EXPERIMENT NO - 2

AIM - panels and basic functionality.

THEORY -

Python Pandas - Panel

The term **Panel data** is derived from econometrics and is partially responsible for the name pandas − **pan(el)-da(ta)**-s.

The names for the 3 axes are intended to give some semantic meaning to describing operations involving panel data. They are −

- **items** − axis 0, each item corresponds to a DataFrame contained inside.
- **major_axis** − axis 1, it is the index (rows) of each of the DataFrames.
- **minor_axis** − axis 2, it is the columns of each of the DataFrames.

pandas.Panel()

A Panel can be created using the following constructor −

```
pandas.Panel(data, items, major_axis, minor_axis, dtype, copy)
```

The parameters of the constructor are as follows −

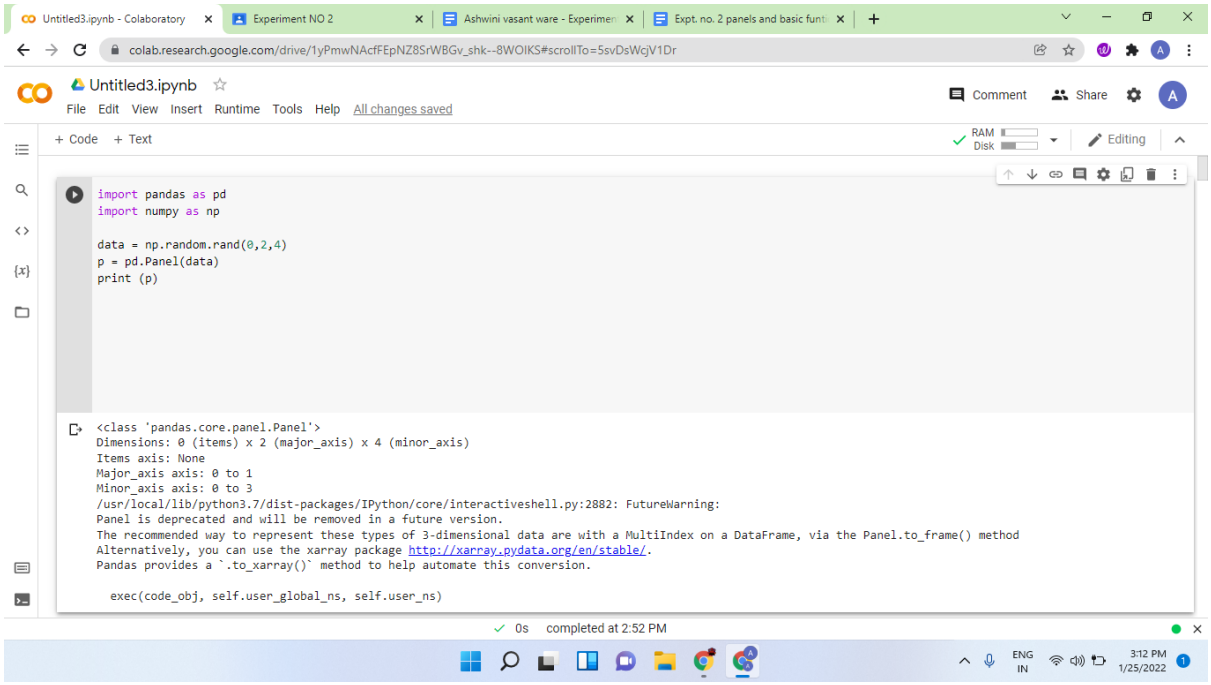
Parameter	Description
data	Data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame
items	axis=0
major_axis	axis=1
minor_axis	axis=2
dtype	Data type of each column
copy	Copy data. Default, **false**

