In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
data = [1,2,3,4]
series1 = pd.Series(data)
series1
```

Out[3]:

```
0    1
1    2
2    3
3    4
dtype: int64
```

In [4]:

```
type(series1)
```

Out[4]:

```
pandas.core.series.Series
```

In [5]:

```
series1.shape  # shape of matrix
```

Out[5]:

```
(4,)
```

In [6]:

```
syed = pd.DataFrame(['Syed', 'Afzal','John'], ['50000','60000','70000'], columns=(['Salary, Name']
))
syed
```

Out[6]:

	Salary, Name
50000	Syed
60000	Afzal
70000	John

In [13]:

```
dab = pd.read_csv('C:/Users/Syed Afzal/Downloads/DABUR.csv')
dab.head(5) # to get first five rows
```

Out[13]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	9/3/2019	447.950012	452.299988	439.399994	443.049988	443.049988	2282758
1	9/4/2019	443.549988	446.799988	433.950012	442.200012	442.200012	2053458
2	9/5/2019	444.450012	445.649994	437.100006	438.950012	438.950012	887639
3	9/6/2019	440.450012	442.649994	437.149994	441.299988	441.299988	500428
4	9/9/2019	441.000000	447.200012	438.000000	444.250000	444.250000	1327946

In [18]:

```
dab.set_index('Date', inplace=True) # set Date as index
```

In [84]:

```
dab.shape # Shape of the dataframe
```

Out[84]:

```
(22, 7)
```

In [85]:

```
dab.tail() # to get last five rows
```

Out[85]:

	Date	Open	High	Low	Close	Adj Close	Volume
17	2019-09-27	446.200012	448.899994	442.000000	447.100006	447.100006	1470958
18	2019-09-30	446.000000	449.250000	442.049988	447.250000	447.250000	1130716
19	2019-10-01	446.000000	451.500000	438.000000	439.649994	439.649994	1043636
20	2019-10-03	436.100006	438.950012	431.649994	433.649994	433.649994	1637904
21	2019-10-04	434.750000	436.049988	424.250000	425.700012	425.700012	2334293

In [21]:

```
dab[2:5]  # print index value starting from 2 and ending at 4, excluding 5
```

In [22]:

```
dab = dab[['Date', 'Open', 'High', 'Low']]  # print only those columns which are mentioned in the l
ist
```

In [92]:

```
dab.Open.max() # Max value among open colums
```

Out[92]:

```
462.85000599999995
```

In [93]:

```
dab.Close.max() # max value of close column
```

Out[93]:

```
460.0
```

In [94]:

```
dab.Close.std() # Stadndard deviation
```

Out[94]:

```
7.860271905765296
```

In [454]:

```
dab.reset_index(inplace=True)
```

Different Ways of creating a DataFrame

In [398]:

```
emp = pd.DataFrame({
        'Name' : ['Syed', 'Afzal'],
        'Salary': [60000, 70000],
        'Desig' : ['Programmer', 'DS']

})

emp

emp.set_index('Name', inplace=True)

emp
```

Out[398]:

Name	Salary	Desig
Syed	60000	Programmer
Afzal	70000	DS

In [136]:

```
#1 Using list of lists
data = [['tom', '10'], ['nick', '15'], ['juli', 14]]

df = pd.DataFrame(data, columns=['Name', 'Age'])
df
```

Out[136]:

	Name	Age
0	tom	10
1	nick	15
2	juli	14

In [141]:

```
#2 By using dictionary or lists

data = {'Name':['Tom', 'Nick', 'Krish','Jack'], 'Age':[20,21,22,19], 'Job':['C','Py','DS','ML']}

sf = pd.DataFrame(data, index=['Emp1', 'Emp2', 'Emp3', 'Emp4'])
sf
```

Out[141]:

	Name	Age	Job
Emp1	Tom	20	C
Emp2	Nick	21	Py
Emp3	Krish	22	DS
Emp4	Jack	19	ML

In [143]:

```
data = pd.DataFrame({'Name':['Tom', 'Nick', 'Krish','Jack'], 'Age':[20,21,22,19], 'Job':['C','Py','
DS','ML']}, index=['Emp1',

p2', 'Emp3', 'Emp4'])
data
```

```
data
```

Out[143]:

	Name	Age	Job
Emp1	Tom	20	C
Emp2	Nick	21	Py
Emp3	Krish	22	DS
Emp4	Jack	19	ML

In [162]:

```
data = pd.DataFrame([{'A':10, 'B':20}, {'A':30, 'B':40}], columns=['A', 'B'], index=['First', 'Seco
nd'])
data1 = pd.DataFrame(data)

print(data, '\n')

print(data1)
```

```
         A   B
First   10  20
Second  30  40

         A   B
First   10  20
Second  30  40
```

In [392]:

```
# Using tuples

empl = pd.DataFrame(
            [

        ('Syed', 60000,'Data Scientist'),
        ('Afzal',70000,'Programmer')
    ], columns=['Name', 'Salary', 'Designation']

)

empl
```

Out[392]:

	Name	Salary	Designation
0	Syed	60000	Data Scientist
1	Afzal	70000	Programmer

In [393]:

```
# Using list of dictionaries

emplo = pd.DataFrame([
                      {'Name':'Syed', 'Salary': 60000, 'Job':'Data Scientist'},
                      {'Name':'Afzal', 'Salary': 70000, 'Job': 'Programmer'}

])

#emplo
```

In [410]:

```
df = emplo.merge(emp, on='Name', suffixes=[' left', ' right'], indicator=True)
df.set_index('Job', inplace=True)
df
```

Out[410]:

Job	Name	Salary left	Salary right	Desig	_merge
Data Scientist	Syed	60000	60000	Programmer	both
Programmer	Afzal	70000	70000	DS	both

In [411]:

```
X = pd.read_csv('C:/Users/Syed Afzal/Downloads/DABUR.csv')
#X.head()
```

In [226]:

```
Y = pd.read_csv('C:/Users/Syed Afzal/Downloads/data.csv', header=0, skiprows=22, nrows=10,
                names=['Date', 'Sym', 'Series', 'Open','High', 'Low', 'LastT','Close','TTQ','TO_lak
h'])
#Y.set_index('Date', inplace=True)
#Y
```

In [248]:

```
Z = pd.read_csv('C:/Users/Syed Afzal/Downloads/Mock.csv', na_values=['n.a', 'not available'])
Z.head(7)
Z.set_index('id', inplace=True)
Z.head()
```

Out[248]:

id	first_name	last_name	email	gender	salary
1	Joy	Mattock	jmattock0@accuweather.com	Female	¥41269.43
2	NaN	NaN	btolan1@icq.com	NaN	£39669.33
3	Jolie	Gounin	jgounin2@yahoo.com	NaN	$44,922.56
4	Debera	Glanester	dglanester3@meetup.com	Female	NaN
5	Calla	Oluwatoyin	coluwatoyin4@google.cn	Female	$41,388.26

In [252]:

```
Z.to_csv('C:/Users/Syed Afzal/Downloads/random.csv')
```

In [277]:

```
E = pd.read_excel('C:/Users/Syed Afzal/Downloads/MOCK_DATA.xlsx')
E.head()
```

Out[277]:

	id	first_name	last_name	email	gender	salary
0	1	Udale	Childes	uchildes0@ox.ac.uk	Male	€47502,39
1	2	Cross	Frisby	cfrisby1@flavors.me	Male	NaN
2	3	Giraud	Avrasin	gavrasin2@amazon.co.uk	NaN	$38080.37
3	4	NaN	NaN	igogarty3@zimbio.com	Female	£39231.01
4	5	Rickert	Gabbatiss	rgabbatiss4@cbsnews.com	Male	NaN

In [285]:

```
# writing 2 dataframes into single excel files with different sheet names
```

```
with pd.ExcelWriter('C:/Users/Syed Afzal/Downloads/new.xlsx') as writer:
    emplo.to_excel(writer,sheet_name='Employees')
    empl.to_excel(writer,sheet_name='Employee')
```

Handling Missing Data

In [455]:

```
E = pd.read_excel('C:/Users/Syed Afzal/Downloads/MOCK_DATA.xlsx')
#E.head(10)
```

In [336]:

```
M = E.fillna(
{
        'first_name': 'None',
        'last_name':'None',
         'email':'None',
         'gender': 'No Data',
        'salary': 0
})
M.set_index('id', inplace=True)
#M.head()
```

In [381]:

```
#F = E.fillna(method='bfill', axis='rows', limit=1)
#F.dropna(thresh=2)
#F.head(20)
```

In [380]:

```
L = X.interpolate(method='linear')
#L.head()
```

In [ ]:

```
```