Chapter - 2
Data Handling Using Pandas - I

Que 1. What is a Series and how is it different from a 1-D array, a list and a dictionary?

Ans. A Series is a one-dimensional array containing a sequence of values of any data type (int, float, list, string, etc) which by default have numeric data labels starting from zero.

The data label associated with a particular value is called its index.

Series vs List

Series	List
1. It is a 1-D data structure. 2. It can have numeric indexes as well as labels. 3. It supports explicit indexing i.e. we can define our own other than default indexing. 4. Duplicate Index can by given. 5. Same types elements contain by Series i.e. Homogeneous Elements.	1. It can 1-D as well as multi-dimensional data structure (nested list). 2. It can take only numeric indexes. 3. Explicit indexing is not possible. Only supports default indexing i.e. 0 to n-1 or -1 to -n. (n is the number of elements. 4. Indexes can not be duplicate. 5. List can store elements of different types i.e. Heterogenous elements

Series vs Dictionary

Series	Dictionary
1. It is essentially a 1D data structure. 2. Its indexes can be numbers as well as labels. 3. Series stores values against indexes/labels, similar to dictionary.	1. It can be1D as well as multi-dimensional (nested dictionary) 2. Its keys can be only of immutable types only. 3. Dictionary stores values against keys, similar to Series.

Series vs NumPy Array

Series	NumPy Array (ndarray)
1. It supports explicit indexing. 2. It supports indexes of numeric as well as string types. 3. It can perform vectorized operations on two similar shape series as well as two dissimilar shapes series. (Using NaN for non-matching indexes/labels)	1. It does not supports explicit indexing. 2. It supports indexes of numeric types only. 3. It can perform vectorized operations only on two

4. It consume more memory compare to NumPy Array	similar types of ndarray. 4. It consume less memory compare to Series.

Que 2. What is a DataFrame and how is it different from a 2-D array?

Ans. A DataFrame is a two-dimensional labelled data structure like a table of MySQL.

It contains rows and columns, and therefore has both a row and column index. Each column can have a different type of value such as numeric, string, boolean, etc., as in tables of a database.

Pandas store such tabular data using a DataFrame.

Dataframe vs 2-D Array (2D ndarray)

Dataframe	2 D Array (2D ndarray)
1. It can store heterogeneous element of different date types. 2. It can have indexes as well as labels for rows and columns. 3. It consumes more memory. (of same size ndarray) 4. Dataframes are expandable. We can add new elements or delete old elements.	1. It can store homogeneous elements generally numbers. 2. It is indexed by positive integers for both rows and columns. 3. It consume lesser memory. (of same size Dataframe). 4. Not expandable. Can not add or delete element in/from the same array.

Que 3. How are DataFrames related to Series?

Ans. Dataframe is the collection of Series. Individual columns of dataframe's can be considered equivalent to Series object.

Differences:-

(A) Individual columns of dataframe are size-mutable, while series are not a size mutable.

(B) Dataframe can contain heterogeneous data while Series can contain homogeneous data.

Que 4. What do you understand by the size of (i) a Series, (ii) a DataFrame?

Ans. (i) size of Series means number of elements in Series. size is an attribute of Series object, which returns number of elements.

seriesObject.size

Example:

```
>>> import pandas as pd
>>> s = pd.Series([1,2,3])
>>> s
0 1
1 2
2 3
dtype: int64
>>> s.size
3
```

(ii) size of Dataframe means number of elements in Dataframe. size is an attribute of DataFrame object, which returns number of elements.

dataframeObject.size

Example:

```
>>> import pandas as pd
>>> d = pd.DataFrame([[1,2,3],[4,5,6]])
>>> d
0 1 2
0 1 2 3
1 4 5 6
>>> d.size
6
```

Que 5. Create the following Series and do the specified operations:

a) EngAlph, having 26 elements with the alphabets as values and default index values.

b) Vowels, having 5 elements with index labels 'a', 'e', 'i', 'o' and 'u' and all the five values set to zero. Check if it is an empty series.

c) Friends, from a dictionary having roll numbers of five of your friends as data and their first name as keys.

d) MTseries, an empty Series. Check if it is an empty series.

e) MonthDays, from a numpy array having the number of days in the 12 months of a year. The labels should be the month numbers from 1 to 12.

Ans.

