SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM: Working with different data formats using pandas a) Perform reading and writing data in text format using read_csv and read_table considering any online dataset in delimited format (CSV).

Program:

```
import pandas as pd

df=pd.read_csv('D:/pyth/courses.csv')

print(df)
```

Output:

```
courses   fees duration

0    html  10000   10days

1       c  20000   20days

2    java  30000   30days
```

```
import pandas as pd

df=pd.read_csv('D:/pyth/courses.csv',index_col='courses')

print(df)
```

Output:

```
              fees duration

courses

html     10000   10days

c        20000   20days

java     30000   30days
```

```
df=pd.read_csv('D:/pyth/courses.csv',header=None,skiprows=2)

print(df)
```

Output:

```
      0     1      2
0     c  20000  20days
1  java  30000  30days
```

```
columns=['subjects','c_fee','c_duration']
```

```
df=pd.read_csv('D:/pyth/courses.csv',names=columns,skiprows=2)
print(df)
```

Output:

```
subjects  c_fee c_duration
0       c  20000     20days
1     java  30000     30days
```

```
df=pd.read_table('D:/pyth/courses.csv',delimiter=',')
print(df)
```

Output:

```
courses   fees duration
0    html  10000   10days
1       c  20000   20days
2    java  30000   30days
```

```
df=pd.read_table('D:/pyth/courses.csv',delimiter=',',index_col=0,nrows=2)
print(df)
```

Output:

```
        fees duration
courses
```

```
html     10000   10days

c        20000   20days
```

```
df=pd.read_table('D:/pyth/courses.csv',delimiter=',',index_col=0,engine='python',skipfooter=2)

print(df)
```

Output:

```
            fees duration

courses

html     10000   10days
```

```
import pandas as pd

import numpy as np

tech={'Courses':["Spark","Hadoop","Python"],'Fee':[22000,np.nan,24000],'Duration':['30days','55days',np.nan]}

df=pd.DataFrame(tech)

print(df)
```

Output:

```
Courses      Fee Duration

0   Spark  22000.0   30days

1  Hadoop      NaN   55days

2  Python  24000.0      NaN
```

```
import pandas as pd

cols=['Name','Surname','DoB','Dept']

df=pd.read_fwf('D:/pyth/samp.txt',header=None,widths=[4,7,3,10],names=cols)

print(df)
```

Output:

```
Name Surname  DoB  Dept

0  kkk   20 it  NaN   NaN
```

AIM:b) Perform reading, writing and parsing data in JSON (Javascript Object Notation) format using read_json.

Program:

```
import pandas as pd

import json

s='{"col1":{"row1":1,"row2":2,"row3":3},"col2":{"row1":"x","row2":"y","row3":"z"}}'

df=pd.read_json(s)

print(df)
```

Output:

```
col1 col2

row1    1    x

row2    2    y

row3    3    z
```

```
df=pd.DataFrame([1,2,3])

df.to_json('D:/pyth/example.json')
```

Output:

```
{"0":{"0":1,"1":2,"2":3}}
```

```
data=[['Axel',32],['Alice',26],['Alex',45]]

df=pd.DataFrame(data,columns=['Name','Age'])

df.to_json('D:/pyth/example1.json')
```

Output:

```
{"Name":{"0":"Axel","1":"Alice","2":"Alex"},"Age":{"0":32,"1":26,"2":45}}
```

```
df=pd.read_json('D:/pyth/example.json')

print(df)
```

Output:

```
   0
0  1
1  2
2  3
```

```
json_str='{"courses":{"r1":"Spark"},"Fee":{"r1":"25000"},"Duration":{"r1":"50days"}}'

df=pd.read_json(json_str)

print(df)
```

Output:

```
courses    Fee Duration
r1   Spark  25000   50days
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM: c) Perform reading and writing of Microsoft Excel Files (xslx) using read_excel.

Program:

```
import pandas as pd

df = pd.read_excel('D:/pyth/courses1.xlsx')

print(df)
```

Output:

```
courses   fees duration

0    html  10000   10days

1       c  20000   20days

2    java  30000   30days
```

```
columns = ['courses','course_fee','course_duration']

df2 = pd.read_excel('D:/pyth/courses1.xlsx',header=None, names = columns)

print(df2)
```

Output:

```
courses course_fee course_duration

0  courses       fees       duration

1     html      10000         10days

2        c      20000         20days

3     java      30000         30days
```

```
df2 = pd.read_excel(‘D:/pyth/courses1.xlsx’,

index_col=0)

print(df2)
```

Output:

```
         fees duration
courses
html     10000   10days
c       20000   20days
java     30000   30days
```

```
import pandas as pd
import numpy as np
technologies = ['Spark','Pandas','Java','Python', 'PHP']
fee = [25000,20000,15000,15000,18000]
duration = ['50 Days','35 Days',np.nan,'30 Days','30 Days']
discount = [2000,1000,800,500,800]
columns=['Courses','Fee','Duration','Discount']
df = pd.DataFrame(list(zip(technologies,fee,duration,discount)), columns=columns)
print(df)
```

Output:

```
   Courses   Fee Duration  Discount
0   Spark  25000  50 Days      2000
1  Pandas  20000  35 Days      1000
2    Java  15000      NaN       800
3  Python  15000  30 Days       500
4     PHP  18000  30 Days       800
```