Create Panel

A Panel can be created using multiple ways like −

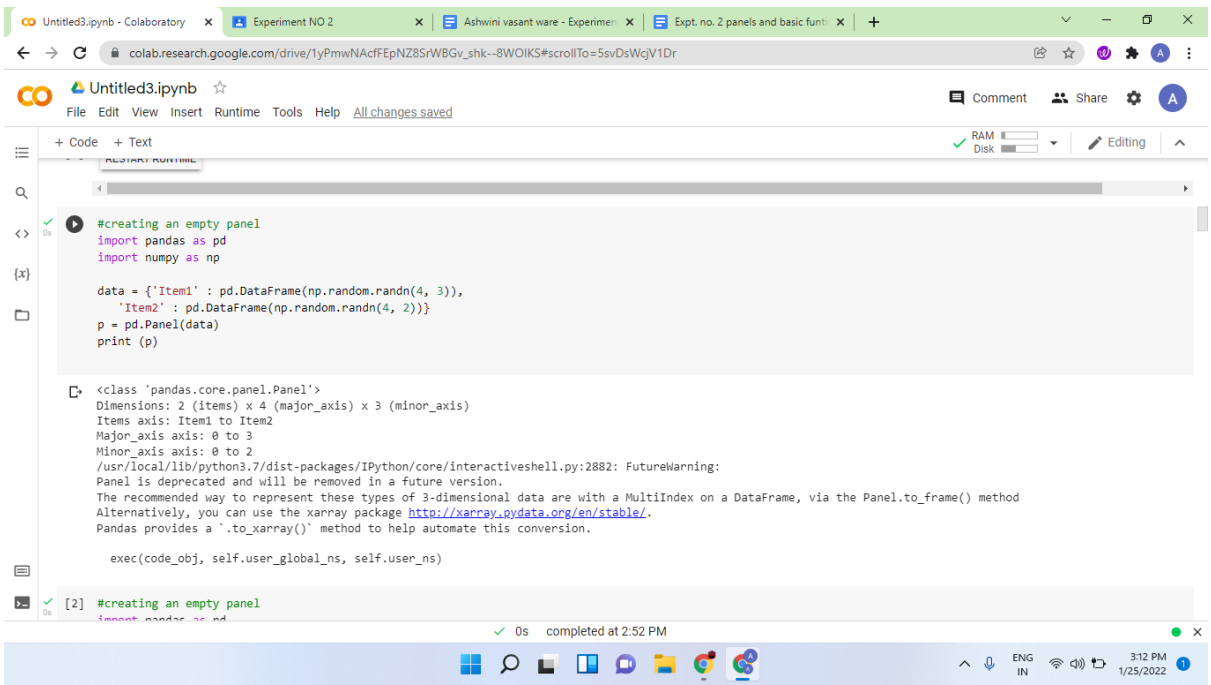
- From ndarrays
- From dict of DataFrames

From 3D ndarray



Note − Observe the dimensions of the empty panel and the above panel, all the objects are different.

From dict of DataFrame Objects



Create an Empty Panel

An empty panel can be created using the Panel constructor as follows −

```
#creating an empty panel
import pandas as pd
p = pd.Panel()
print (p)
```

Screenshot:

Selecting the Data from Panel

Select the data from the panel using −

- Items
- Major_axis
- Minor_axis

Using Items

```
# creating an empty panel
import pandas as pd
import numpy as np
data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
   'Item2' : pd.DataFrame(np.random.randn(4, 2))}
p = pd.Panel(data)
print (p['Item1'])
```

```
          0         1         2
0 -0.186489 -0.766594  0.104655
1  0.009048 -0.843827 -1.898634
2 -1.952249  0.059473 -0.257979
3  0.944721  0.572385 -0.561105
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
Panel is deprecated and will be removed in a future version.
The recommended way to represent these types of 3-dimensional data are with a MultiIndex on a DataFrame, via the Pane
Alternatively, you can use the xarray package http://xarray.pydata.org/en/stable/.
Pandas provides a `.to_xarray()` method to help automate this conversion.
```

We have two items, and we retrieved item1. The result is a DataFrame with 4 rows and 3 columns, which are the **Major_axis** and **Minor_axis** dimensions.

Using major_axis

Data can be accessed using the method **panel.major_axis(index)**.

```
[7]  # creating an empty panel
     import pandas as pd
     import numpy as np
     data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
        'Item2' : pd.DataFrame(np.random.randn(4, 2))}
     p = pd.Panel(data)
     print (p.major_xs(1))
```

```
       Item1     Item2
0  0.385775 -2.197001
1 -1.287173  0.199529
2 -0.506260       NaN
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
Panel is deprecated and will be removed in a future version.
The recommended way to represent these types of 3-dimensional data are with a MultiIndex on a DataFrame, via the Panel.to_frame() method
Alternatively, you can use the xarray package http://xarray.pydata.org/en/stable/.
Pandas provides a `.to_xarray()` method to help automate this conversion.

  exec(code_obj, self.user_global_ns, self.user_ns)
```

