Cont. Pandas - I

🎯 Session Objectives:

- ✅ Understand what Pandas is and its importance
- ✅ Install and import the Pandas library
- ✅ Understand data structures in Pandas
- ✅ Understand what a Series is
- ✅ Differentiate Pandas Series vs NumPy Arrays
- ✅ Create Series from scalar, list, array, and dictionary
- ✅ Access Series elements using indexing and slicing
- ✅ Understand attributes of Series
- ✅ Learn basic mathematical operations on Series

```
import numpy as np
import pandas as pd

# Dictionary [Key [Label/Index] : Value[Data]]
_employee_dict = {
    'emp_id' : 'emp101',
    'name' : 'Utkarsh',
    'age' : 29,
    'gender' : 'M',
    'salary' : '$10,00,000',
    'designation' : 'Senior Analyst',
    'email' : 'utk232@gmail.com',
    'state' : 'Delhi',
    'country' : 'India'
}
series = pd.Series(_employee_dict)
series
```

```
emp_id                  emp101
name                   Utkarsh
age                         29
gender                       M
salary              $10,00,000
designation      Senior Analyst
email          utk232@gmail.com
state                     Delhi
country                   India
dtype: object
```

```
# Normal Indexing .iloc[Positional Based Indexing]
print(series['email'])
print(series.iloc[4]) # salary
```

```
utk232@gmail.com
$10,00,000
```

```
# Using .loc [Labelled Based Indexing]
print(series.loc['state']) # index
print(series.loc['name' : 'email']) # slicing
```

```
Delhi
name                    Utkarsh
age                          29
gender                        M
salary                $10,00,000
designation       Senior Analyst
email           utk232@gmail.com
dtype: object
```

```
# .loc[Calling multiple Columns]
print(series.loc[['name' , 'gender', 'email' , 'country']])
```

```
name                 Utkarsh
gender                     M
email      utk232@gmail.com
country               India
dtype: object
```

```
data = [11,22,33,44,55,66,77,88,99]
series = pd.Series(data)
series
```

```
0    11
1    22
2    33
3    44
4    55
5    66
6    77
7    88
8    99
dtype: int64
```

```
series.iloc[0:4] # [11,22,33,44]
```

```
0    11
1    22
2    33
3    44
dtype: int64
```

```
series.iloc[0:8:3] # [11,44,77]
```

```
0    11
3    44
6    77
dtype: int64
```

```
series.loc[0]
```

```
11
```

```
series.loc[0:4] # label bases 4 will be included -> index read
```

```
0    11
1    22
2    33
3    44
4    55
dtype: int64
```

```
data = [11,22,33,44,55,66,77,88,99]
label = ['a','b','c','d','e','f','g','h','i']
series = pd.Series(data , label)
series
```

```
a    11
b    22
c    33
d    44
e    55
f    66
g    77
h    88
i    99
dtype: int64
```

```
series.iloc[0:4] # [11,22,33,44]
```

```
a    11
b    22
c    33
d    44
dtype: int64
```

```
series.loc['a':'e']
```

```
a    11
b    22
c    33
d    44
e    55
dtype: int64
```

```
# Attributes of Series
car_list = np.array(['Taigun','Thar','Magnite','Brezza',
                     'Harrier','Slavia','City','Fortuner','Creta'])
brand_list = np.array(['VW','Mahindra','Nissan','Suzuki',
                       'Tata','Skoda','Honda','Toyota','Hyundai'])
car_model = pd.Series(
    car_list,
    index = brand_list,
    name = 'Car Model 😍'
)
car_model
```

```
VW            Taigun
Mahindra        Thar
Nissan       Magnite
Suzuki        Brezza
Tata         Harrier
Skoda         Slavia
Honda           City
Toyota      Fortuner
Hyundai       Creta
Name: Car Model 😍, dtype: object
```

```
car_model.name
```

```
'Car Model 😍'
```

```
car_model.index # Return the List of 'index' [Brand_list] data
```

```
Index(['VW', 'Mahindra', 'Nissan', 'Suzuki', 'Tata', 'Skoda', 'Honda',
       'Toyota', 'Hyundai'],
      dtype='object')
```

```
car_model.values # Return the data list [Car_name] corresponding to 'index'
```

```
array(['Taigun', 'Thar', 'Magnite', 'Brezza', 'Harrier', 'Slavia', 'City',
       'Fortuner', 'Creta'], dtype=object)
```

```
car_model.dtype # Data Type of the value it stores. Object -> <'O'>
```

