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

What is Pandas?

A high level data manipulation tools build on Numpy And Matplotlib.

It is used to :

- Import / Export Data Easily.
- Clean and Analyze the Data.
- Perform Statistical Operations.
- Visualize the data.

Why is Pandas Important?

1. Simple Syntax for Complex Task. Power Query Editor
2. Effecient Operations using Numpy in Backend.
3. Work with multiple formats - .csv , .excel , .json, .sql SQLAlchemy
4. Data Cleaning - Handle Missing or Inconsistent Value Imputing [Stats]
5. Powerful Analysis Tools - Filtering , Grouping , Pivoting , Melting , Aggregations, etc..

```
pip install pandas
conda install pandas
```

It has 2 types of Structures:

1. Series -> One Dimensional Array (Like One Column)
2. DataFrame -> Two Dimensional Array (Like a Table having rows or cols) {Spreadsheet}

```
import pandas as pd
```

What is Series?

A Series is :

1. A 1D Labelled Array of the data
2. Each element has an index
3. Can Store int , float, str, bool and object
4. Mutable (values can be updated)

`Note: Think of it as a single column from an Excel Sheet [Univariate Analysis]`

```
# Series (data , Index) [By Default indexing starts from 0]
import numpy as np
import pandas as pd
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
```

```
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) # Positional Argument
# series = pd.Series(data,index = label) # KeyWord Argument
series
```

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

```
# indexing starts with 1
data = [11,22,33,44,55,66,77,88,99]
label = range(1,10) # [1,2...9]
series = pd.Series(data,label) # Positional Argument
# series = pd.Series(data,index = label) # KeyWord Argument
series
```

```
1    11
2    22
3    33
4    44
5    55
6    66
7    77
8    88
9    99
dtype: int64
```

```python
# Dictionary {Key-Value} -> Series(Index - Data)
_employee_dict = {
    'emp_id' : 'emp101',
    'name' : 'Shyam Sundar',
    'age' : 27,
    'gender' : 'M',
    'salary' : '$10,00,000',
    'designation' : 'Senior Analyst',
    'email' : 'shyam.sundar@gmail.com',
    'state' : 'Andhra Pradesh',
    'country' : 'India'
}
series = pd.Series(_employee_dict)
series
```

```
emp_id                         emp101
name                      Shyam Sundar
age                                 27
gender                               M
salary                     $10,00,000
designation            Senior Analyst
email          shyam.sundar@gmail.com
state                  Andhra Pradesh
country                         India
dtype: object
```

```python
type(series)
```

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

```python
print(type(series))
```

```
<class 'pandas.core.series.Series'>
```

```python
# How to Access an elements from a Series.
# .iloc[Positional Based Indexing][Non-Inclusive] [1,10)
# .loc [Labelled Based Indexing][Inclusive] [1,10]
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
```

```python
print(series[-1]) # -ve indexing is not allowed
# KeyError: -1
```

```python
print(series.iloc[-1]) # 99 # only .iloc can perform -ve indexing
print(series[6]) # +ve indexing are allowed # 77
print(series.loc[0]) # 11
print(series.iloc[-5]) # 55
# print(series.loc[-5]) # Error # KeyError: -5
print(series.loc[4]) # 55
```

```
99
77
11
55
55
```

```python
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) # Positional Argument
print(series)
# print(series[0]) # KeyError
print(series['a']) # 11
print(series.iloc[-5]) # 55
print(series.iloc[7]) # 88
# print(series.loc[7]) # KeyError
print(series.loc['i']) # 99
```

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

```python
# Slicing
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) # Positional Argument
print(series)
print("\n Labelled Based Slicing using .loc")
print(series.loc['a':'e']) # [11,22,33,44,55]
print("\n Positional Based Slicing using .iloc")
print(series.iloc[0:5]) # [11,22,33,44,55]
```

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

 Labelled Based Slicing using .loc
a    11
b    22
c    33
d    44
e    55
dtype: int64

 Positional Based Slicing using .iloc
a    11
b    22
c    33
d    44
e    55
dtype: int64
```