Using minor_axis

Data can be accessed using the method **panel.minor_axis(index).**

```
# creating an empty panel
import pandas as pd
import numpy as np
data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
   'Item2' : pd.DataFrame(np.random.randn(4, 2))}
p = pd.Panel(data)
print (p.minor_xs(1))
```

```
      Item1     Item2
0 -0.411117 -0.323368
1  0.405002  0.790631
2 -0.812568 -0.060681
3 -1.414392 -1.876026
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning:
Panel is deprecated and will be removed in a future version.
The recommended way to represent these types of 3-dimensional data are with a MultiIndex on a DataFrame, via the Panel.to_frame() method
Alternatively, you can use the xarray package http://xarray.pydata.org/en/stable/.
Pandas provides a `.to_xarray()` method to help automate this conversion.

  exec(code_obj, self.user_global_ns, self.user_ns)
```

Note − Observe the changes in the dimensions.

Python Pandas - Basic Functionality

By now, we learnt about the three Pandas DataStructures and how to create them. We will majorly focus on the DataFrame objects because of its importance in the real time data processing and also discuss a few other DataStructures.

Series Basic Functionality

Sr.No.	Attribute or Method & Description
1	**axes** Returns a list of the row axis labels

2	**dtype** Returns the dtype of the object.
3	**empty** Returns True if series is empty.
4	**ndim** Returns the number of dimensions of the underlying data, by definition 1.
5	**size** Returns the number of elements in the underlying data.
6	**values** Returns the Series as ndarray.
7	**head()** Returns the first n rows.
8	**tail()** Returns the last n rows.

Let us now create a Series and see all the above tabulated attributes operation.

Example

```
[9]  import pandas as pd
     import numpy as np

     #Create a series with 100 random numbers
     s = pd.Series(np.random.randn(4))
     print (s)
```

```
0    0.873887
1   -0.959885
2   -0.628112
3    2.345342
dtype: float64
```

axes

Returns the list of the labels of the series.

```
[10] import pandas as pd
     import numpy as np

     #Create a series with 100 random numbers
     s = pd.Series(np.random.randn(4))
     print ("The axes are:")
     print (s.axes)
```

```
The axes are:
[RangeIndex(start=0, stop=4, step=1)]
```

empty

Returns the Boolean value saying whether the Object is empty or not. True indicates that the object is empty.

```
import pandas as pd
import numpy as np

#Create a series with 100 random numbers
s = pd.Series(np.random.randn(4))
print ("Is the Object empty?")
print (s.empty)
```

```
Is the Object empty?
False
```

Returns the number of dimensions of the object. By definition, a Series is a 1D data structure, so it returns

```
[13] import pandas as pd
     import numpy as np

     #Create a series with 4 random numbers
     s = pd.Series(np.random.randn(4))
     print (s)

     print ("The dimensions of the object:")
     print (s.ndim)
```

```
0   -0.154731
1    0.279359
2   -0.550324
3   -0.459259
dtype: float64
The dimensions of the object:
1
```

size

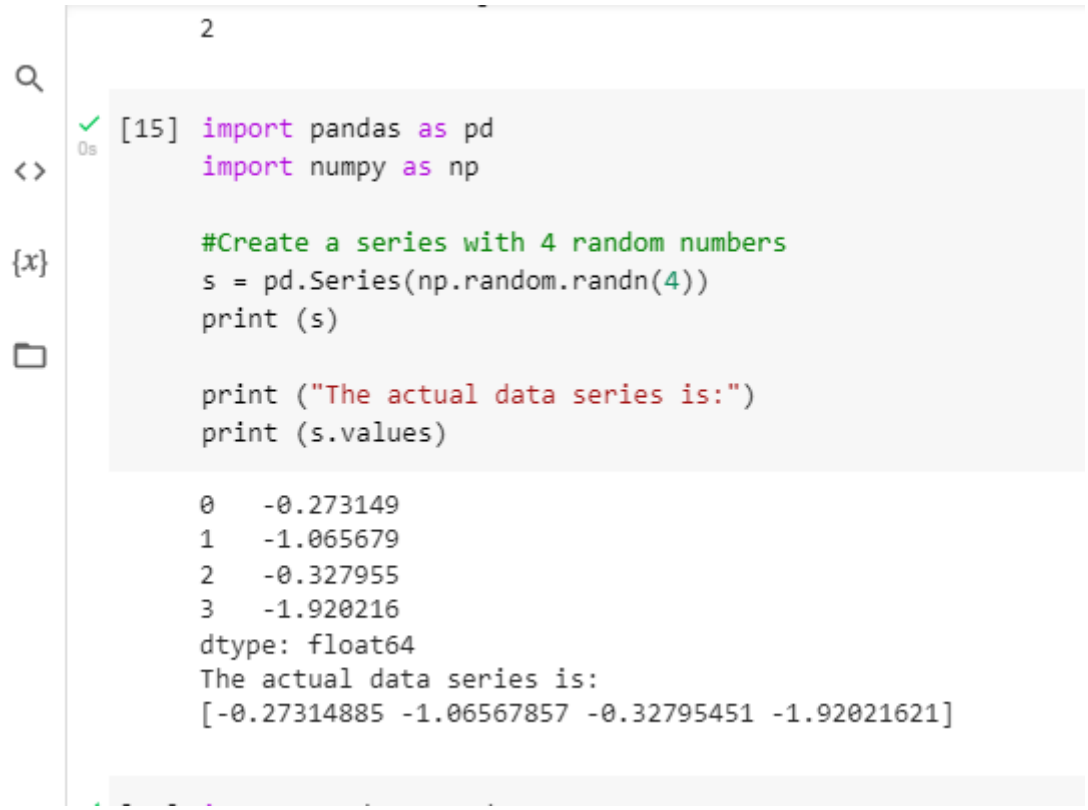
Returns the size(length) of the series.

```
import pandas as pd
import numpy as np

#Create a series with 4 random numbers
s = pd.Series(np.random.randn(2))
print (s)
print ("The size of the object:")
print (s.size)
```

```
0    0.014469
1    0.680633
dtype: float64
The size of the object:
2
```

