Pandas (adv. Python)

Pandas is a data manipulation and analysis tool that is built on Numpy.
Pandas uses a data structure known as DataFrame (think of it as Microsoft excel in Python).
DataFrames empower programmers to store and manipulate data in a tabular fashion (rows and columns).
Series Vs. DataFrame? Series is considered a single column of a DataFrame.

pandas –

1. Series- 1D (sirf 1 column hoga pure data me)
2. dataFrame- 2D (1 ya 1 se jada column hote hain)

Importing pandas

```
import pandas as pd
```

Creating list to series

```
# Let's define a Python list that contains 5 stocks: Nvidia, Microsoft, FaceBook, Amazon, and Boeing
my_list = ['NVDA','MSFT','FB', 'AMZN', 'BA']
my_list
```

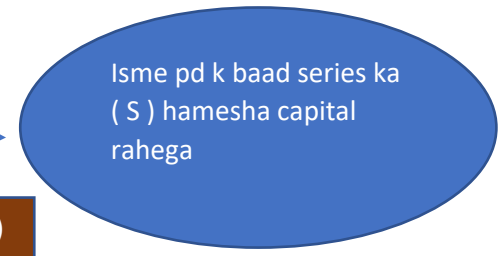
```
['NVDA', 'MSFT', 'FB', 'AMZN', 'BA']
```

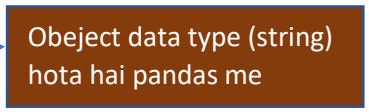
```
# Let's confirm the Datatype
type(my_list)
```

```
list
```

```
series_1 = pd.Series(data = my_list)
series_1
```

```
0    NVDA
1    MSFT
2      FB
3    AMZN
4      BA
dtype: object
```





1. Note that series is formed of data and associated index (numeric index has been automatically generated)
2. Check Pandas Documentation for More information: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html#pandas.Series
3. Object datatype is used for text data (String)

Another way to creating list to series :-

```
# Let's define another Pandas Series that contains numeric values (stock prices) instead of text data
# Note that we have int64 datatype which means it's integer stored in 64 bits in memory
series_2 = pd.Series(data = [100, 200, 500, 1000, 5000])
series_2
```

```
0     100
1     200
2     500
3    1000
4    5000
dtype: int64
```

32 bit operating system hoga to int32 aayaga or 64 bit hoga to int64 aayaga ye os batata hai.

DEFINE A PANDAS SERIES WITH CUSTOM INDEX

```
# Let's define a Python list that contains 5 stocks: Nvidia, Microsoft, FaceBook, Amazon, and Boeing
my_list = ['NVDA','MSFT','FB', 'AMZN', 'BA']
my_list
```

```
['NVDA', 'MSFT', 'FB', 'AMZN', 'BA']
```

```
# Let's define a python list as shown below. This python list will be used for the Series index:
my_labels = ['stock#1', 'stock#2', 'stock#3', 'stock#4', 'stock#5']
my_labels
```

```
['stock#1', 'stock#2', 'stock#3', 'stock#4', 'stock#5']
```

```
series_3 = pd.Series(data = my_list, index = my_labels)
```

```
# Let's view the series
series_3
```

```
stock#1    NVDA
stock#2    MSFT
stock#3      FB
stock#4    AMZN
stock#5      BA
dtype: object
```

```
# Let's obtain the datatype
type(series_3)
```

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

DEFINE A PANDAS SERIES FROM A DICTIONARY

A Dictionary consists of a collection of key-value pairs.
Or key as a indexing kaam karta hai value as a value karat hai kaam.

```
my_dict = {'Bank client ID' : 111,
           'Bank Client Name' : 'Steve',
           'Net worth [$]' : 3500,
           'Years with bank' : 9}
```

```
# Show the dictionary
my_dict
```

```
{'Bank client ID': 111,
 'Bank Client Name': 'Steve',
 'Net worth [$]': 3500,
 'Years with bank': 9}
```

```
# Confirm the dictionary datatype
type(my_dict)
```

```
dict
```

Now converting above dictionary into series

```
# Let's define a Pandas Series Using the dictionary
series_4 = pd.Series(data=my_dict)
series_4
```

```
Bank client ID        111
Bank Client Name    Steve
Net worth [$]        3500
Years with bank         9
dtype: object
```