```python
# Dictionary {Key-Value} -> Series(Index - Data)
_employee_dict = {
    'emp_id' : 'emp101',
    'name' : 'Shyam Sundar',
    'age' : 27,
    'gender' : 'M',
    'salary' : '$10,00,000',
    'designation' : 'Senior Analyst',
    'email' : 'shyam.sundar@gmail.com',
    'state' : 'Andhra Pradesh',
    'country' : 'India'
}
series = pd.Series(_employee_dict)
print(series)
# Normal Indexing
print(series['email']) # 'shyam.sundar@gmail.com'
# print(series[4]) # KeyError
print(series.iloc[4]) # '$10,00,000'
```

```
emp_id                         emp101
name                      Shyam Sundar
age                                 27
gender                               M
salary                     $10,00,000
designation             Senior Analyst
email          shyam.sundar@gmail.com
state                   Andhra Pradesh
country                          India
dtype: object
shyam.sundar@gmail.com
$10,00,000
```

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

```
Andhra Pradesh
name                      Shyam Sundar
age                                 27
gender                               M
salary                     $10,00,000
designation             Senior Analyst
email          shyam.sundar@gmail.com
dtype: object
```

```python
print(series.loc['name' : 'postalcode']) # slicing[Inclusive]
# KeyError: 'postalcode'
```

```python
# .loc[Calling Multiple Columns]
print(series.loc[['name','email','designation','salary','state']])
```

```
name                      Shyam Sundar
email          shyam.sundar@gmail.com
designation             Senior Analyst
salary                     $10,00,000
state                   Andhra Pradesh
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] # [11-44]
```

```
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
```

```
11
```

```
series.loc[0:4] # [11 to 55] [Inclusive]
```

```
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) # Positional Argument
print(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] # [11-44]
```

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

```
series.loc['a':'e'] # [11-55]
```

```
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, # keyword-argument
    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
```

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

```
car_model.values
```

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

```
car_model.dtypes
```

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

```
car_model.shape
```

```
(9,)
```

```
car_model.size
```

```
9
```

```
car_model.empty # True/False [False-> as it holds data in it]
```

```
False
```

```
# hasnans -> has NaNs [Not a Number]
car_model.hasnans # False [No Missing Value]
```

```
False
```

```
# SELECT <DISTINCT> NAME FROM TABLENAME -> Unique
car_model.is_unique # True
```

```
True
```

```
# ndim [Numpy N-Dimensional Array]
car_model.ndim # 1 [Series]
```

```
1
```

```
# Basic Mathematical Operations
series = pd.Series([7,9,11,15,17,21,29,55,77,99])
print(series+11)
```

```
0     18
1     20
2     22
3     26
4     28
5     32
6     40
7     66
8     88
9    110
dtype: int64
```

```
series = pd.Series([7,9,11,15,17,21,29,55,77,99])
print(series**2)
```

```
0      49
1      81
2     121
3     225
4     289
5     441
6     841
7    3025
8    5929
9    9801
dtype: int64
```

```
series = pd.Series([7,9,11,15,17,21,29,55,77,99])
print(series * 10)
```

```
0     70
1     90
2    110
3    150
4    170
5    210
6    290
7    550
8    770
9    990
dtype: int64
```

```
series = pd.Series([7,9,11,15,17,21,29,55,77,99])
print(series // 10)
```

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

```
# Attributes of Series
car_list = np.array(['Taigun','Thar','Magnite','Brezza','City','Thar',
                     'Harrier','Slavia','City','Fortuner','Creta'])
car_model = pd.Series(
    car_list,
    name = 'Car Model 😍'
)
car_model
```

```
0       Taigun
1         Thar
2      Magnite
3       Brezza
4         City
5         Thar
6      Harrier
7       Slavia
8         City
9     Fortuner
10       Creta
Name: Car Model 😍, dtype: object
```

