```
import pandas as pd
import numpy as np
dict1 = {
    "Names" : ["Adeeb", "Kundan", "Arbaaz", "Sujeet", "Saurav"],
    "Marks" : [9.25, 9.10, 9.65, 8.85, 9.55],
    "Gender" : ["Male","Male","Male","Male","Male"],
    "City" : ["Patna","Samastipur","Jasedi","Gorakhpur","Muzaffarpur"]
}
DF = pd.DataFrame(dict1) # displays dict1 in tabular form
print(DF)
```

```
    Names  Marks Gender         City
0   Adeeb   9.25   Male        Patna
1  Kundan   9.10   Male   Samastipur
2  Arbaaz   9.65   Male       Jasedi
3  Sujeet   8.85   Male    Gorakhpur
4  Saurav   9.55   Male  Muzaffarpur
```

```
DF.to_csv("friends.csv") # exports DF in form of .csv file
DF.to_csv("friends-index-false.csv") # file doesn't contains index

# if dataframe is large and only few rows are needed to be displayed
print(DF.head(2)) # top 2 rows
print(DF.tail(2)) # bottom 2 rows
```

```
    Names  Marks Gender        City
0   Adeeb   9.25   Male       Patna
1  Kundan   9.10   Male  Samastipur
    Names  Marks Gender         City
3  Sujeet   8.85   Male    Gorakhpur
4  Saurav   9.55   Male  Muzaffarpur
```

```
# analysing numeric columns
print(DF.describe())
```

```
          Marks
count  5.000000
mean   9.280000
std    0.327109
min    8.850000
25%    9.100000
50%    9.250000
75%    9.550000
max    9.650000
```

```
# SERIES -----> Create, Manipulate, Querry, Delete
arr1=[10,20,30,40,50]
print(pd.Series(arr1)) # converting arr1 to series (default indexing)
order=list("abcde")
print(pd.Series(arr1,index=order)) # converting arr1 to series (manual
indexing)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
# creating series with python disctionary
d={'a':1,'b':2,'c':3,'d':4,'e':5}
D=pd.Series(d)
print(D)
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

```
# modifying index of D
D.index=list("ABCDE")
print(D)
```

```
A    1
B    2
C    3
D    4
E    5
dtype: int64
```

```
# Slicing
print(D[:3])
print(D[2:])
# Appending in series
D=D.append(pd.Series([6,7,8,9]))
print(D)
```

```
A    1
B    2
C    3
dtype: int64
C    3
D    4
E    5
dtype: int64
```

```
A    1
B    2
C    3
D    4
E    5
0    6
1    7
2    8
3    9
dtype: int64
```

```
# Deleting an index from series
D=D.drop('C')
print(D)
```

```
A    1
B    2
D    4
E    5
0    6
1    7
2    8
3    9
dtype: int64
```

```
# Series operations
arr1=[1,2,3,4,5]
arr2=[6,7,8,9,10,11,12]
s1=pd.Series(arr1)
s2=pd.Series(arr2)
print(s1.add(s2)) # adds corresponding elements. If corresponding
element doesn't exist, then NaN is displayed as result.
print(s1.sub(s2)) # subtract corresponding elements.
print(s1.mul(s2)) # multiply corresponding elements.
print(s1.div(s2)) # divides corresponding elements.
```

```
0     7.0
1     9.0
2    11.0
3    13.0
4    15.0
5     NaN
6     NaN
dtype: float64
0   -5.0
1   -5.0
2   -5.0
3   -5.0
4   -5.0
5    NaN
6    NaN
```

```
dtype: float64
0     6.0
1    14.0
2    24.0
3    36.0
4    50.0
5     NaN
6     NaN
dtype: float64
0    0.166667
1    0.285714
2    0.375000
3    0.444444
4    0.500000
5         NaN
6         NaN
dtype: float64
```

```
print("Median: ", s1.median(), s2.median()) # median
print("Maximum: ", s1.max(), s2.max()) # minimum
print("Minumum: ", s1.min(), s2.min()) # maximum
```

```
Median:  3.0 9.0
Maximum:  5 12
Minumum:  1 6
```

## CREATING DATAFRAMES

```
dates=pd.date_range('today',periods=6) # range of dates from today to
(today+6)
num_arr=np.random.randn(6,4) # matrix of random numbers of size 6 x 4
cols=list("ABCD") # list of letters
df1=pd.DataFrame(num_arr,index=dates,columns=cols) # creating
dataframe
print(df1)
```