(a) Method -1

```
>>> import pandas as pd
>>> list1 = [chr(n) for n in range(97, 123)]
>>> EngAlph = pd.Series(list1)
>>> print(EngAlph)
```

Method-2

```
>>> import pandas as pd
>>> import string
>>> list1 = list(string.ascii_lowercase)
>>> EngAlph = pd.Series(list1)
>>> EngAlph
```

(b)

```
>>> import pandas as pd
>>> Vowels = pd.Series(0, index = ['a', 'e', 'i', 'o', 'u'])
>>> Vowels
>>> Vowles.empty
False #Not Empty
```

(c) >>> import pandas as pd

>>> dict = {'Suman' : 1, 'Raman' : 2, 'Anjeev' : 3 , 'Singh' : 4, 'Mohit': 5}

>>> Friends = pd.Series(dict)

>>> Friends

Suman 1
Raman 2
Anjeev 3
Singh 4
Mohit 5
dtype: int64

(d) >>> import pandas as pd

>>> MTseries = pd.Series([])
# pd.Series( ) is also valid for creating empty Series, but it raise DeprecationWarning
>>> MTSeries.empty

True

(e) >>> import numpy as np

>>> import pandas as pd

>>> monthDaysAr = np.array([31,28,31,30,31,30,31,31,30,31,30,31])

>>> MonthDays = pd.Series(monthDaysAr, index = range(1,13))

>>> MonthDays

Que 6. Using the Series created in Question 5, write commands for the following:

a) Set all the values of Vowels to 10 and display the Series.

b) Divide all values of Vowels by 2 and display the Series.

c) Create another series Vowels1 having 5 elements with index labels 'a', 'e', 'i', 'o' and 'u' having values [2,5,6,3,8] respectively.

d) Add Vowels and Vowels1 and assign the result to Vowels3.

e) Subtract, Multiply and Divide Vowels by Vowels1.

f) Alter the labels of Vowels1 to ['A', 'E', 'I', 'O', 'U'].

Ans.

(a) >>> Vowels[:] = 10

>>> Vowels

a 10

e 10

i 10

o 10

u 10

(b) >>> Vowels = Vowels / 2

>>> Vowels

a 5.0

e 5.0

i 5.0

o 5.0

u 5.0

(c) >>> import pandas as pd

>>> Vowels1 = pd.Series([2, 5, 6, 3, 8] , index = ['a', 'e', 'i', 'o', 'u'])

>>> Vowels1

a 2

e 5

i 6

o 3

u 8

(d) >>> Vowels3 = Vowels + Vowels1

>>> Vowels3

a 7.0

e 10.0

i 11.0

o 8.0

u 13.0

(e) >>> Vowels – Vowels1

a 3.0

e 0.0

i -1.0

o 2.0

u -3.0

>>> Vowels * Vowels1

a 10.0

e 25.0

i 30.0

o 15.0

u 40.0

>>> Vowels / Vowels1

a 2.500000

e 1.000000

i 0.833333

o 1.666667

u 0.625000

(f) >>> Vowels1.index = ['A', 'E', 'I', 'O', 'U']

>>> Vowles1

A 2

E 5

I 6

O 3

U 8

Que 7. Using the Series created in Question 5, write commands for the following:

a) Find the dimensions, size and values of the Series EngAlph, Vowels, Friends, MTseries, MonthDays.

b) Rename the Series MTseries as SeriesEmpty.

c) Name the index of the Series MonthDays as monthno and that of Series Friends as Fname.

d) Display the 3rd and 2nd value of the Series Notes Friends, in that order.

e) Display the alphabets 'e' to 'p' from the Series EngAlph.

f) Display the first 10 values in the Series EngAlph.

g) Display the last 10 values in the Series EngAlph.

h) Display the MTseries.

Ans.

(a) Series EngAlph

```
>>> EngAlph.shape
(26, )
>>> EngAlph.size
26
>>> EngAlph.values
array(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'],
dtype=object)
```

Series Vowels

```
>>> print(Vowels.shape)
(5,)
>>> print(Vowels.size)
5
>>> print(Vowels.values)
[5. 5. 5. 5. 5.]
>>> Vowels.values
array([5., 5., 5., 5., 5.])
```

Series Friends

```
>>> print(Friends.shape)
(5,)

>>> print(Friends.size)
5


>>> print(Friends.values)
[1 2 3 4 5]
```

Series MTSeries

>>> print(MTseries.shape)

(0,)

>>> print(MTseries.size)

0

>>> print(MTseries.values)

[]

Serries MonthDays

>>> print(MonthDays.shape)

(12,)

>>> print(MonthDays.size)

12

>>> print(MonthDays.values)

[31 28 31 30 31 30 31 31 30 31 30 31]

(b) >>> MTseries.rename("SeriesEmpty")

Series([], Name: SeriesEmpty, dtype: float64)

(c) >>> MonthDays.index.name = 'monthno'

>>> Friends.index.name = 'Fname'

(d) >>> Friends[2 : 0 : -1]

Anjeev 3
Raman 2
dtype: int64

(e) >>> EngAlph[4 : 16]

4 e
5 f
6 g
7 h
8 i

9 j
10 k
11 l
12 m
13 n
14 o
15 p
dtype: object

(f) >>> EngAlph.head(10)
0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
dtype: object

Note : EngAlph[0:10] -> Gives the same output

(g) >>> EngAlph.tail(10)
16 q
17 r
18 s
19 t
20 u
21 v
22 w
23 x
24 y
25 z
dtype: object
Note : EngAlph[16:26] , will give the same output

(h) **>>> print(MTseries)**
Series([], dtype: float64)

Que 8. Using the Series created in Question 5, write commands for the following:

a) Display the names of the months 3 through 7 from the Series MonthDays.

b) Display the Series MonthDays in reverse order.