```
car_model.is_unique # False
```

```
False
```

```
# Students_Data -> Name as Index , and Marks as Value
name = pd.Series(['Shyam','Swinki','Rajat','Vaibhav','Deepak','Harisha','Pragya','Aryan',
                  'Shahzain','Kajal','Umang','Shubham'])
marks = np.array([95,99,91,89,85,91,77,95,97,92,81,90])
stud_series = pd.Series(marks,name)
stud_series
```

```
Shyam       95
Swinki      99
Rajat       91
Vaibhav     89
Deepak      85
Harisha     91
Pragya      77
Aryan       95
Shahzain    97
Kajal       92
Umang       81
Shubham     90
dtype: int32
```

```
stud_series.index
```

```
Index(['Shyam', 'Swinki', 'Rajat', 'Vaibhav', 'Deepak', 'Harisha', 'Pragya',
       'Aryan', 'Shahzain', 'Kajal', 'Umang', 'Shubham'],
      dtype='object')
```

```
stud_series.values
```

```
array([95, 99, 91, 89, 85, 91, 77, 95, 97, 92, 81, 90])
```

```
# Indexing .iloc Vs loc
stud_series['Shubham'] # case-sensitive
```

```
90
```

```
# stud_series[7] # KeyError
stud_series.loc['Swinki']
```

```
99
```

```
stud_series.iloc[-3]
```

```
92
```

```
stud_series.loc[['Shyam','Rajat','Deepak']]
```

```
Shyam     95
Rajat     91
Deepak    85
dtype: int32
```

```
# Limit -> Order By ['Sorting'] ['SQL']
# .head() -> preview of the Series/DataFrame [by default 5 rows returns]
stud_series.head() # First 5 values from the Series
```

```
Shyam      95
Swinki     99
Rajat      91
Vaibhav    89
Deepak     85
dtype: int32
```

```
# .tail() -> preview of the Series/DataFrame [by default 5 rows returns]
stud_series.tail() # Last 5 values from the series
```

```
Aryan       95
Shahzain    97
Kajal       92
Umang       81
Shubham     90
dtype: int32
```

```
stud_series.head(7) # First 7 values from the Series
```

```
Shyam      95
Swinki     99
Rajat      91
Vaibhav    89
Deepak     85
Harisha    91
Pragya     77
dtype: int32
```

```
stud_series.tail(9) # Last 9 values from the Series
```

```
Vaibhav     89
Deepak      85
Harisha     91
Pragya      77
Aryan       95
Shahzain    97
Kajal       92
Umang       81
Shubham     90
dtype: int32
```

```
stud_series.values
```

```
array([95, 99, 91, 89, 85, 91, 77, 95, 97, 92, 81, 90])
```

```
stud_series.values.dtype
```

```
dtype('int32')
```

```
# Data Profiling [Statistical Analysis]
stud_series.describe()
```

```
count    12.000000
mean     90.166667
std       6.478402
min      77.000000
25%      88.000000
50%      91.000000
75%      95.000000
max      99.000000
dtype: float64
```

```
car_model.describe()
```

```
count       11
unique       9
top       Thar
freq         2
Name: Car Model 😍, dtype: object
```

```python
# Frequency - Mode
car_model.value_counts()
```

```
Car Model 😍
Thar        2
City        2
Taigun      1
Magnite     1
Brezza      1
Harrier     1
Slavia      1
Fortuner    1
Creta       1
Name: count, dtype: int64
```

```python
# Sorting -> Ascending To Descending
stud_series.sort_values()
```

```
Pragya      77
Umang       81
Deepak      85
Vaibhav     89
Shubham     90
Rajat       91
Harisha     91
Kajal       92
Shyam       95
Aryan       95
Shahzain    97
Swinki      99
dtype: int32
```

```python
# Sorting -> High to Low []
# Sort a Series in ascending or descending order by some criterion.
stud_series.sort_values(ascending = False) # Desc to Asc
```

```
Swinki      99
Shahzain    97
Shyam       95
Aryan       95
Kajal       92
Rajat       91
Harisha     91
Shubham     90
Vaibhav     89
Deepak      85
Umang       81
Pragya      77
dtype: int32
```