```
                                   A         B         C         D
2022-07-05 14:36:15.377793  0.593193  1.240269 -0.208822 -0.200679
2022-07-06 14:36:15.377793 -0.912069  0.250535  1.372377  0.223373
2022-07-07 14:36:15.377793 -0.633513  1.031650  0.882148 -1.362749
2022-07-08 14:36:15.377793 -0.290228  1.796676  0.184126  0.072104
2022-07-09 14:36:15.377793 -0.024622  0.459481 -1.088387 -2.016159
2022-07-10 14:36:15.377793  0.041543 -0.949165 -0.580626  0.373499
```

```
print(df1.dtypes) # data types of objects present in fd1
print(df1.head(2)) # top 2 entries in df1
print(df1.tail(2)) # bottom 2 entries in fd1
```

```
A    float64
B    float64
C    float64
D    float64
```

```
dtype: object
                                   A         B         C         D
2022-07-05 14:36:15.377793  0.593193  1.240269 -0.208822 -0.200679
2022-07-06 14:36:15.377793 -0.912069  0.250535  1.372377  0.223373
                                   A         B         C         D
2022-07-09 14:36:15.377793 -0.024622  0.459481 -1.088387 -2.016159
2022-07-10 14:36:15.377793  0.041543 -0.949165 -0.580626  0.373499
```

```
print(df1.index, df1.columns, df1.values) # attributes of DataFrame
function which was used to create df1
```

```
DatetimeIndex(['2022-07-05 14:36:15.377793', '2022-07-06
14:36:15.377793',
               '2022-07-07 14:36:15.377793', '2022-07-08
14:36:15.377793',
               '2022-07-09 14:36:15.377793', '2022-07-10
14:36:15.377793'],
              dtype='datetime64[ns]', freq='D') Index(['A', 'B', 'C',
'D'], dtype='object') [[ 0.59319301  1.24026881 -0.2088224  -
0.20067868]
 [-0.91206878  0.25053455  1.37237718  0.22337309]
 [-0.633513    1.0316501   0.88214769 -1.36274875]
 [-0.29022834  1.79667601  0.18412582  0.07210363]
 [-0.02462198  0.45948103 -1.08838699 -2.0161593 ]
 [ 0.04154273 -0.94916484 -0.5806257   0.37349942]]
```

```
print(df1.T) # transpose of orignial DataFrame df1
```

```
   2022-07-05 14:36:15.377793  2022-07-06 14:36:15.377793  \
A                    0.593193                   -0.912069
B                    1.240269                    0.250535
C                   -0.208822                    1.372377
D                   -0.200679                    0.223373

   2022-07-07 14:36:15.377793  2022-07-08 14:36:15.377793  \
A                   -0.633513                   -0.290228
B                    1.031650                    1.796676
C                    0.882148                    0.184126
D                   -1.362749                    0.072104

   2022-07-09 14:36:15.377793  2022-07-10 14:36:15.377793
A                   -0.024622                    0.041543
B                    0.459481                   -0.949165
C                   -1.088387                   -0.580626
D                   -2.016159                    0.373499
```

```
print(df1.describe())
```

```
              A         B         C         D
count  6.000000  6.000000  6.000000  6.000000
mean  -0.204283  0.638241  0.093469 -0.485102
```

```
std    0.533153  0.955124  0.917432  0.974157
min   -0.912069 -0.949165 -1.088387 -2.016159
25%   -0.547692  0.302771 -0.487675 -1.072231
50%   -0.157425  0.745566 -0.012348 -0.064288
75%    0.025002  1.188114  0.707642  0.185556
max    0.593193  1.796676  1.372377  0.373499
```

```
# creating new dictionary
data = {"animal" :
["cat","cat","snake","dog","dog","cat","snake","cat","dog","dog"],
      "age" : [2.5,3,0.5,np.nan,5,2,4.5,np.nan,7,3],
      "visits" : [1,3,2,3,2,3,1,1,2,1],
      "priority" :
["yes","yes","no","yes","no","no","no","yes","no","no"]
}
labels = list("abcdefghij")
df2 = pd.DataFrame(data,index=labels)
print(df2)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  2.0       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

```
print(df2.sort_values(by="age")) # sorting on the basis of age
```

```
  animal  age  visits priority
c  snake  0.5       2       no
f    cat  2.0       3       no
a    cat  2.5       1      yes
b    cat  3.0       3      yes
j    dog  3.0       1       no
g  snake  4.5       1       no
e    dog  5.0       2       no
i    dog  7.0       2       no
d    dog  NaN       3      yes
h    cat  NaN       1      yes
```

```
print(df2[1:3]) # slicing
```

```
  animal  age  visits priority
b    cat  3.0       3      yes
c  snake  0.5       2       no
```

```
print(df2[["age","visits"]]) # query dataframe by tag
```

```
   age  visits
a  2.5       1
b  3.0       3
c  0.5       2
d  NaN       3
e  5.0       2
f  2.0       3
g  4.5       1
h  NaN       1
i  7.0       2
j  3.0       1
```

```
df3=df2.copy() # copying dataframe
print(df3)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  2.0       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

