Experiment - 2

Objective: Understanding data formats of Pandas: Series, Dataframe. Importing different types of Datasets.

Pandas:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Pandas deals with the following three data structures:

- Series
- DataFrame
- Panel

These data structures are built on top of NumPy array, which means they are fast.

Dimension & Description

Data Structure	**Dimensions**	**Description**
Series	1	1D labeled homogeneous array, size-immutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

Mutability: All Pandas data structures are value mutable (i.e., can be changed) and except Series all are size mutable. Series is size immutable.

* * Series * *

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

`pandas.Series`

A pandas Series can be created using the following constructor −

```
pandas.Series( data, index, dtype, copy)
```

The parameters of the constructor are as follows:

Sr. No	**Parameter & Description**
1	**data:** data takes various forms like ndarray, list, constants
2	**index:** Index values must be unique and hashable, same length as data. Default `np.arrange(n)` if no index is passed.
3	**dtype:** dtype is for data type. If None, data type will be inferred
4	**copy:** Copy data. Default False

A series can be created using various inputs like:

- Array
- Dict [means **dictionary**]
- Scalar value or constant

Program 1: Create a basic series (Empty Series)

```
#import the pandas library and aliasing as pd
import pandas as pd
srs = pd.Series()
print (srs)
```

Output:

```
Series([], dtype: float64)
```

Create a Series from ndarray:

If data is an ndarray, then index passed must be of the same length. If no index is passed, then by default index will be `range(n)` where n is array length, i.e., [0, 1, 2, 3, …., range(len(array))-1].

Program 2: Create a series using array

```
#import the pandas library and aliasing as pd
import pandas as pd
#import the numpy library and aliasing as np
import numpy as np
arr = np.array (['a', 'b', 'c', 'd'])
print (arr)
print ('\n')
srs = pd.Series (arr)
print (srs)
```

Output:

```
['a' 'b' 'c' 'd']

0    a
1    b
2    c
3    d
dtype: object
```

Program 3: Create a series using array and explicitly specify the index

```
#import the pandas library and aliasing as pd
import pandas as pd
#import the numpy library and aliasing as np
import numpy as np
arr = np.array (['a', 'b', 'c', 'd'])
print (arr)
print ('\n')
srs = pd.Series (arr, index = [5, 7, 9, 11])
```

```
print (srs)
```

Output:

```
['a' 'b' 'c' 'd']

5     a
7     b
9     c
11    d
dtype: object
```

[note: we have specified the index as 5, 7, 9, 11]

Create a Series from dict

A **dict** (or **dictionary**) can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index. If index is passed, the values in data corresponding to the labels in the index will be pulled out.

Program 4: Create a series using dictionary

```
import pandas as pd
# Create the data of the series as a dictionary
dct = {'a' : 16, 'b' : 19, 'c' : 17, 'd' : 21}
srs_dct = pd.Series (dct)
print (srs_dct)
```

Output:

```
a    16
b    19
c    17
d    21
dtype: int64
```

Program 5: Create a series using dictionary and explicitly specify the index

```
import pandas as pd
# Create the data of the series as a dictionary
dct = {'a' : 16, 'b' : 19, 'c' : 17, 'd' : 21}
srs_dct = pd.Series (dct, index = ['c', 'a', 'b', 'd'])
print (srs_dct)
```

Output:

```
c    17
a    16
b    19
d    21
dtype: int64
```

Create a Series from Scalar

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index

Program 6: Create a series using a scaler

```
import pandas as pd
# Create the data of the series as a dictionary
srs_scl = pd.Series (5, index = [1, 2, 3, 4])
print (srs_scl)
```

Output:

```
1    5
2    5
3    5
4    5
dtype: int64
```

* * DataFrame * *

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Features of DataFrame:

- Potentially columns are of different types
- Size – Mutable
- Labeled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

pandas.DataFrame

A pandas DataFrame can be created using the following constructor:

```
pandas.DataFrame(data, index, columns, dtype, copy)
```

The parameters of the constructor are as follows:

Sr. No	Parameter & Description
1	**data:** data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.
2	**index:** For the row labels, the Index to be used for the resulting frame is Optional Default `np.arange(n)` if no index is passed.
3	**columns:** For column labels, the optional default syntax is - `np.arange(n).` This is only true if no index is passed.
4	**dtype:** Data type of each column.
5	**copy:** This command (or whatever it is) is used for copying of data, if the default is False.

A pandas DataFrame can be created using various inputs like:

- Lists
- dict
- Series
- Numpy ndarrays
- Another DataFrame

Create an Empty DataFrame

A basic DataFrame, which can be created is an Empty Dataframe.

Program 1: Create a basic DataFrame (Empty DataFrame)

```
import pandas as pd
dtf = pd.DataFrame ()
print (dtf)
```

Output:

```
Empty DataFrame
Columns: []
Index: []
```

Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists.