values

Returns the actual data in the series as an array.



Head & Tail

To view a small sample of a Series or the DataFrame object, use the head() and the tail() methods.

head() returns the first **n** rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.

```
[16] import pandas as pd
     import numpy as np

     #Create a series with 4 random numbers
     s = pd.Series(np.random.randn(4))
     print ("The original series is:")
     print (s)

     print ("The first two rows of the data series:")
     print (s.head(2))
```

```
The original series is:
0    0.409497
1   -2.065677
2    0.789591
3   -0.882725
dtype: float64
The first two rows of the data series:
0    0.409497
1   -2.065677
dtype: float64
```

tail() returns the last **n** rows(observe the index values). The default number of elements to display is five, but you may pass a custom number.

```
[17] import pandas as pd
     import numpy as np

     #Create a series with 4 random numbers
     s = pd.Series(np.random.randn(4))
     print ("The original series is:")
     print (s)

     print ("The last two rows of the data series:")
     print (s.tail(2))
```

```
The original series is:
0    1.312196
1   -1.146876
2   -1.647552
3   -1.332377
dtype: float64
The last two rows of the data series:
2   -1.647552
3   -1.332377
dtype: float64
```

DataFrame Basic Functionality

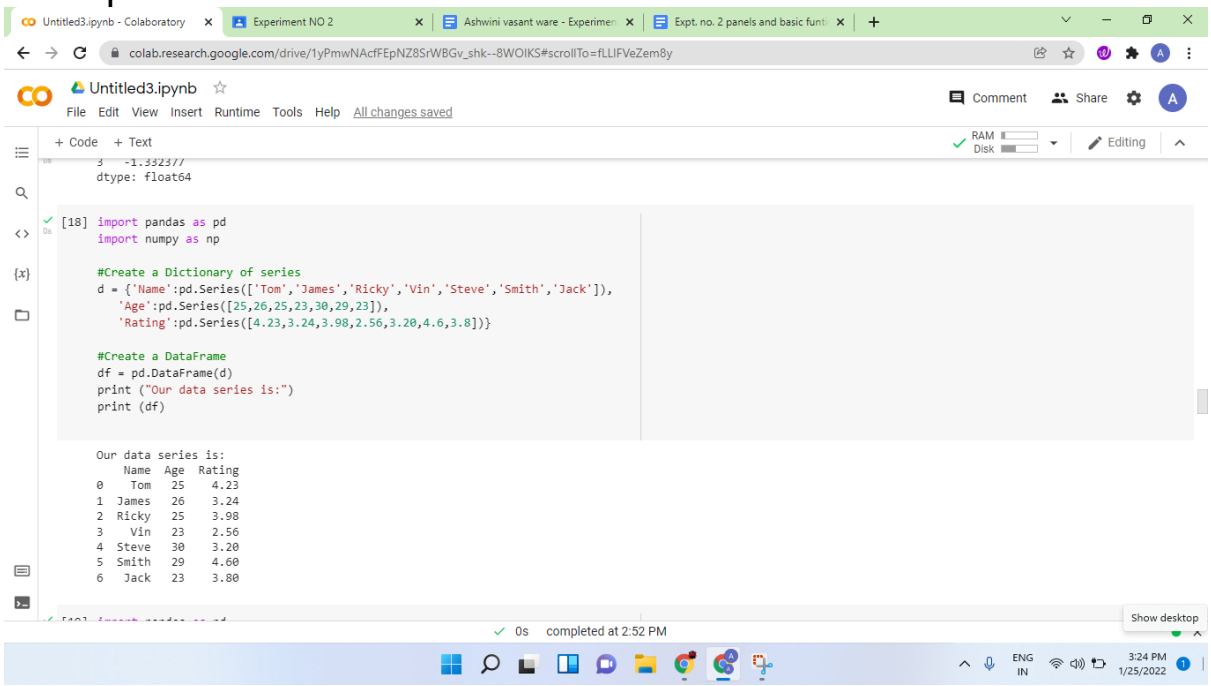
Let us now understand what DataFrame Basic Functionality is. The following tables lists down the important attributes or methods that help in DataFrame Basic Functionality.