```
print(df3.isnull()) # checking quantities which are null
```

```
   animal    age  visits  priority
a   False  False   False     False
b   False  False   False     False
c   False  False   False     False
d   False   True   False     False
e   False  False   False     False
f   False  False   False     False
g   False  False   False     False
h   False   True   False     False
i   False  False   False     False
j   False  False   False     False
```

```
df3.loc["f","age"]=1.55 # locating and changing values ---->
df3.loc[row_name,col_name] = new_value
print(df3)
```

```
  animal   age  visits priority
a    cat  2.50       1      yes
b    cat  3.00       3      yes
c  snake  0.50       2       no
d    dog   NaN       3      yes
e    dog  5.00       2       no
```

```
f    cat  1.55       3       no
g  snake  4.50       1       no
h    cat   NaN       1      yes
i    dog  7.00       2       no
j    dog  3.00       1       no
```

```
print(df3["age"].mean()) # print average of age
print(df3["visits"].mean()) # print average of visits
print(df3["age"].sum()) # sum of ages
print(df3["visits"].sum()) # sum of visits
print(df3["age"].max()) # maximum of ages
print(df3["visits"].max()) # maximum of visits
print(df3["age"].min()) # minimum of ages
print(df3["visits"].min()) # minimum of visits
print(df3.sum()) # sum of every attribute
print(df3.max()) # maximum of every attribute
print(df3.min()) # minimum of every attribute
```

```
3.38125
1.9
27.05
19
7.0
3
0.5
1
animal      catcatsnakedogdogcatsnakecatdogdog
age                                      27.05
visits                                      19
priority              yesyesnoyesnonononoyesnono
dtype: object
animal      snake
age           7.0
visits          3
priority      yes
dtype: object
animal      cat
age         0.5
visits        1
priority     no
dtype: object
```

```
string =
pd.Series(['A','C','D','Aaa','BaCa',np.nan,'CBA','cow','owl'])
print(string.str.lower()) # changing case to lower
print(string.str.upper()) # changing case to upper
print(string)
```

```
0       a
1       c
2       d
```

```
3     aaa
4    baca
5     NaN
6     cba
7     cow
8     owl
dtype: object
0       A
1       C
2       D
3     AAA
4    BACA
5     NaN
6     CBA
7     COW
8     OWL
dtype: object
0       A
1       C
2       D
3     Aaa
4    BaCa
5     NaN
6     CBA
7     cow
8     owl
dtype: object
```

**OPERATIONS FOR DataFrame MISSING VALUES**

```
df4=df3.copy()
print(df4.fillna(df4["age"].mean())) # fill every NULL cell with
average value of age
```

```
  animal      age  visits priority
a    cat  2.50000       1      yes
b    cat  3.00000       3      yes
c  snake  0.50000       2       no
d    dog  3.38125       3      yes
e    dog  5.00000       2       no
f    cat  1.55000       3       no
g  snake  4.50000       1       no
h    cat  3.38125       1      yes
i    dog  7.00000       2       no
j    dog  3.00000       1       no
```

```
print(df4.dropna(how="any")) # deleting data with NULL value
```

```
  animal   age  visits priority
a    cat  2.50       1      yes
b    cat  3.00       3      yes
```

```
c  snake  0.50       2       no
e    dog  5.00       2       no
f    cat  1.55       3       no
g  snake  4.50       1       no
i    dog  7.00       2       no
j    dog  3.00       1       no
```

### DataFrame file operations