```python
stud_series # Orginal Table as the above changes are Temporary
```

```
Shyam       95
Swinki      99
Rajat       91
Vaibhav     89
Deepak      85
Harisha     91
Pragya      77
Aryan       95
Shahzain    97
Kajal       92
Umang       81
Shubham     90
dtype: int32
```

```
# sort_index() # Object [Albhatically Sort -> 'ASCII'] 'A'->'Z'
stud_series.sort_index()
```

```
Aryan       95
Deepak      85
Harisha     91
Kajal       92
Pragya      77
Rajat       91
Shahzain    97
Shubham     90
Shyam       95
Swinki      99
Umang       81
Vaibhav     89
dtype: int32
```

```
# sort_index() # Object [Albhatically Sort -> 'ASCII'] 'Z'->'A'
stud_series.sort_index(ascending = False) # reversed
```

```
Vaibhav     89
Umang       81
Swinki      99
Shyam       95
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Harisha     91
Deepak      85
Aryan       95
dtype: int32
```

```
stud_series # Orginal Table as the above changes are Temporary
```

```
Shyam       95
Swinki      99
Rajat       91
Vaibhav     89
Deepak      85
Harisha     91
Pragya      77
Aryan       95
Shahzain    97
Kajal       92
Umang       81
Shubham     90
dtype: int32
```

```
stud_series.index
```

```
Index(['Shyam', 'Swinki', 'Rajat', 'Vaibhav', 'Deepak', 'Harisha', 'Pragya',
       'Aryan', 'Shahzain', 'Kajal', 'Umang', 'Shubham'],
      dtype='object')
```

```
stud_series.values
```

```
array([95, 99, 91, 89, 85, 91, 77, 95, 97, 92, 81, 90])
```

```
0-9 > A to Z > a to z, in Ascending Order
0-9 < A to Z < a to z, in Descending Order
```

```python
# Permanent Changes
# Marks High to Low
stud_series = stud_series.sort_values(ascending = False)
stud_series
```

```
Swinki      99
Shahzain    97
Shyam       95
Aryan       95
Kajal       92
Rajat       91
Harisha     91
Shubham     90
Vaibhav     89
Deepak      85
Umang       81
Pragya      77
dtype: int32
```

```python
stud_series.index
```

```
Index(['Swinki', 'Shahzain', 'Shyam', 'Aryan', 'Kajal', 'Rajat', 'Harisha',
       'Shubham', 'Vaibhav', 'Deepak', 'Umang', 'Pragya'],
      dtype='object')
```

```python
stud_series.values
```

```
array([99, 97, 95, 95, 92, 91, 91, 90, 89, 85, 81, 77])
```

```python
# Permanent Changes
stud_series.sort_index(ascending = False , inplace = True) # 'Z' to 'A'
stud_series
```

```
Vaibhav     89
Umang       81
Swinki      99
Shyam       95
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Harisha     91
Deepak      85
Aryan       95
dtype: int32
```

```python
stud_series.index
```

```
Index(['Vaibhav', 'Umang', 'Swinki', 'Shyam', 'Shubham', 'Shahzain', 'Rajat',
       'Pragya', 'Kajal', 'Harisha', 'Deepak', 'Aryan'],
      dtype='object')
```

```python
stud_series.values
```

```
array([89, 81, 99, 95, 90, 97, 91, 77, 92, 91, 85, 95])
```

```
# 'Dropping an elements' from a Series
stud_series.drop('Umang') # Temporary Drop
```

```
Vaibhav     89
Swinki      99
Shyam       95
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Harisha     91
Deepak      85
Aryan       95
dtype: int32
```

```
stud_series # Changes won't reflected on my original Series
```

```
Vaibhav     89
Umang       81
Swinki      99
Shyam       95
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Harisha     91
Deepak      85
Aryan       95
dtype: int32
```

```
# If you are dropping a person from stud_series that doesn't exist.
stud_series.drop('Chirag' , inplace = True)
stud_series # KeyError: "['Chirag'] not found in axis"
```