- **Attributes –** values shape,size {jike piche parenthese () , nahi lagte vo attributes hote hain}
- **Methods-** tail(), head(),etc. {jike piche parenthese () , lagte vo Methods hote hain}
- **Indexers-** [] {core python ki tarah same indexing}

PANDAS ATTRIBUTES

Attributes/Properties: do not use parantheses "()"

- **values**

```
my_list = ['NVDA','MSFT','FB', 'AMZN', 'Google']
my_series = pd.Series(data = my_list)
my_series
```

```
0      NVDA
1      MSFT
2        FB
3      AMZN
4    Google
dtype: object
```

Values output hamesha array me deta hai

```
my_series.values
```

```
array(['NVDA', 'MSFT', 'FB', 'AMZN', 'Google'], dtype=object)
```

- **index**

```
# index is used to return the index (axis labels) of the Series
my_series.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

- **dtype**

```
# dtype is used to return the datatype of the Series ('O' stands for 'object' datatype)
my_series.dtype
```

```
dtype('O')
```

- **is_unique**

```
# Check if all elements are unique or not
my_series.is_unique
```

- ```
  True
  ```
  ```

```
my_series=pd.Series(data=['Google','Microsoft','Amazon','Facebook','Tiktok'])
my_series
```

```
0       Google
1    Microsoft
2       Amazon
3     Facebook
4       Tiktok
dtype: object
```

```
my_series.values
```

```
array(['Google', 'Microsoft', 'Amazon', 'Facebook', 'Tiktok'],
      dtype=object)
```

```
my_series.is_unique
```

```
True
```

```
my_series.size
```

```
5
```

```
my_series.shape
```

```
(5,)
```

```
my_series.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

```
my_series.ndim
```

```
1
```

Head () – bina k 5 rows dikhata hai suru k or iski max limit 60 rows hoti hai

Tail() – bina k 5 rows dikhata hai last k or iski max limit 60 rows hoti hai

```
my_series = pd.Series(data = [100, 200, 500, 1000, 5000,6000])
my_series
```

```
0     100
1     200
2     500
3    1000
4    5000
5    6000
dtype: int64
```

```
my_series.tail(2)
```

```
4    5000
5    6000
dtype: int64
```

```
my_series.memory_usage()
```

```
176
```

IMPORT CSV DATA (1-D) USING PANDAS

Data file used in note → click here

```
# Pandas read_csv is used to read a csv file and store data in a DataFrame by default (DataFrames will be covered shortly!)
# Use Squeeze to convert it into a Pandas Series (One-dimensional)
# Notice that no foramtting exists when a Series is plotted
sp500 = pd.read_csv("C:\\Users\\Uttam Thakur\\Downloads\\S_P500_Prices.csv", squeeze = False)
```

```
sp500
```

	sp500
0	1295.500000
1	1289.089966
2	1293.670044
3	1308.040039
4	1314.500000
...	...
2154	3327.770020
2155	3349.159912
2156	3351.280029
2157	3360.469971
2158	3333.689941

2159 rows × 1 columns

Squeeze= true hoga toh series me datatype aayaga or false hoga toh data frame me datatype aayaga

```
my_series=pd.read_csv('Downloads\S_P500_Prices.csv',squeeze=False)
my_series
```

```
C:\Users\baliy\AppData\Local\Temp\ipykernel_14272\1706403962.py:1: FutureWarning: The squeeze argument has been deprecated and will be removed in a future version. A
pend .squeeze("columns") to the call to squeeze.


  my_series=pd.read_csv('Downloads\S_P500_Prices.csv',squeeze=False)
```

	sp500
0	1295.500000
1	1289.089966
2	1293.670044
3	1308.040039
4	1314.500000
...	...
2154	3327.770020
2155	3349.159912
2156	3351.280029
2157	3360.469971
2158	3333.689941

2159 rows × 1 columns

9. PERFORM MATH OPERATIONS ON PANDAS SERIES

```
# Let's import CSV data as follows:
sp500 = pd.read_csv('S&P500_Prices.csv', squeeze = True)
sp500
```