```
df1 = pd.DataFrame([['a','b'],['c','d']],
index=['row1','row2'],
columns=['col1','col2'])
df1.to_excel('D:/pyth/output.xlsx')
```

Output:

```
         col1      col2
row1     a            b
```

AIM: 5a)Interacting with Web APIs and Databases a) Predict the last 30 GitHub issues for pandas using request and response object‘s json method. Move the extracted data to DataFrame and extract fields of interest.

Programs:

```
import requests

import pandas as pd

resp=requests.get('https://reqres.in/api/users')

resp_dict=resp.json()

#print(resp_dict)

df=pd.DataFrame(resp_dict.get('data'))

print(df)
```

Output:

```
   id           email            first_name last_name                    avatar

0  1  george.bluth@reqres.in     George     Bluth     https://reqres.in/img/faces/1-image.jpg

1  2   janet.weaver@reqres.in     Janet      Weaver   https://reqres.in/img/faces/2-image.jpg

2  3   emma.wong@reqres.in        Emma       Wong     https://reqres.in/img/faces/3-image.jpg

3  4   eve.holt@reqres.in         Eve        Holt     https://reqres.in/img/faces/4-image.jpg

4  5   charles.morris@reqres.in   Charles    Morris   https://reqres.in/img/faces/5-image.jpg

5  6   tracey.ramos@reqres.in     Tracey     Ramos    https://reqres.in/img/faces/6-image.jpg
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:-	**ROLLNO:-**	**SECTION:-** A

AIM: Data Cleaning and Preparation

a) Perform data cleaning by creating a DataFrame and identifying missing data using NA(Not Available) handling methods, filter out missing data using dropna function, fill the missing data using fillna function and remove duplicates using duplicated and drop_duplicates functions.

Programs:

```
import pandas as pd

import numpy as np

dict={'First Score':[100,90,np.nan,95],

    'Second Score':[30,45,56,np.nan],

    'Third Score':[np.nan,40,80,98]}

df=pd.DataFrame(dict)

print(df.isnull())

print(df.notnull())

print(df.fillna(0))

print(df.dropna())

print(df.dropna(how='all'))

print()

dfd=pd.DataFrame({'brand':['yum yum','yum yum','Indomie','Indomie','Indomie'],

            'style':['cup','cup','cup','pack','pack'],

            'rating':[4,4,3.5,15,5]})

print(dfd.drop_duplicates())

print(dfd.duplicated())
```

OUTPUT:

```
First Score  Second Score  Third Score

0       False        False        True

1       False        False       False

2        True        False       False

3       False         True       False

   First Score  Second Score  Third Score

0        True         True       False

1        True         True        True

2       False         True        True

3        True        False        True

   First Score  Second Score  Third Score

0       100.0         30.0         0.0

1        90.0         45.0        40.0

2         0.0         56.0        80.0

3        95.0          0.0        98.0

   First Score  Second Score  Third Score

1        90.0         45.0        40.0

   First Score  Second Score  Third Score

0       100.0         30.0         NaN

1        90.0         45.0        40.0

2         NaN         56.0        80.0

3        95.0          NaN        98.0


     brand style  rating
```

```
0  yum yum   cup     4.0

2  Indomie   cup     3.5

3  Indomie  pack    15.0

4  Indomie  pack     5.0

0    False

1     True

2    False

3    False

4    False

dtype: bool
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:-	**ROLLNO:-**	**SECTION:-** A

AIM: Perform data transformation by modifying set of values using map and replace method and create transformed version of original dataset without modification using rename method.

Programs:

```
*import numpy as np

import pandas as pd

data=pd.DataFrame(np.arange(12).reshape((3,4)),index=['Ohio','Colorado','NewYork'],columns=['one','two','three','four'])

print(data)

data.rename(index=str.title,columns=str.upper)
```

output:

```
one  two  three  four

Ohio       0    1      2     3

Colorado    4    5     6     7

New York    8    9     10    11

ONE      TWO     THREE   FOUR

Ohio     0          1          2          3

Colorado            4          5          6          7

New York            8          9          10         11
```

```
import pandas as pd

rankings={'test':['India','South Africa','England','New Zealand','Australia'],

          'odi':['England','India','New Zealand','South Africa','Pakistan'],
         't20':['Pakistan','India','Australia','England','New Zealand']}
rankings_pd=pd.DataFrame(rankings)
print(rankings_pd)
rankings_pd.rename(columns={'test':'Test'},inplace=True)
print("\n After modifying first column:\n",rankings_pd.columns)
```

output:

```
          test          odi         t20
0        India      England     Pakistan
1  South Africa        India       India
2      England   New Zealand    Australia
3   New Zealand  South Africa      England
4    Australia      Pakistan  New Zealand
 After modifying first column:
 Index(['Test', 'odi', 't20'], dtype='object')
```

```
*import pandas as pd
rankings={'test':['India','South Africa','England','New Zealand','Australia'],
         'odi':['England','India','New Zealand','South Africa','Pakistan'],
         't20':['Pakistan','India','Australia','England','New Zealand']}
rankings_pd=pd.DataFrame(rankings)
print(rankings_pd.columns)
rankings_pd.rename(columns={'test':'TEST','odi':'ODI','t20':'T20'},inplace=True)
print(rankings_pd.columns)
```

output:

```
Index(['test', 'odi', 't20'], dtype='object')
Index(['TEST', 'ODI', 'T20'], dtype='object')
```

```
*import pandas as pd
rankings={'test':['India','South Africa','England','New Zealand','Australia'],
        'odi':['England','India','New Zealand','South Africa','Pakistan'],
        't20':['Pakistan','India','Australia','England','New Zealand']}
rankings_pd=pd.DataFrame(rankings)
print(rankings_pd.columns)
rankings_pd.columns=['TEST','ODI','T20']
print(rankings_pd.columns)
```

output:

```
Index(['test', 'odi', 't20'], dtype='object')
Index(['TEST', 'ODI', 'T20'], dtype='object')
```

```
*import pandas as pd
rankings={'test':['India','South Africa','England','New Zealand','Australia'],
        'odi':['England','India','New Zealand','South Africa','Pakistan'],
        't20':['Pakistan','India','Australia','England','New Zealand']}
rankings_pd=pd.DataFrame(rankings)
print(rankings_pd.columns)
rankings_pd.set_axis(['A','B','C'])
print(rankings_pd.columns)
```

```
*import pandas as pd
rankings={'test':['India','South Africa','England','New Zealand','Australia'],
        'odi':['England','India','New Zealand','South Africa','Pakistan'],
        't20':['Pakistan','India','Australia','England','New Zealand']}
```

```
rankings_pd=pd.DataFrame(rankings)

print(rankings_pd.columns)

rankings_pd=rankings_pd.add_prefix('col_')

rankings_pd.add_suffix('_1')

rankings_pd.head()
```

output:

```
Index(['test', 'odi', 't20'], dtype='object')

col_testcol_odi col_t20

0          India     EnglandPakistan

1          South Africa     India     India

2          EnglandNew Zealand     Australia

3          New Zealand     South Africa     England

4          Australia          Pakistan          New Zealand
```

```
*import pandas as pd

rankings={'test':['India','South Africa','England','New Zealand','Australia'],

        'odi':['England','India','New Zealand','South Africa','Pakistan'],

        't20':['Pakistan','India','Australia','England','New Zealand']}

rankings_pd=pd.DataFrame(rankings)

print(rankings_pd.columns)

rankings_pd.columns=rankings_pd.columns.str.replace('test','Col_TEST')

rankings_pd.columns=rankings_pd.columns.str.replace('odi','Col_ODI')

rankings_pd.columns=rankings_pd.columns.str.replace('t20','Col_T20')

rankings_pd.head()
```

output:

```
Index(['test', 'odi', 't20'], dtype='object')

Col_TEST        Col_ODI         Col_T20

0       India   EnglandPakistan

1       South Africa    India   India

2       EnglandNew Zealand      Australia

3       New Zealand     South Africa    England

4       Australia       Pakistan        New Zealand
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:-	**ROLLNO:-**	**SECTION:-** A

AIM: Create a DataFrame with normally distributed data using random sampling and detect possible outliers.

Programs:

```
*import pandas as pd

import numpy as np

data=pd.DataFrame(np.random.randn(8,4))

print(data)

print()

print("describing data:")

print(data.describe())

print("find values in one of the columns exceeding 3 in absolute value:")

col=data[2]

print(col)

print(col[np.abs(col)>3])

print()

print("""np.sign(data) produces 1 and -1 values based on whether the values in data are positive or negative:""")

print(np.sign(data).head())
```

output

```
          0         1         2         3

0  1.397417 -0.915911 -1.868478  1.222061

1 -0.799012 -0.615517 -0.921358  0.144931
```

```
2 -0.649373 -0.207782 -0.027464 -0.566833

3  0.991145 -0.826265 -0.054891  0.921313

4  1.225472  1.234326 -1.782433 -0.367189

5 -1.003251 -0.731320  0.321124 -0.792045

6  1.673949 -0.154877  0.060318  1.285203

7 -0.565810 -0.399903 -0.013440  0.480208
```

describing data:

```
             0        1        2        3

count  8.000000  8.000000  8.000000  8.000000

mean   0.283817 -0.327156 -0.535828  0.290956

std    1.132719  0.689688  0.872680  0.815197

min   -1.003251 -0.915911 -1.868478 -0.792045

25%   -0.686783 -0.755056 -1.136627 -0.417100

50%    0.212667 -0.507710 -0.041178  0.312569

75%    1.268458 -0.194556  0.005000  0.996500

max    1.673949  1.234326  0.321124  1.285203
```

find values in one of the columns exceeding 3 in absolute value:

```
0   -1.868478

1   -0.921358

2   -0.027464

3   -0.054891

4   -1.782433

5    0.321124

6    0.060318
```

```
7   -0.013440
Name: 2, dtype: float64
Series([], Name: 2, dtype: float64)
```

np.sign(data) produces 1 and -1 values based on whether the values in data are positive or negative:

```
     0    1    2    3
0  1.0 -1.0 -1.0  1.0
1 -1.0 -1.0 -1.0  1.0
2 -1.0 -1.0 -1.0 -1.0
3  1.0 -1.0 -1.0  1.0
4  1.0  1.0 -1.0 -1.0
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Autonomous)
Department of Information Technology
III B. Tech – I Semester
(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM: 7. Data Wrangling a) Perform hierarchical indexing by creating a series with a list of lists (or arrays) as the index, select subsets of data at outer and inner levels using partial indexing.

Programs:

```
import numpy as np

import pandas as pd

lt = [["bar", "bar", "baz", "baz", "foo", "foo", "qux", "qux"],

      ["one", "two", "one", "two", "one", "two", "one", "two"]]

tuples = list(zip(*lt)) #adding two lists

index = pd.MultiIndex.from_tuples(tuples, names=["first", "second"])

s = pd.Series(np.random.randn(8), index=index)

print(s,"\n")

print("partial indexing")

p=s.loc['bar']

print("outer level\n",p)

q=s.loc['bar','two']

print("\ninner level\n",q)
```

OUTPUT:

```
first  second
bar    one       0.954755
       two       0.528785
baz    one      -0.380499
       two       0.974536
foo    one       1.229188
```

```
       two     -1.041887
qux   one      0.362889
       two     -0.037661
dtype: float64

partial indexing
outer level
 second
one    0.954755
two    0.528785
dtype: float64

inner level
 0.5287849179340118
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM: b) Rearrange the tabular data with hierarchical indexing using unstack and stack method.

Programs:

```
import pandas as pd

import numpy as np

#Usual Method of indexing

index = [('California', 2000), ('California', 2010),

        ('New York', 2000), ('New York', 2010),

        ('Texas', 2000), ('Texas', 2010)]

populations = [33871648, 37253956,

             18976457, 19378102,

             20851820, 25145561]

pop = pd.Series(populations, index=index)

print(pop)
```

Output:

```
(California, 2000)    33871648

(California, 2010)    37253956

(New York, 2000)      18976457

(New York, 2010)      19378102

(Texas, 2000)         20851820

(Texas, 2010)         25145561

dtype: int64
```

#Pandas MultiIndex

```
pop[('California', 2010):('Texas', 2000)]

pop[[i for i in pop.index if i[1] == 2010]]

index = pd.MultiIndex.from_tuples(index)

print(index)

pop = pop.reindex(index)

print(pop)
```

Output:

```
MultiIndex([('California', 2000),('California', 2010), ( 'New York', 2000), ( 'New York', 2010),

('Texas',2000), ('Texas', 2010)], )

California    2000    33871648

              2010     37253956

New York    2000    18976457

             2010    19378102

Texas       2000      20851820

            2010      25145561

dtype: int64
```

#The unstack() method

```
pop[:, 2010]

pop_df = pop.unstack()

print(pop_df)
```

Output:

```
              2000        2010

California    33871648  37253956

New York    18976457  19378102

Texas         20851820  25145561
```

#the stack() method

```
pop_df.stack()

pop_df = pd.DataFrame({'total': pop, 'under18': [9267089, 9284094,4687374, 4318033,5906301,

6879014]})

print(pop_df)
```

Output:

```
                    total     under18

California 2000  33871648  9267089

           2010  37253956  9284094

New York   2000  18976457  4687374

           2010  19378102  4318033

Texas      2000  20851820  5906301

           2010  25145561  6879014
```

```
#compute the fraction of people under 18 by year, given the above data:

df = pd.DataFrame(np.random.rand(4, 2), index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],

           columns=['data1', 'data2'])

print(df)
```

Output:

```
       data1      data2

a 1   0.352333   0.805914

  2   0.840838   0.374076

b 1   0.685044   0.810710

  2   0.471645   0.162683
```

#MultiIndex by default:

```
data = {('California', 2000): 33871648,('California', 2010): 37253956, ('Texas', 2000): 20851820,
('Texas', 2010): 25145561, ('New York', 2000): 18976457, ('New York', 2010): 19378102}

pd.Series(data)

pd.MultiIndex.from_arrays([['a', 'a', 'b', 'b'], [1, 2, 1, 2]])

pd.MultiIndex.from_tuples([('a', 1), ('a', 2), ('b', 1), ('b', 2)])

pd.MultiIndex.from_product([['a', 'b'], [1, 2]])

pop.index.names = ['state', 'year']

print(pop)
```

Output:

```
  state      year

California  2000   33871648

            2010   37253956

New York    2000   18976457

            2010   19378102

Texas       2000   20851820

            2010   25145561

dtype: int64
```

```
# hierarchical indices and columns

index = pd.MultiIndex.from_product([[2013, 2014], [1, 2]],names=['year', 'visit'])

columns = pd.MultiIndex.from_product([['Bob', 'Guido', 'Sue'], ['HR', 'Temp']],

 names=['subject', 'type'])

data = np.round(np.random.randn(4, 6), 1)

data[:, ::2] *= 10

data += 37

health_data = pd.DataFrame(data, index=index, columns=columns)

print(health_data)
```