Sr.No.	Attribute or Method & Description
1	**T** Transposes rows and columns.
2	**axes** Returns a list with the row axis labels and column axis labels as the only members.
3	**dtypes** Returns the dtypes in this object.

4	**empty** True if NDFrame is entirely empty [no items]; if any of the axes are of length 0.
5	**ndim** Number of axes / array dimensions.
6	**shape** Returns a tuple representing the dimensionality of the DataFrame.
7	**size** Number of elements in the NDFrame.
8	**values** Numpy representation of NDFrame.
9	**head()** Returns the first n rows.
10	**tail()** Returns last n rows.

Let us now create a DataFrame and see all how the above mentioned attributes operate.

Example



T (Transpose)

Returns the transpose of the DataFrame. The rows and columns will interchange.

```
[19] import pandas as pd
     import numpy as np

     # Create a Dictionary of series
     d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
        'Age':pd.Series([25,26,25,23,30,29,23]),
        'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

     # Create a DataFrame
     df = pd.DataFrame(d)
     print ("The transpose of the data series is:")
     print (df.T)

     The transpose of the data series is:
                0      1      2     3      4      5     6
     Name     Tom  James  Ricky   Vin  Steve  Smith  Jack
     Age       25     26     25    23     30     29    23
     Rating  4.23   3.24   3.98  2.56    3.2    4.6   3.8
```

axes

Returns the list of row axis labels and column axis labels.

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
   'Age':pd.Series([25,26,25,23,30,29,23]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Row axis labels and column axis labels are:")
print (df.axes)
```

```
Row axis labels and column axis labels are:
[RangeIndex(start=0, stop=7, step=1), Index(['Name', 'Age', 'Rating'], dtype='object')]
```

dtypes

Returns the data type of each column.

```
[21] import pandas as pd
     import numpy as np

     #Create a Dictionary of series
     d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
        'Age':pd.Series([25,26,25,23,30,29,23]),
        'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

     #Create a DataFrame
     df = pd.DataFrame(d)
     print ("The data types of each column are:")
     print (df.dtypes)
```

```
The data types of each column are:
Name       object
Age         int64
Rating    float64
```

✓ 0s completed at 2:52 PM

empty

Returns the Boolean value saying whether the Object is empty or not; True indicates that the object is empty.

```
[22] import pandas as pd
     import numpy as np

     #Create a Dictionary of series
     d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
        'Age':pd.Series([25,26,25,23,30,29,23]),
        'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

     #Create a DataFrame
     df = pd.DataFrame(d)
     print ("The data types of each column are:")
     print (df.dtypes)
```

