Experiment No: 11

Aim: Program to demonstrate Data Series using Pandas

Theory:

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

After the pandas have been installed into the system, you need to import the library. This module is generally imported as:

import pandas as pd

Here, pd is referred to as an alias to the Pandas.

Pandas generally provide two data structures for manipulating data, They are:

- **Series**
- **DataFrame**

Series:
Pandas Series is a one-dimensional labelled array capable of holding data of any type (integer, string, float, python objects, etc.).
The axis labels are collectively called indexes.
Pandas Series is nothing but a column in an excel sheet.

Creating an empty Series :
A basic series, which can be created is an Empty Series.

```
# import pandas as pd
import pandas as pd
# Creating empty series
ser = pd.Series()
print(ser)
```

Creating a series from array:
In order to create a series from array, we have to import a numpy module and have to use array() function.

```
# import pandas as pd
import pandas as  pd
# import numpy as np
import numpy as np
```

```
 # simple array
data =
  np.array(['g', 'e',
  'e', 'k', 's'])ser =
  pd.Series(data)
print(ser)
```

Creating a series from array with index :
In order to create a series from array with index, we have to provide index with samenumber of element as it is in array.

```
# import pandas as pd
import pandas as pd

# import numpy as np
import numpy as np

# simple array
data = np.array(['g', 'e', 'e', 'k', 's'])

# providing an index
ser = pd.Series(data, index =[10,
11, 12, 13, 14])print(ser)
```

Creating a series from Lists:
In order to create a series from list, we have to first create a list after that we can create aseries from list.

```
imp
  ort
  pan
  das
  as
  pd
  # a
  sim
  ple
  list
list = ['g', 'e', 'e', 'k', 's']
```

```
# create
series
form a
listser =
pd.Series
(list)
print(ser)
```

Creating a series from Dictionary:

In order to create a series from dictionary, we have to first create a dictionary after that wecan make a series using dictionary. Dictionary key are used to construct a index.

```
imp

ort

pand

as as

pd#

a

simp

le

dicti

onar

y

dict
 =
 {'G
 ee
 ks'
 :
 10,
 'for
 ' :
 20,
```

```
 'geeks' : 30}
# create series
from
dictionaryser
=
pd.Series(dict
)
print(ser)
```

Creating a series from Scalar value:
In order to create a series from scalar value, an index must be provided. The scalar valuewill be repeated to match the length of index.

```
import pandas as pd

import numpy as np

# giving a scalar value with index
ser = pd.Series(10, index =[0, 1, 2, 3, 4, 5])

print(ser)
```

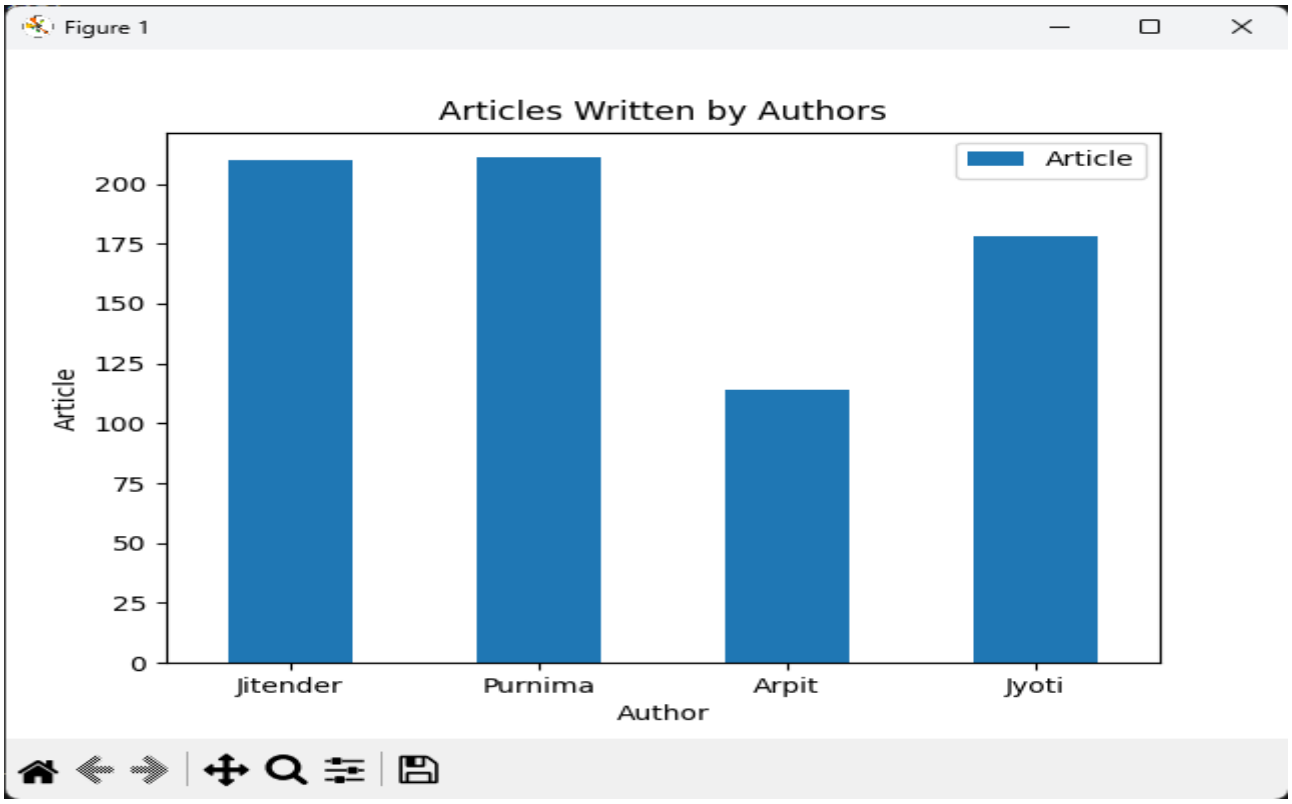
Creating a series using NumPy functions :
In order to create a series using numpy function, we can use different function of numpylike numpy.linspace(), numpy.random.radn().

```
# import
pandas and
numpy
import
pandas as
pd import
numpy as
np

# series with numpy linspace()
ser1 =
pd.Series(np.linspace(
3, 33, 3))print(ser1)

# series with numpy linspace()
ser2 =
pd.Series(np.linspace(1,
100, 10))print("\n",
ser2)
```

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt

author = ['Jitender', 'Purnima', 'Arpit', 'Jyoti']
article = [210, 211, 114, 178]
age = [21, 21, 24, 23]

auth_series = pd.Series(author)
article_series = pd.Series(article)
age_series = pd.Series(age)

frame = {'Author': auth_series, 'Article': article_series, 'Age': age_series}
result = pd.DataFrame(frame)

result.plot.bar(x='Author', y='Article', rot=0)  # Set rot=0 to keep x-axis labels horizontal
plt.xlabel('Author')
plt.ylabel('Article')
plt.title('Articles Written by Authors')
 plt.show()
```

Output :



Conclusion:

The experiment successfully showcased the utility of Pandas in handling and manipulatingdata series effectively. Through various operations and analyses, Pandas demonstrated its capability to streamline data management tasks, providing researchers and analysts with a powerful tool for data exploration and interpretation.