Output:

```
subject     Bob       Guido       Sue

type        HR   Temp   HR   Temp   HR   Temp

year visit

2013 1      34.0   38.4   45.0   37.7   51.0   36.6

        2      43.0   36.6   20.0   37.1   35.0   36.5

2014 1      42.0   38.2   49.0   36.4   25.0   37.3

        2      48.0   39.2   36.0   38.2   18.0   37.1
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Autonomous)
Department of Information Technology
III B. Tech – I Semester
(20BT50532) PYTHON FOR DATA SCIENCE LAB

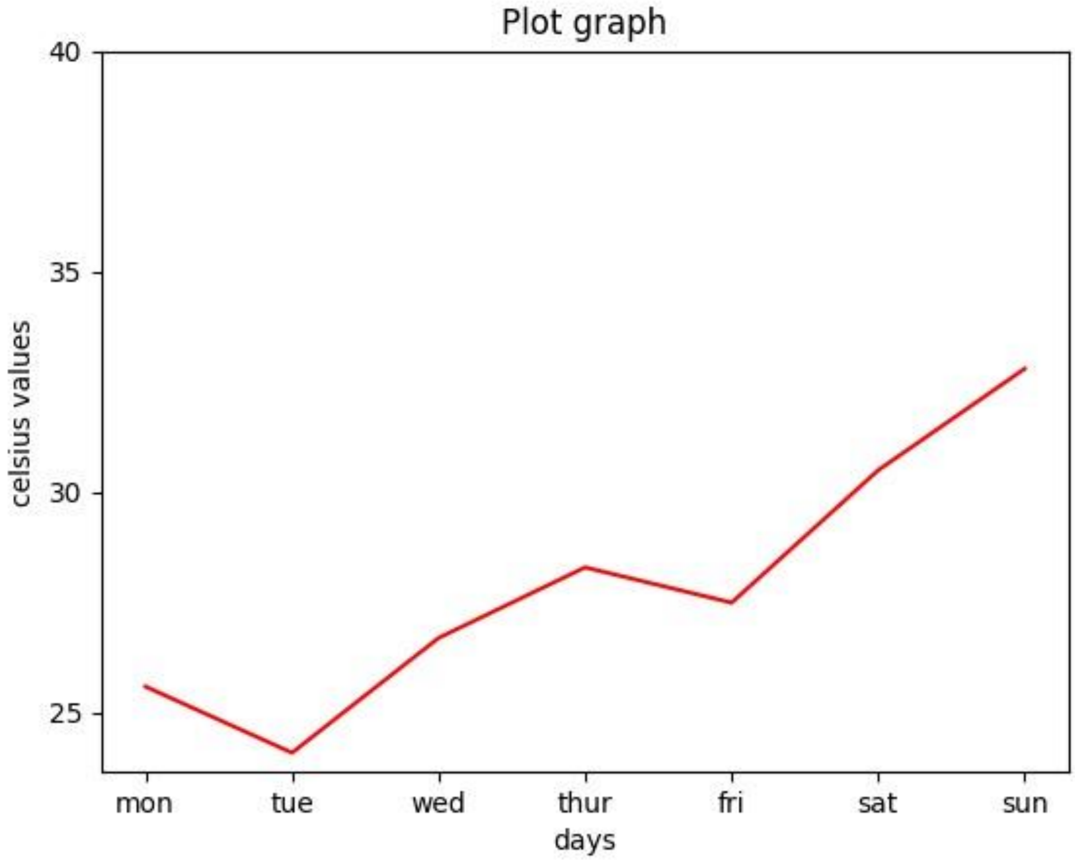
NAME:-	**ROLLNO:-**	**SECTION:-** A

AIM: 8a) Create a Line Plot by setting the title, axis labels, ticks, ticklabels, annotations on subplots and save

Programs:

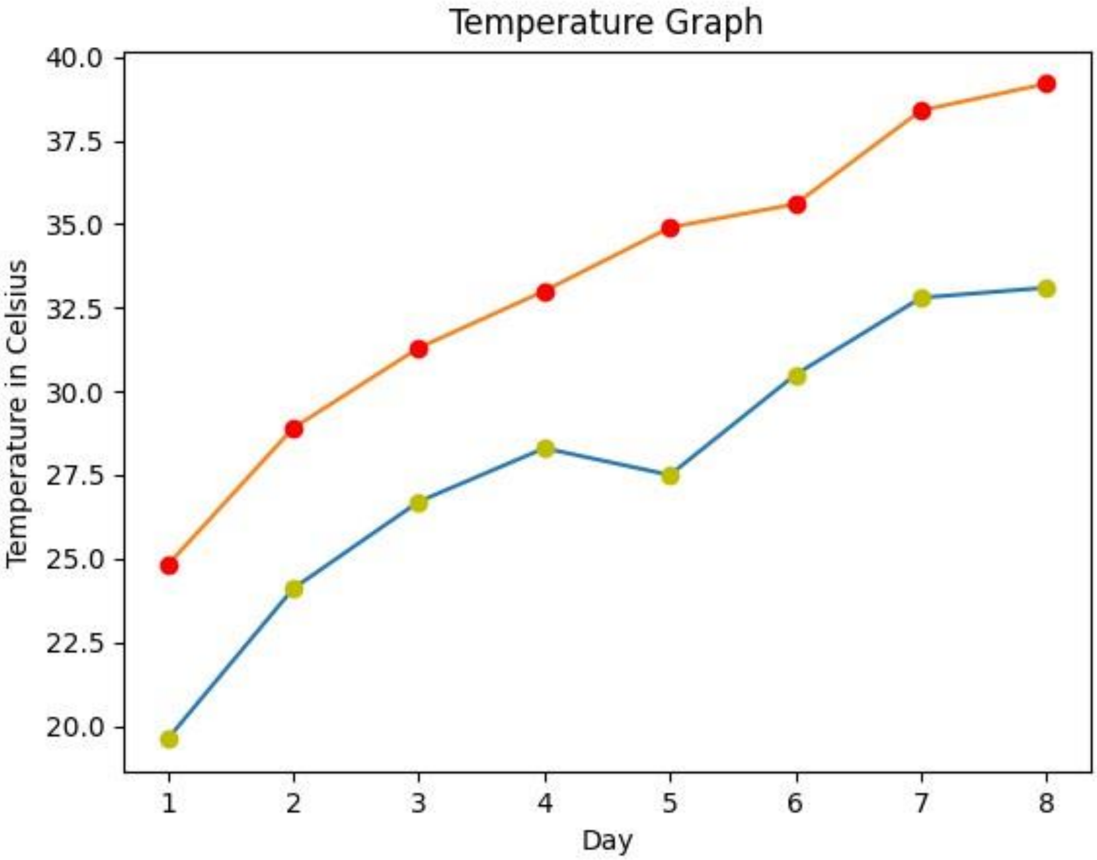
```
import matplotlib.pyplot as plt

