mexico	Telmex, América Móvil, Grupo Carso	166
8	9	Mukesh Ambani	$83.4 billion	65	India	Reliance Industries	130

In [ ]:
```
# To see data not onlt true /false alone
# `
df[df["Nationality"] != "United States"]
```