Ans.

(a)

```
MonthDays[2:7]
3 31
4 30
5 31
6 30
7 31
dtype: int32
```

(b)

```
MonthDays[::-1]
12 31
11 30
10 31
9 30
8 31
7 31
6 30
5 31
4 30
3 31
2 28
1 31
dtype: int32
```

Que 9. Create the following DataFrame Sales containing year wise sales figures for five sales persons in INR. Use the years as column labels, and sales person names as row labels.

	2014	2015	2016	2017
Madhu	100.5	12000	20000	50000
Kusum	150.8	18000	50000	60000
Kinshuk	200.9	22000	70000	70000
Ankit	30000	30000	100000	80000
Shruti	40000	45000	125000	90000

Ans.

```
salesDict = {
2014 : {'Madhu' : 100.5, 'Kusum' : 150.8,
'Kinshuk' : 200.9, 'Ankit':30000, 'Shruti': 40000},
2015 : {'Madhu' : 12000, 'Kusum' : 18000,
'Kinshuk' : 22000, 'Ankit':30000, 'Shruti':
45000},
2016 : {'Madhu' : 20000, 'Kusum' : 50000,
'Kinshuk' : 70000, 'Ankit':100000, 'Shruti':
125000}, 2017 : {'Madhu' : 50000, 'Kusum' :
60000, 'Kinshuk' : 70000, 'Ankit':80000, 'Shruti':
90000}
}
Sales = pd.DataFrame(salesDict)
print(Sales)
```

Output:

```
2014 2015 2016 2017
Madhu 100.5 12000 20000 50000
Kusum 150.8 18000 50000 60000
Kinshuk 200.9 22000 70000 70000
Ankit 30000.0 30000 100000 80000
Shruti 40000.0 45000 125000 90000
```

Que 10. Use the DataFrame created in Question 9 above to do the following:

a) Display the row labels of Sales.

b) Display the column labels of Sales.

c) Display the data types of each column of Sales.

d) Display the dimensions, shape, size and values of Sales.

e) Display the last two rows of Sales.

f) Display the first two columns of Sales.

g) Create a dictionary using the following data. Use this dictionary to create a DataFrame Sales2.

	2018
Madhu	160000
Kusum	110000
Kinshuk	500000
Ankit	340000
Shruti	900000

h) Check if Sales2 is empty or it contains data.

Ans.

(a) >>> Sales.index
Index(['Madhu', 'Kusum', 'Kinshuk', 'Ankit', 'Shruti'], dtype='object')

(b) >>> Sales.columns
Int64Index([2014, 2015, 2016, 2017], dtype='int64')

(c) >>> Sales.dtypes
2014 float64
2015 int64
2016 int64
2017 int64
dtype: object

(d) >>> Sales.ndim, Sales.shape, Sales.size, Sales.values
(2, (5, 4), 20, array([[1.005e+02, 1.200e+04, 2.000e+04, 5.000e+04],
[1.508e+02, 1.800e+04, 5.000e+04, 6.000e+04],
[2.009e+02, 2.200e+04, 7.000e+04, 7.000e+04],
[3.000e+04, 3.000e+04, 1.000e+05, 8.000e+04],
[4.000e+04, 4.500e+04, 1.250e+05, 9.000e+04]]))

(e) Method 1:

```
>>> Sales.iloc[3:,]
2014 2015 2016 2017
Ankit 30000.0 30000 100000 80000
Shruti 40000.0 45000 125000 90000
```

Method 2:

```
Sales.tail(2)
2014 2015 2016 2017
Ankit 30000.0 30000 100000 80000
Shruti 40000.0 45000 125000 90000
```

(f)

```
>>> Sales.iloc[:,:2]
            2014   2015
Madhu      100.5  12000
Kusum      150.8  18000
Kinshuk    200.9  22000
Ankit    30000.0  30000
Shruti   40000.0  45000
```

(g)

```
>>> Dict2 = {2018 : [160000, 110000, 500000, 840000, 900000]}
>>> Sales2 = pd.DataFrame(Dict2, index=['Madhu', 'Kusum', 'Kinshuk',
'Ankit', 'Shruti'])
>>> Sales2
          2018
Madhu   160000
Kusum   110000
Kinshuk 500000
Ankit   840000
Shruti  900000
```