```
# L.H.S = R.H.S [Permanent Drop]
stud_series = stud_series.drop('Harisha')
stud_series
```

```
Vaibhav     89
Swinki      99
Shyam       95
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Deepak      85
Aryan       95
dtype: int32
```

```
# .replace() -> String Manipulation
stud_series.replace(95,98,inplace = True)
stud_series
```

```
Vaibhav     89
Swinki      99
Shyam       98
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Deepak      85
Aryan       98
dtype: int32
```

```
# .tolist()
stud_series.index.tolist()
```

```
['Vaibhav',
 'Swinki',
 'Shyam',
 'Shubham',
 'Shahzain',
 'Rajat',
 'Pragya',
 'Kajal',
 'Deepak',
 'Aryan']
```

```
type(stud_series.index.tolist())
```

```
list
```

```
stud_series[stud_series.index == 'Aryan']
```

```
Aryan    98
dtype: int32
```

```
stud_series[stud_series.index == 'Aryan'].replace(98,95)
```

```
Aryan    95
dtype: int32
```

```
stud_series[stud_series.index == 'Aryan'].replace(98,95,inplace = True)
```

```
stud_series[stud_series.index == 'Aryan'] = stud_series[stud_series.index == 'Aryan'].replace(98,95)
stud_series
```

```
Vaibhav     89
Swinki      99
Shyam       98
Shubham     90
Shahzain    97
Rajat       91
Pragya      77
Kajal       92
Deepak      85
Aryan       95
dtype: int32
```

```
stud_series = stud_series[stud_series.index == 'Aryan'].replace(98,95)
stud_series # Aryan in Stud_series
```

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

```
Vaibhav     False
Swinki      False
Shyam       False
Shubham     False
Shahzain    False
Rajat       False
Pragya      False
Kajal       False
Deepak      False
Aryan       False
dtype: bool
```

```
stud_series.isna()
```

```
Vaibhav     False
Swinki      False
Shyam       False
Shubham     False
Shahzain    False
Rajat       False
Pragya      False
Kajal       False
Deepak      False
Aryan       False
dtype: bool
```

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

```
Vaibhav     True
Swinki      True
Shyam       True
Shubham     True
Shahzain    True
Rajat       True
Pragya      True
Kajal       True
Deepak      True
Aryan       True
dtype: bool
```

```
stud_series.notnull().sum() # 10
```

```
10
```

```
stud_series.isna().sum() # False [0] -> 0
```

```
0
```

```
# Mathematical Operations on 2 Different Series
label1 = ['a','b','c','d','e']
label2 =  ['p','q','a','d','c']
seriesA = pd.Series([11,22,33,44,55] , index = label1)
seriesB = pd.Series([5,15,-9,-21,0] , index = label2)
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
```

```
print(seriesA.shape)
print(seriesB.shape)
```

```
(5,)
(5,)
```

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

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

```
resultSeries.isna()
```

```
a    False
b     True
c    False
d    False
e     True
p     True
q     True
dtype: bool
```

```
resultSeries.isna().sum() # Count of Missing Values
```

```
4
```

```
resultSeries.notnull() # Boolean Return [True for Non-NullValue]
```

```
a     True
b    False
c     True
d     True
e    False
p    False
q    False
dtype: bool
```

```
resultSeries.notnull().sum() # Count of Non-Null Values
```

```
3
```

```
# Cleaning -> Handling a Missing Values
# Impact Analysis [.drop] Vs Imputing[.fill]
resultSeries.dropna() # Temporary Changes
```

```
a     2.0
c    33.0
d    23.0
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)
```

```
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)
resultSeries
```

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

```
# Mathematical Operations on 2 Different Series
label1 = ['a','b','c','d','e']
label2 =  ['p','q','a','d','c']
seriesA = pd.Series([11,22,33,44,55] , index = label1)
seriesB = pd.Series([5,15,-9,-21,0] , index = label2)
print("\n SeriesA:")
print(seriesA)
print("\n SeriesB:")
```

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

```
resultSeries.dropna(inplace = True)
resultSeries # Permanent Changes
```

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