Program 2: Write a program to create a DataFrame from Lists

Example-1:

```
import pandas as pd
lst = [41, 31, 17, 51, 61]
dfl = pd.DataFrame (lst)
print (dfl)
```

Output:

```
    0
0  41
1  31
2  17
3  51
4  61
```

Example-2:

```
import pandas as pd
lst = [[41, 31], [17, 51], [61, 71]]
dfl = pd.DataFrame (lst)
print (dfl)
```

Output:

```
    0   1
0  41  31
1  17  51
2  61  71
```

Example-3:

```
import pandas as pd
lst = [[41, 31], [17, 51], [61, 71]]
dfl = pd.DataFrame (lst, index = ['a', 'b', 'c'])
print (dfl)
```

Output:

```
    0   1
a  41  31
b  17  51
```

```
c  61  71
```

Example-4:

```
import pandas as pd
lst = [['apple', 50], ['banana', 30], ['mango', 150],
['grapes', 60]]
dfl = pd.DataFrame (lst, index = ['a', 'b', 'c', 'd'], columns
= ['fruit', 'price'], dtype = float)
print (dfl)
```

Output:

```
    fruit  price
a   apple   50.0
b  banana   30.0
c   mango  150.0
d  grapes   60.0
```

Create a DataFrame from Dict of ndarrays / Lists

All the ndarrays must be of same length. If index is passed, then the length of the index should equal to the length of the arrays.

If no index is passed, then by default, index will be `range(n)`, where n is the array length.

Program 2: Write a program to create a DataFrame from Dict of ndarrays / Lists

Example-1:

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve',
'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print (df)
```

Output:

```
    Name  Age
0    Tom   28
1   Jack   34
2  Steve   29
3  Ricky   42
```

Example-2:

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve',
'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, columns = ['Age', 'Name'],
index=['Roll 1:','Roll 2:','Roll 3:','Roll 4:'])
print (df)
```

Output:

```
         Age   Name
Roll 1:   28    Tom
Roll 2:   34   Jack
Roll 3:   29  Steve
Roll 4:   42  Ricky
```

Create a DataFrame from Dict of Series

Dictionary of Series can be passed to form a DataFrame. The resultant index is the union of all the series indexes passed.

Program 3: Write a program to create a DataFrame from Dict of Series

```
import pandas as pd
srs = {'col-1' : pd.Series([5, 6, 8], index = ['a', 'b', 'c']),
       'col-2' : pd.Series([9, 7, 3], index = ['a', 'b', 'c'])}
df = pd.DataFrame (srs)
print (df)
```

Output:

```
   col-1  col-2
a      5      9
b      6      7
c      8      3
```

Importing different types of Datasets

Program 4: Write a program to import the dataset **kmeans_blobs.csv** located in the directory "C:\Users\KUNAL KUNDU\OneDrive\Desktop\Dataset\" (change the directory location as per your machine)

Example-1:

```
import pandas as pd
dst = pd.read_csv(r"C:\Users\KUNAL
KUNDU\OneDrive\Desktop\Dataset\kmeans_blobs.csv")
print (dst)
```

Output:

```
    ID       x       y  cluster
0    0  24.412  32.932        2
1    1  35.190  12.189        1
2    2  26.288  41.718        2
3    3   0.376  15.506        0
4    4  26.116   3.963        1
5    5  25.893  31.515        2
6    6  23.606  15.402        1
7    7  28.026  15.470        1
8    8  26.360  34.488        2
9    9  23.013  36.213        2
10  10  27.819  41.867        2
11  11  39.634  42.230        2
12  12  35.477  35.104        2
13  13  25.768   5.967        1
14  14  -0.684  21.105        0
15  15   3.387  17.810        0
```

Example-2:

```
import pandas as pd
path = r"C:\Users\KUNAL
KUNDU\OneDrive\Desktop\Dataset\kmeans_blobs.csv"
dst = pd.read_csv(path)
print (dst)
```

Output:

```
    ID       x       y  cluster
0    0  24.412  32.932        2
1    1  35.190  12.189        1
2    2  26.288  41.718        2
3    3   0.376  15.506        0
4    4  26.116   3.963        1
5    5  25.893  31.515        2
6    6  23.606  15.402        1
7    7  28.026  15.470        1
8    8  26.360  34.488        2
9    9  23.013  36.213        2
10  10  27.819  41.867        2
11  11  39.634  42.230        2
12  12  35.477  35.104        2
13  13  25.768   5.967        1
14  14  -0.684  21.105        0
15  15   3.387  17.810        0
```

* * Panel * *

Panel is Removed in more recent versions. In previous versions, it was a three-dimensional data structure, but its functionalities have been integrated into the **MultiIndex** levels of DataFrames, which allows handling higher-dimensional data more flexibly.