```
The data types of each column are:
Name       object
Age         int64
Rating    float64
dtype: object
```

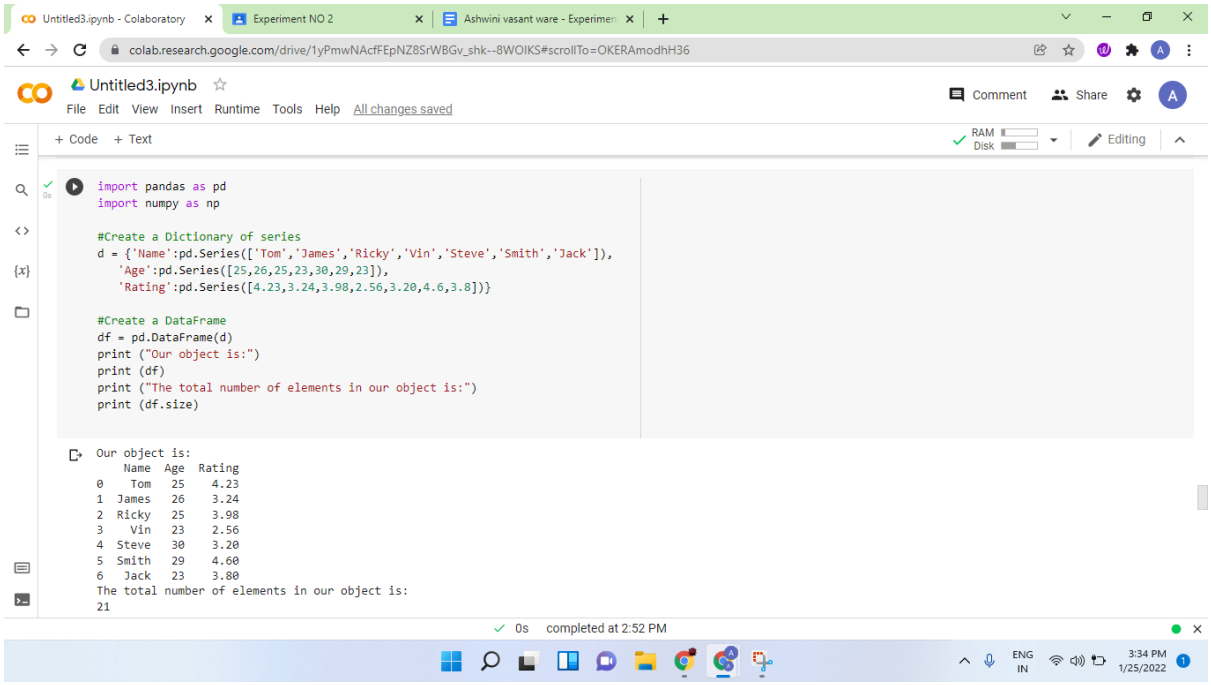
ndim

Returns the number of dimensions of the object. By definition, DataFrame is a 2D object.

```
[24] import pandas as pd
     import numpy as np

     #Create a Dictionary of series
     d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
        'Age':pd.Series([25,26,25,23,30,29,23]),
        'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

     #Create a DataFrame
     df = pd.DataFrame(d)
     print ("Our object is:")
     print (df)
     print ("The dimension of the object is:")
     print (df.ndim)
```

```
Our object is:
    Name  Age  Rating
0    Tom   25    4.23
1  James   26    3.24
2  Ricky   25    3.98
3    Vin   23    2.56
4  Steve   30    3.20
5  Smith   29    4.60
6   Jack   23    3.80
The dimension of the object is:
```

Shape



Returns a tuple representing the dimensionality of the DataFrame. Tuple (a,b), where a represents the number of rows and **b** represents the number of columns.

size

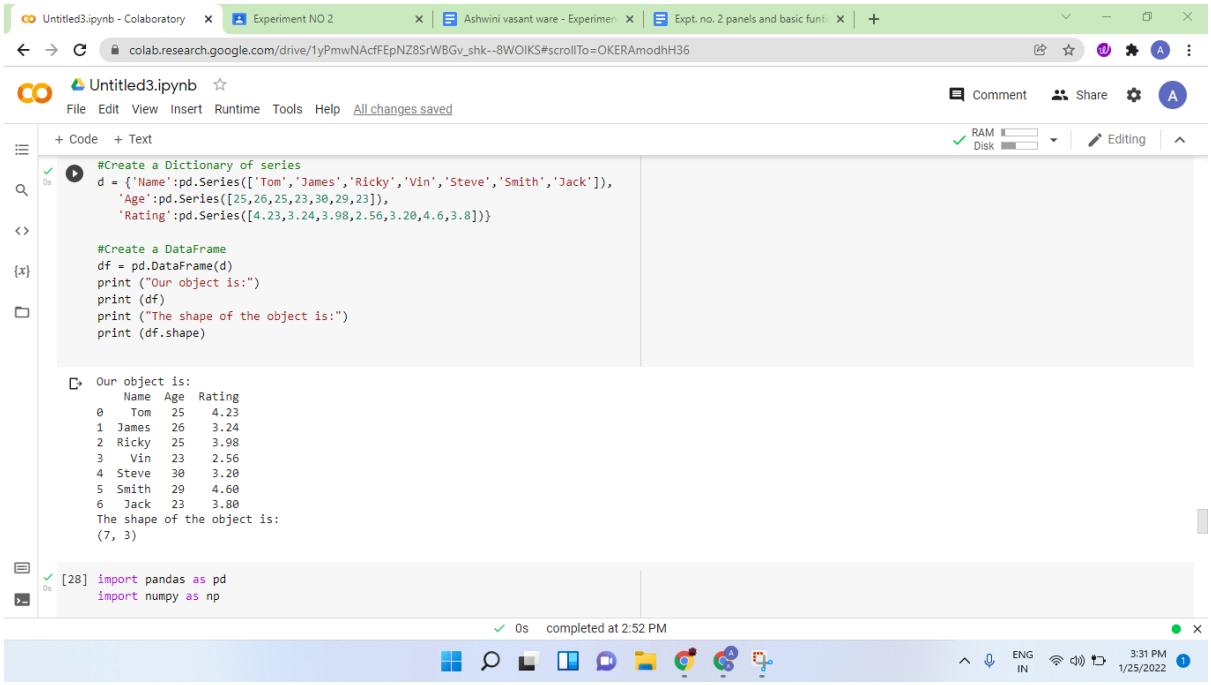
```
[26] import pandas as pd
     import numpy as np