(h) >>> Sales2.empty

False

Que 11. Use the DataFrame created in Question 9 above to do the following:

a) Append the DataFrame Sales2 to the DataFrame Sales.

b) Change the DataFrame Sales such that it becomes its transpose.

c) Display the sales made by all sales persons in the year 2017.

d) Display the sales made by Madhu and Ankit in the year 2017 and 2018.

e) Display the sales made by Shruti 2016.

f) Add data to Sales for salesman Sumeet where the sales made are [196.2, 37800, 52000, 78438, 38852] in the years [2014, 2015, 2016, 2017, 2018] respectively.

g) Delete the data for the year 2014 from the DataFrame Sales.

h) Delete the data for sales man Kinshuk from the DataFrame Sales.

i) Change the name of the salesperson Ankit to Vivaan and Madhu to Shailesh.

j) Update the sale made by Shailesh in 2018 to 100000.

k) Write the values of DataFrame Sales to a comma separated file SalesFigures.csv on the disk. Do not write the row labels and column labels.

l) Read the data in the file SalesFigures.csv into a DataFrame SalesRetrieved and Display it. Now update the row labels and column labels of SalesRetrieved to be the same as that of Sales.

2. Data Handling Using Pandas - I

Ans.

(a) >>> Sales = pd.concat([Sales, Sales2], axis = 1])

```
>>> Sales
            2014   2015    2016   2017    2018
Madhu      100.5  12000   20000  50000  160000
Kusum      150.8  18000   50000  60000  110000
Kinshuk    200.9  22000   70000  70000  500000
Ankit    30000.0  30000  100000  80000  840000
Shruti   40000.0  45000  125000  90000  900000
```

(b)

```
>>> Sales = Sales.T
>>> Sales
         Madhu     Kusum   Kinshuk     Ankit    Shruti
2014     100.5     150.8     200.9   30000.0   40000.0
2015   12000.0   18000.0   22000.0   30000.0   45000.0
2016   20000.0   50000.0   70000.0  100000.0  125000.0
2017   50000.0   60000.0   70000.0   80000.0   90000.0
2018  160000.0  110000.0  500000.0  840000.0  900000.0
```

(c) >>> Sales.loc[2017, :] # transposed sales

(d) >>> Sales.loc[[2017, 2018], ['Madhu', 'Ankit']] # transposed sales

(e) >>> Sales.loc[[2016], ['Shruti']]

(f) #For Transposed Sales
SalesT.loc[:, 'Sumit'] = [196.2, 37800, 52000, 78438, 38852]

#For Actual Sales
Sales.loc['Sumit',:] = [196.2, 37800, 52000, 78438, 38852]

```
>>> SalesT.loc[:, 'Sumit'] = [196.2, 37800, 52000, 78438, 38852]
>>> SalesT
         Madhu     Kusum   Kinshuk     Ankit    Shruti    Sumit
2014     100.5     150.8     200.9   30000.0   40000.0    196.2
2015   12000.0   18000.0   22000.0   30000.0   45000.0  37800.0
2016   20000.0   50000.0   70000.0  100000.0  125000.0  52000.0
2017   50000.0   60000.0   70000.0   80000.0   90000.0  78438.0
2018  160000.0  110000.0  500000.0  840000.0  900000.0  38852.0
>>> Sales.loc['Sumit',:] = [196.2, 37800, 52000, 78438, 38852]
>>> Sales
            2014     2015      2016     2017      2018
Madhu      100.5  12000.0   20000.0  50000.0  160000.0
Kusum      150.8  18000.0   50000.0  60000.0  110000.0
Kinshuk    200.9  22000.0   70000.0  70000.0  500000.0
Ankit    30000.0  30000.0  100000.0  80000.0  840000.0
Shruti   40000.0  45000.0  125000.0  90000.0  900000.0
Sumit      196.2  37800.0   52000.0  78438.0   38852.0
>>>
```

(g) del Sales[2014]

(h) drop Seles(['Kinshuk'])

(i) Sales.rename (index = {‘Ankit’ : ‘ Vivaan’, ‘Madhu’ : ‘Shailesh’})

(j) Sales.loc[‘Shailesh’, 2018] = 100000

(k) >>> Sales.to_csv(“d:\\MyPythonProgram\\SalesFigures.csv”, header = False, index = False)

(l) >>> SalesRetrived = pd.read_csv(“d:\\MyPythonProgram\\SalesFigures.csv”, names = [2014, 2015, 2016, 2017, 2018])
>>> Sales.Retrived = SalesRetrived.rename(index = {0 : ‘Madhu’, 1 : ‘Kusum’, 2 : ‘Kinshuk’, 3 : ‘Ankit’, 4 : ‘Shruti’, 5 : ‘Sumeet’})