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

```
# shape of a car_model
car_model.shape
```

```
(9,)
```

```
# size -> Return the number of Values in a Series -> 9
car_model.size
```

```
9
```

```
# car_model.empty -> Returns True/False
car_model.empty # False as our data doesn't have any missing Value
```

```
False
```

```python
# hasnans -> Return Boolean -> Missing Value <NaN>
car_model.hasnans # False
```

```
False
```

```python
# SELECT <DISTINCT> -> Returns Booean -> And check whether the data consumes all uniques or not
car_model.is_unique
```

```
True
```

```python
# ndim -> 1 [Reprsenting 1D Array or a Series]
car_model.ndim
```

```
1
```

```python
# Basic Mathematical Operations
series = pd.Series([7,9,11,15,21,29,55,77,99])
print(series + 11)
```

```
0     18
1     20
2     22
3     26
4     32
5     40
6     66
7     88
8    110
dtype: int64
```

```python
series = pd.Series([7,9,11,15,21,29,55,77,99])
print(series ** 2)
```

```
0      49
1      81
2     121
3     225
4     441
5     841
6    3025
7    5929
8    9801
dtype: int64
```

```python
series = pd.Series([7,9,11,15,21,29,55,77,99])
print(series * 10)
```

```
0     70
1     90
2    110
3    150
4    210
5    290
6    550
7    770
8    990
dtype: int64
```

```
series = pd.Series([7,9,11,15,21,29,55,77,99])
print(series // 10)
```

```
0    0
1    0
2    1
3    1
4    2
5    2
6    5
7    7
8    9
dtype: int64
```

```
# Students_data -> Name as Index & marks as Values
name = np.array(['Utkarsh','Prabhakar','Sanchita','Lubhani',
                 'Ali','Kushagra','Aditya','Nihal','Surya'])

marks = np.array([95,92,91,93,77,99,81,92,77])
stud_series = pd.Series(marks , name)
stud_series
```

```
Utkarsh      95
Prabhakar    92
Sanchita     91
Lubhani      93
Ali          77
Kushagra     99
Aditya       81
Nihal        92
Surya        77
dtype: int32
```

```
stud_series.index
```

```
Index(['Utkarsh', 'Prabhakar', 'Sanchita', 'Lubhani', 'Ali', 'Kushagra',
       'Aditya', 'Nihal', 'Surya'],
      dtype='object')
```

```
stud_series.values
```

```
array([95, 92, 91, 93, 77, 99, 81, 92, 77])
```

```
# indexing .iloc & .loc
stud_series['Sanchita']
```

```
91
```

```
stud_series[7] # KeyError
```

```
stud_series.iloc[7] # 'Nihal' -> 92 [Positional Based Indexing]
```

```
92
```

```
stud_series.loc['Kushagra']
```

```
99
```

```
stud_series.loc[['Lubhani','Ali','Aditya']]
```

```
Lubhani    93
Ali        77
Aditya     81
dtype: int32
```

```
# Limit -> Order BY [Sorting] ['SQL']
# .head() -> Top 5 values ['Original Table']
stud_series.head() # Provides Top 5 Values
```

```
Utkarsh      95
Prabhakar    92
Sanchita     91
Lubhani      93
Ali          77
dtype: int32
```

```
stud_series.head(7)
```

```
Utkarsh      95
Prabhakar    92
Sanchita     91
Lubhani      93
Ali          77
Kushagra     99
Aditya       81
dtype: int32
```

```
# .tail() -> Picking up the bottom values
stud_series.tail()
```

```
Ali         77
Kushagra    99
Aditya      81
Nihal       92
Surya       77
dtype: int32
```

```
stud_series.tail(7)
```

```
Sanchita    91
Lubhani     93
Ali         77
Kushagra    99
Aditya      81
Nihal       92
Surya       77
dtype: int32
```

```
stud_series.values
```

```
array([95, 92, 91, 93, 77, 99, 81, 92, 77])
```

```
# describe() -> Returns Statistical Summary of the Series
stud_series.describe()
```

```
count     9.000000
mean     88.555556
std       8.094923
min      77.000000
25%      81.000000
50%      92.000000
75%      93.000000
max      99.000000
dtype: float64
```

```
car_model.describe()
```

```
count          9
unique         9
top       Taigun
freq           1
Name: Car Model 😍, dtype: object
```

```
# Frequency ->
stud_series.value_counts()
```

```
92    2
77    2
95    1
91    1
93    1
99    1
81    1
Name: count, dtype: int64
```

```
# Frequency -> Car Model [Categorical Data]
car_model.value_counts()
```

```
Car Model 😍
Taigun      1
Thar        1
Magnite     1
Brezza      1
Harrier     1
Slavia      1
City        1
Fortuner    1
Creta       1
Name: count, dtype: int64
```

```python
# sort_values -> sorting the series by its values (default : ascending)
stud_series.sort_values()
```

```
Ali          77
Surya        77
Aditya       81
Sanchita     91
Prabhakar    92
Nihal        92
Lubhani      93
Utkarsh      95
Kushagra     99
dtype: int32
```

```python
# Descending -> (Attribute : ascending = False)
stud_series.sort_values(ascending = False)
```