```
# Apply Sum Method on Pandas Series
sp500.sum()
```

```
# Apply count Method on Pandas Series
sp500.count()
```

```
# Obtain the maximum value
sp500.max()
```

```
# Obtain the minimum value
sp500.min()
```

```
# My favourite: Describe!
# Describe is used to obtain all statistical information in one place
sp500.describe()
```

Shorting pandas series

- **sort_values( )**

```
x=pd.read_csv('C:\\Users\\lnxhunt\\Downloads\\S_P500_Prices (1).csv', squeeze=True)
```

```
C:\Users\lnxhunt\AppData\Local\Temp\ipykernel_1660\2691892639.py:1: FutureWarning: The squeeze argument has been deprecated and
will be removed in a future version. Append .squeeze("columns") to the call to squeeze.


  x=pd.read_csv('C:\\Users\\lnxhunt\\Downloads\\S_P500_Prices (1).csv', squeeze=True)
```

```
x
```

```
0       1295.500000
1       1289.089966
2       1293.670044
3       1308.040039
4       1314.500000
           ...
2154    3327.770020
2155    3349.159912
2156    3351.280029
2157    3360.469971
2158    3333.689941
Name: sp500, Length: 2159, dtype: float64
```

Isme ye assending to descending hain values

Sabse minimum value check karne k liye min function ka use kar sakte hain

```
min(x)
```

```
1278.040039
```

```
max(x)
```

```
3386.149902
```

Sabse maximum value

Agar value descending se ascending me chiye toh :-

```
x.sort_values(ascending= False )
```

```
2037    3386.149902
2035    3380.159912
2033    3379.449951
2034    3373.939941
2038    3373.229980
           ...
2       1293.670044
1       1289.089966
99      1285.500000
98      1278.180054
97      1278.040039
Name: sp500, Length: 2159, dtype: float64
```

Sabse highest value

Sabse lowest value

Isse actual data me change nahi ho raha hai.

Agar actual data me changes karna hai toh

{ inplace = True } command dalna hoga

```
x
```

```
0       1295.500000
1       1289.089966
2       1293.670044
3       1308.040039
4       1314.500000
           ...
2154    3327.770020
2155    3349.159912
2156    3351.280029
2157    3360.469971
2158    3333.689941
Name: sp500, Length: 2159, dtype: float64
```

Original data me change nahi hua hai

Make change in original data

```
x.sort_values(ascending= False,inplace=True)
```

```
x
```

```
2037    3386.149902
2035    3380.159912
2033    3379.449951
2034    3373.939941
2038    3373.229980
           ...
2       1293.670044
1       1289.089966
99      1285.500000
98      1278.180054
97      1278.040039
Name: sp500, Length: 2159, dtype: float64
```

ab original data me change ho gaya hai , { inplace } ki help se

Statical information of data like(std, sum,count,mean ,25%,50%,75)

```
x.describe()
```

```
count    2159.000000
mean     2218.749554
std       537.321727
min      1278.040039
25%      1847.984985
50%      2106.629883
75%      2705.810059
max      3386.149902
Name: sp500, dtype: float64
```

(in) function :-

ye values dekhne k kaam aata hai ki value data me exist karti hai ni nahi

```
1293.670044 in x
```

```
False
```

Ye value exist kar rahi hai fir bhi false ara hai kyu ki ye by default indexing dekhta hai

```
1293.670044 in x.values
```

```
True
```

Agar values check karna hai ki exist karta hai ki nahi{ values } attribute ka use karna hoga

[Dataframe]

1. Meta Data-->info()
2. Statcial information-->describe()
3. find out the null values-->isna().sum,isnull().sum()
4. drop column-->need axis=1
5. add new column
6. rename column
7. Drop null values
8. Fill null values
9. groupby
10. concat
11. Merge
12. chnge the datatype
13. loc,iloc
14. sort_values
15. Set index
16. reset index
17. indexing

```
y=pd.read_csv('C:\\Users\\lnxhunt\\Downloads\\S_P500_Prices (1).csv')
```

```
type(y)
```

```
pandas.core.frame.DataFrame
```

Metadata

```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2159 entries, 0 to 2158
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   sp500   2159 non-null   float64
dtypes: float64(1)
memory usage: 17.0 KB
```

Describe

```
y.describe()
```

	sp500
count	2159.000000
mean	2218.749554
std	537.321727
min	1278.040039
25%	1847.984985
50%	2106.629883
75%	2705.810059
max	3386.149902