     #Create a Dictionary of series
     d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
        'Age':pd.Series([25,26,25,23,30,29,23]),
        'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

     #Create a DataFrame
     df = pd.DataFrame(d)
     print ("Our object is:")
     print (df)
     print ("The total number of elements in our object is:")
     print (df.size)
```

```
Our object is:
    Name  Age  Rating
0    Tom   25    4.23
1  James   26    3.24
2  Ricky   25    3.98
3    Vin   23    2.56
4  Steve   30    3.20
5  Smith   29    4.60
6   Jack   23    3.80
The total number of elements in our object is:
21
```

Returns the number of elements in the DataFrame.

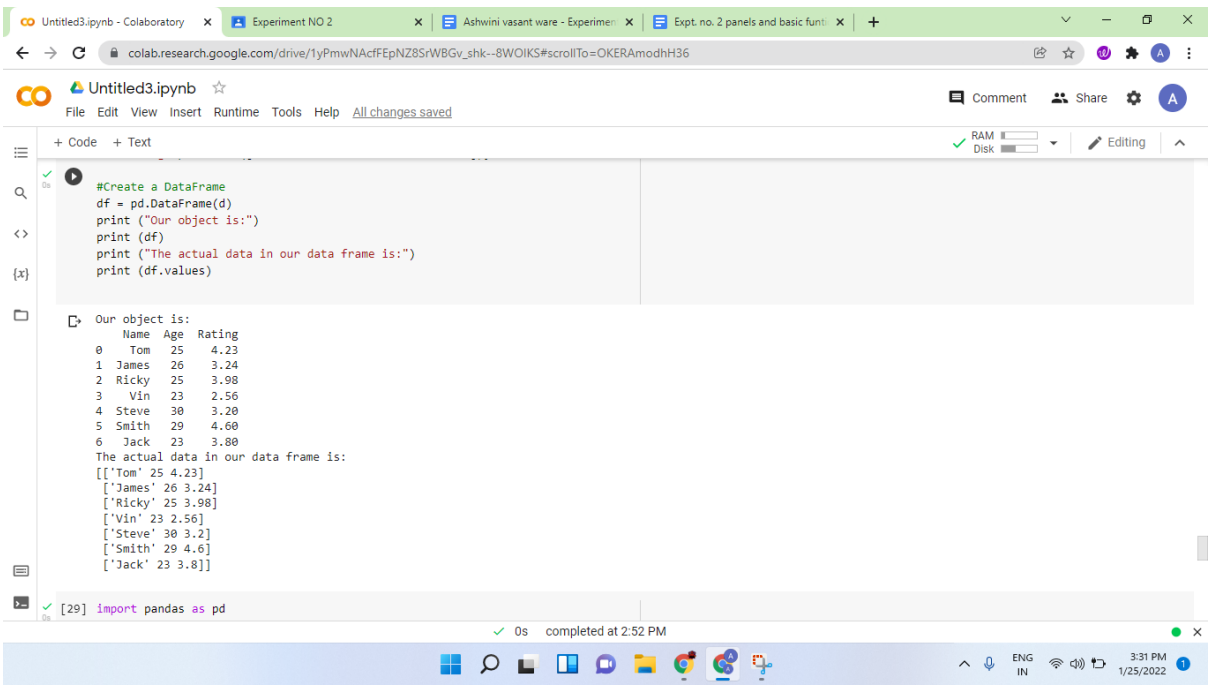
values

Returns the actual data in the DataFrame as an **NDarray.**

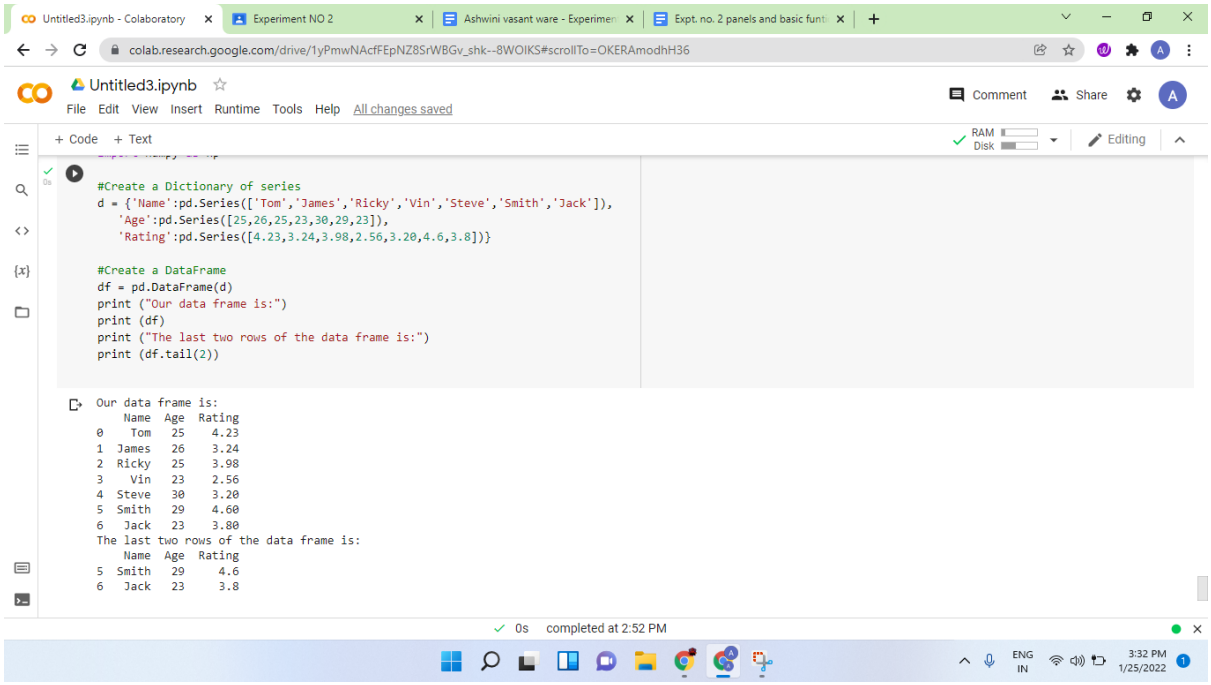


Head & Tail

To view a small sample of a DataFrame object, use the **head()** and tail() methods. **head()** returns the first **n** rows (observe the index values). The default number of elements to display is five, but you may pass a custom number.



tail() returns the last **n** rows (observe the index values). The default number of elements to display is five, but you may pass a custom number.


Untitled3.ipynb - Colaboratory
Experiment NO 2
Ashwini vasant ware - Experimen
Expt. no. 2 panels and basic funt
colab.research.google.com/drive/1yPmwNAcfFEpNZ8SrWBGv_shk--8WOIKS#scrollTo=OKERAmodhH36
Untitled3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
'Age':pd.Series([25,26,25,23,30,29,23]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}
#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data frame is:")
print (df)
print ("The last two rows of the data frame is:")
print (df.tail(2))
Our data frame is:
Name Age Rating
0 Tom 25 4.23
1 James 26 3.24
2 Ricky 25 3.98
3 Vin 23 2.56
4 Steve 30 3.20
5 Smith 29 4.60
6 Jack 23 3.80
The last two rows of the data frame is:
Name Age Rating
5 Smith 29 4.6
6 Jack 23 3.8
0s completed at 2:52 PM
ENG
IN
3:32 PM
1/25/2022