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
Surya        77
dtype: int32
```

```python
# Sort the Series w.r.t Index -> sort_index (by default -> Ascending)
# In our case -> its a categorical value -> Alphabetical Order Sorting
stud_series.sort_index(ascending = True)
```

```
Aditya       81
Ali          77
Kushagra     99
Lubhani      93
Nihal        92
Prabhakar    92
Sanchita     91
Surya        77
Utkarsh      95
dtype: int32
```

```python
# Descending -> Alphabetical Order [Z->A]
stud_series.sort_index(ascending = False)
```

```
Utkarsh      95
Surya        77
Sanchita     91
Prabhakar    92
Nihal        92
Lubhani      93
Kushagra     99
Ali          77
Aditya       81
dtype: int32
```

```python
stud_series.index
```

```
Index(['Utkarsh', 'Prabhakar', 'Sanchita', 'Lubhani', 'Ali', 'Kushagra',
       'Aditya', 'Nihal', 'Surya'],
      dtype='object')
```

```
stud_series.values
```

```
array([95, 92, 91, 93, 77, 99, 81, 92, 77])
```

```
stud_series
```

```
Utkarsh      95
Prabhakar    92
Sanchita     91
Lubhani      93
Ali          77
Kushagra     99
Aditya       81
Nihal        92
Surya        77
dtype: int32
```

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

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
Surya        77
dtype: int32
```

```
# Marks High To Low
stud_series = stud_series.sort_values(ascending = False)
stud_series
```

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
Surya        77
dtype: int32
```

```
stud_series.index
```

```
Index(['Kushagra', 'Utkarsh', 'Lubhani', 'Prabhakar', 'Nihal', 'Sanchita',
       'Aditya', 'Ali', 'Surya'],
      dtype='object')
```

```
stud_series.values
```

```
array([99, 95, 93, 92, 92, 91, 81, 77, 77])
```

```python
# dropping an student from stud_series
stud_series.drop('Surya')
```

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
dtype: int32
```

```python
stud_series
```

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
Surya        77
dtype: int32
```

```python
# dropping an student from stud_series -> Permanently
stud_series.drop('Surya' , inplace = True)
```

```python
stud_series
```

```
Kushagra     99
Utkarsh      95
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
dtype: int32
```

```python
# Drop a stud_record where the student doesn't exist -> KeyError
stud_series.drop('Palash') # KeyError: "['Palash'] not found in axis"
```

```python
# replace -> Replace the marks where the marks = 95 change it to 97
stud_series.replace(95,97 , inplace = True) # Permanent Replace
stud_series
```

```
Kushagra     99
Utkarsh      97
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
dtype: int32
```

```python
stud_series.index.tolist()
```

```
['Kushagra',
 'Utkarsh',
 'Lubhani',
 'Prabhakar',
 'Nihal',
 'Sanchita',
 'Aditya',
 'Ali']
```