```
df3.to_csv("animals.csv")
```

```
df_animals=pd.read_csv("animals.csv")
print(df_animals.head(3))
```

```
  Unnamed: 0 animal  age  visits priority
0          a    cat  2.5       1      yes
1          b    cat  3.0       3      yes
2          c  snake  0.5       2       no
```

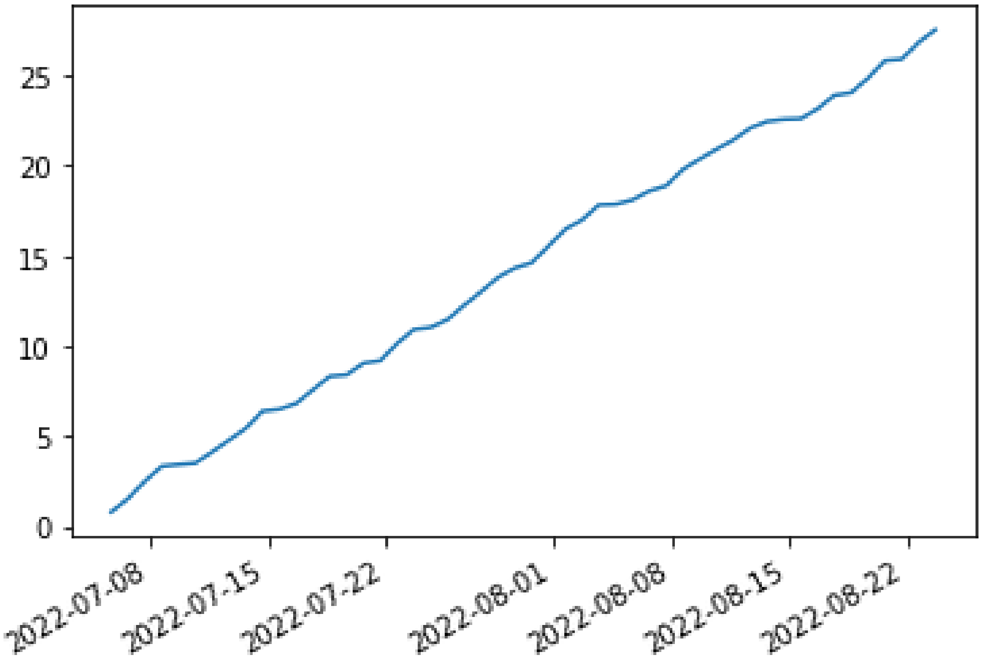
```
df3.to_excel("animals.xlsx", sheet_name="Sheet1")
df_animals2=pd.read_excel("animals.xlsx","Sheet1",index_col=None,na_va
lues=["NA"])
print(df_animals2)
```

```
  Unnamed: 0 animal   age  visits priority
0          a    cat  2.50       1      yes
1          b    cat  3.00       3      yes
2          c  snake  0.50       2       no
3          d    dog   NaN       3      yes
4          e    dog  5.00       2       no
5          f    cat  1.55       3       no
6          g  snake  4.50       1       no
7          h    cat   NaN       1      yes
8          i    dog  7.00       2       no
9          j    dog  3.00       1       no
```

### Visualization in pandas

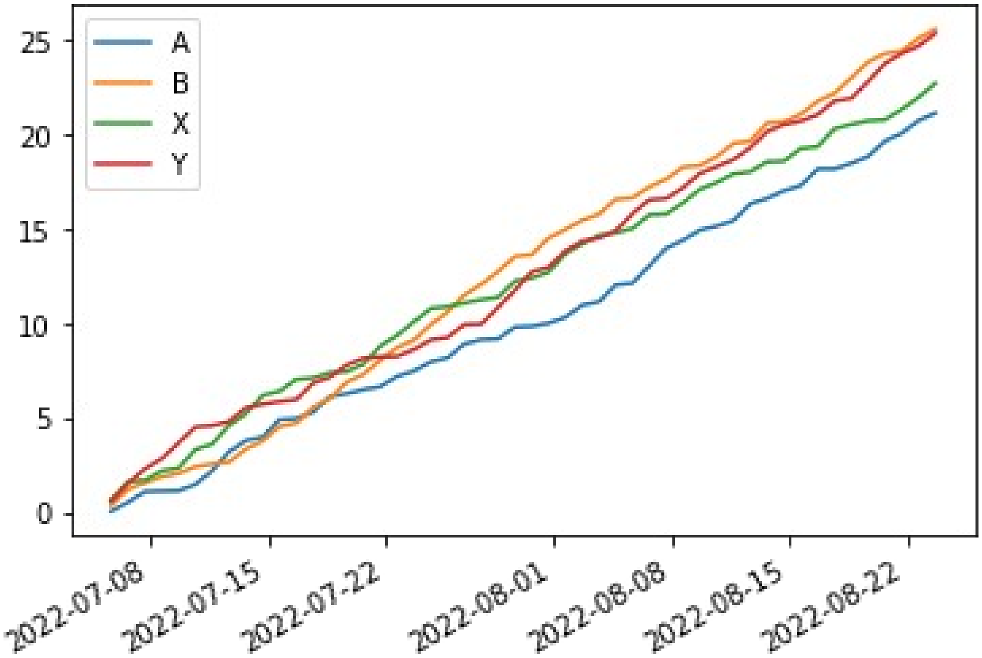
```
%matplotlib inline
ts=pd.Series(np.random.rand(50),
index=pd.date_range('today',periods=50))
ts=ts.cumsum()
ts.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f288f1f7c90>
```



```
ts2=pd.DataFrame(np.random.rand(50,4), index=ts.index,
columns=list("ABXY"))
ts2=ts2.cumsum()
ts2.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f288ebfd690>
```


A
B
X
Y
25
20
15
10
5
0
2022-07-08
2022-07-15
2022-07-22
2022-08-01
2022-08-08
2022-08-15
2022-08-22