```python
stud_series[stud_series.index == 'Nihal']
```

```
Nihal    92
dtype: int32
```

```python
stud_series[stud_series.index == 'Nihal'].replace(92,96)
```

```
Nihal    96
dtype: int32
```

```
stud_series
```

```
Kushagra     99
Utkarsh      97
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
dtype: int32
```

```
# HomeWork
stud_series[stud_series.index == 'Nihal'].replace(92,96,inplace=True)
stud_series
```

```
Kushagra     99
Utkarsh      97
Lubhani      93
Prabhakar    92
Nihal        92
Sanchita     91
Aditya       81
Ali          77
dtype: int32
```

```
# isnull() / isna() / notnull() -> Boolean
stud_series.isnull() # False represent no null value present
```

```
Kushagra     False
Utkarsh      False
Lubhani      False
Prabhakar    False
Nihal        False
Sanchita     False
Aditya       False
Ali          False
dtype: bool
```

```
stud_series.isna()
```

```
Kushagra     False
Utkarsh      False
Lubhani      False
Prabhakar    False
Nihal        False
Sanchita     False
Aditya       False
Ali          False
dtype: bool
```

```
stud_series.notnull() # True
```

```
Kushagra     True
Utkarsh      True
Lubhani      True
Prabhakar    True
Nihal        True
Sanchita     True
Aditya       True
Ali          True
dtype: bool
```

```
stud_series.notnull().sum() # -> True[1] -> Total number of not null values
```

```
8
```

```
stud_series.isnull().sum() # -> False[0] -> Sum of all zero is 0
```

```
0
```

```
# Mathematical Operations on 2 different Series
seriesA = pd.Series([11,22,33,44,55] , index = ['a','b','c','d','e'])
seriesB = pd.Series([5,15,-9,-21,0] , index = ['p','q','a','d','c'])
print("\n SeriesA: ")
print(seriesA)
print("\n SeriesB: ")
print(seriesB)
```

```
 SeriesA:
a    11
b    22
c    33
d    44
e    55
dtype: int64

 SeriesB:
p     5
q    15
a    -9
d   -21
c     0
dtype: int64
```

```
resultSeries = seriesA + seriesB
resultSeries
```

```
a     2.0
b     NaN
c    33.0
d    23.0
e     NaN
p     NaN
q     NaN
dtype: float64
```

```
resultSeries = seriesA * seriesB
resultSeries
```

```
# Handling Missing Values [NaNs]
resultSeries.fillna(0) # Temporary Changes
```

```
a     2.0
b     0.0
c    33.0
d    23.0
e     0.0
p     0.0
q     0.0
dtype: float64
```

```
resultSeries
```

```
a     2.0
b     NaN
c    33.0
d    23.0
e     NaN
p     NaN
q     NaN
dtype: float64
```

```
resultSeries.fillna(0, inplace = True) # Permanent Changes
resultSeries
```

```
a     2.0
b     0.0
c    33.0
d    23.0
e     0.0
p     0.0
q     0.0
dtype: float64
```

```python
# Mathematical Operations on 2 different Series
seriesA = pd.Series([11,22,33,44,55] , index = ['a','b','c','d','e'])
seriesB = pd.Series([5,15,-9,-21,0] , index = ['p','q','a','d','c'])
print("\n SeriesA: ")
print(seriesA)
print("\n SeriesB: ")
print(seriesB)
```

```
 SeriesA:
a    11
b    22
c    33
d    44
e    55
dtype: int64

 SeriesB:
p     5
q    15
a    -9
d   -21
c     0
dtype: int64
```

```python
resultSeries = seriesA * seriesB
resultSeries
```

```
a    -99.0
b      NaN
c      0.0
d   -924.0
e      NaN
p      NaN
q      NaN
dtype: float64
```

```python
# Handle the missing Values -> by dropping the NaN
# [dropna]
resultSeries.dropna() # Temporary Changes
```

```
a    -99.0
c      0.0
d   -924.0
dtype: float64
```

```python
resultSeries
```

```
a    -99.0
b      NaN
c      0.0
d   -924.0
e      NaN
p      NaN
q      NaN
dtype: float64
```

```python
# Permanent Changes -> [inplace = True]
resultSeries.dropna(inplace = True)
resultSeries
```

```
a    -99.0
c      0.0
d   -924.0
dtype: float64
```