import numpy as pd

days=list(range(1,8))

celsius=[25.6,24.1,26.7,28.3,27.5,30.5,32.8]

ax=plt.axes()

plt.xlabel("days")

plt.ylabel("celsius values")

plt.title("Plot graph")

plt.plot(days,celsius,color="red")

ax.set_yticks([25,30,35,40])

ax.set_xticks(days)

ax.set_xticklabels(["mon","tue","wed","thur","fri","sat","sun"])

plt.show()
```

Output:



```
import matplotlib.pyplot as plt

import pandas as pd

days = list(range(1,9))

celsius_min = [19.6, 24.1, 26.7, 28.3, 27.5, 30.5, 32.8, 33.1]

celsius_max = [24.8, 28.9, 31.3, 33.0, 34.9, 35.6, 38.4, 39.2]

fig, ax = plt.subplots()

ax.set(xlabel='Day',ylabel='Temperature in Celsius',title='Temperature Graph')

ax.plot(days, celsius_min,days, celsius_min,"oy",days, celsius_max,days, celsius_max, "or")

ax.set_xticks(days)

plt.show()
```

Output:


Temperature Graph
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
Temperature in Celsius
1
2
3
4
5
6
7
8
Day

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

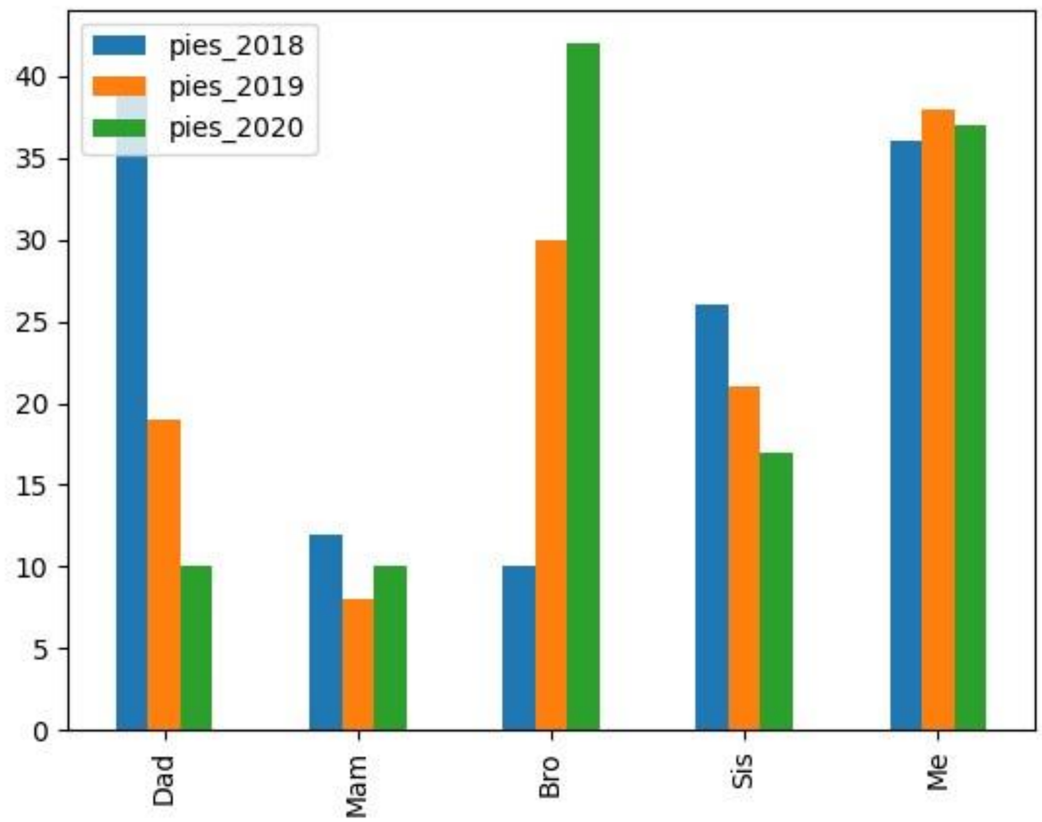
AIM: 8b) Create Bar Plots using Series and DataFrame index. i) Create bar plots with a DataFrame to group the values in each row together in a group in bars side by side for each value.

Programs:

```
import pandas as pd

import matplotlib.pyplot as plt

plotdata = pd.DataFrame({"pies_2018":[40, 12, 10, 26, 36],"pies_2019":[19, 8, 30, 21, 38],"pies_2020":[10, 10, 42, 17, 37]},index=["Dad", "Mam", "Bro", "Sis", "Me"])

print(plotdata)

plotdata.plot(kind="bar")

plt.show()
```

Output:

```
      pies_2018  pies_2019  pies_2020

Dad       40         19          10

Mam       12          8          10

Bro       10         30          42

Sis       26         21          17

Me        36         38          37
```



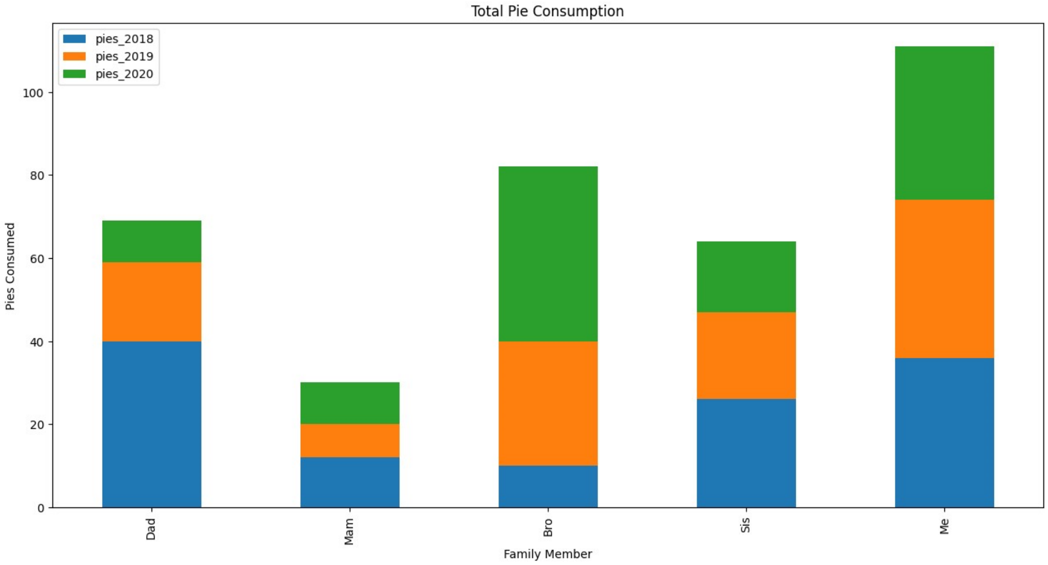
ii) Create stacked bar plots from a DataFrame

```
import pandas as pd
import matplotlib.pyplot as plt
plotdata = pd.DataFrame({"pies_2018":[40, 12, 10, 26, 36],"pies_2019":[19, 8, 30, 21,
38],"pies_2020":[10, 10, 42, 17, 37]},index=["Dad", "Mam", "Bro", "Sis", "Me"])
print(plotdata)
plotdata.plot(kind='bar',stacked=True)
plt.title("Total Pie Consumption")
plt.xlabel("Family Member")
plt.ylabel("Pies Consumed")
plt.show()
```

Output:

```
pies_2018  pies_2019  pies_2020
Dad        40         19         10
```

Mam	12	8	10
Bro	10	30	42
Sis	26	21	17
Me	36	38	37



SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

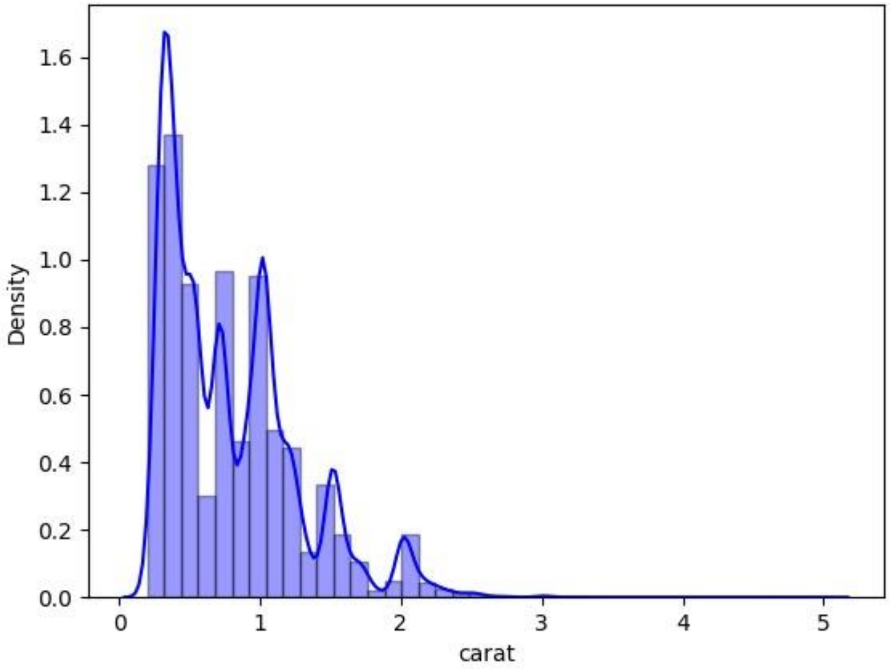
NAME:-	**ROLLNO:-**	**SECTION:-** A

Aim: 8c) Create Histogram to display the value frequency and Density Plot to generate continuous probability distribution function for observed data.

Programs:

```
import seaborn as sns

import matplotlib.pyplot as plt

df=sns.load_dataset("diamonds")

sns.distplot(a=df.carat,bins=40,color="blue",hist_kws={"edgecolor":'black'})

plt.show()
```

Output:

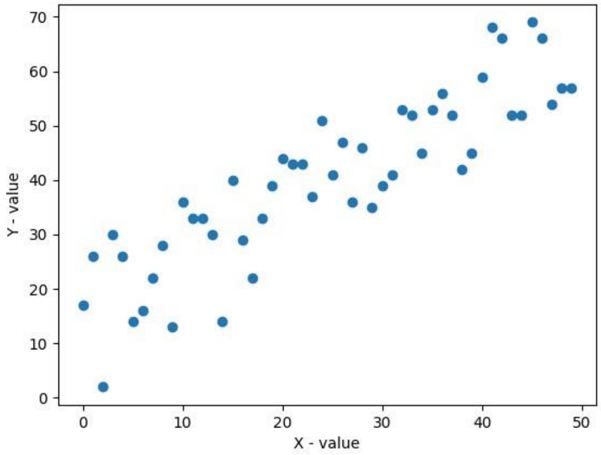


AIM: 8d) Create Scatter Plot and examine the relationship between two one-dimensional data series.

Programs:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

x=pd.Series(range(50))

y=pd.Series(range(50) + np.random.randint(0,30,50))

plt.scatter(x, y)

plt.xlabel('X - value')

plt.ylabel('Y - value')

plt.show()
```

Output:

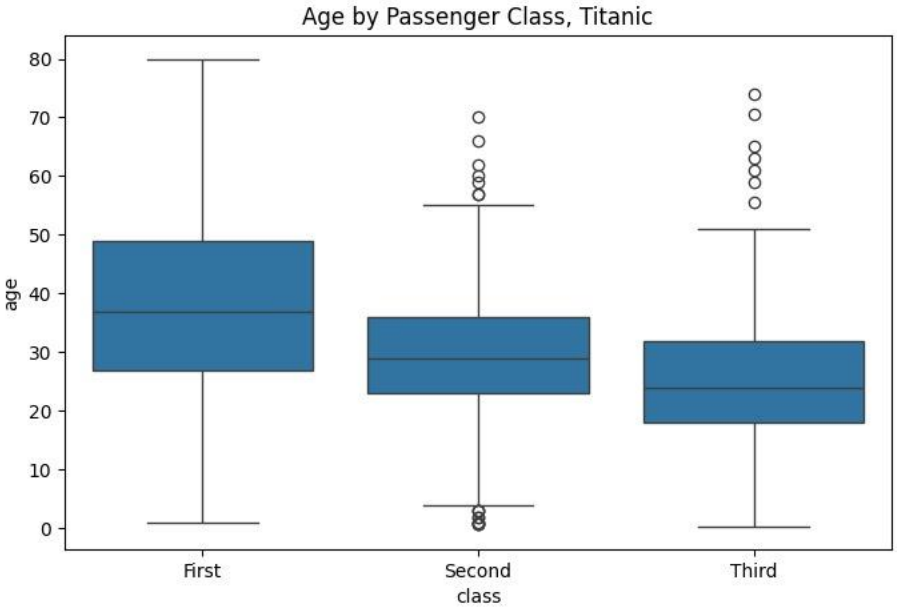


Aim: 8e) Create Box plots to visualize data with many categorical variables

Programs:

```
import seaborn as sns

import matplotlib.pyplot as plt

Text=(0.5, 1.0, 'Age by Passenger Class, Titanic')

titanic = sns.load_dataset('titanic')

plt.figure(figsize=(8,5))

sns.boxplot(x='class',y='age',data=titanic)

plt.title("Age by Passenger Class, Titanic")
```

Output:



SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Autonomous)

Department of Information Technology

III B. Tech – I Semester

(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM:9a) To create time series using datetime object in pandas indexed by timestamps.

Program:

```
import pandas as pd

from datetime import datetime

import numpy as np

range_date = pd.date_range(start ='1-1-2018', end ='1-05-2018', freq ='5H')

df = pd.DataFrame(range_date, columns =['date'])

df['data'] = np.random.randint(0, 100, size =(len(range_date)))

print(df.head(10))
```

Output:

```
                   date  data

0 2018-01-01 00:00:00    70

1 2018-01-01 05:00:00    95

2 2018-01-01 10:00:00    80

3 2018-01-01 15:00:00    82

4 2018-01-01 20:00:00    21

5 2018-01-02 01:00:00    44

6 2018-01-02 06:00:00    15

7 2018-01-02 11:00:00    41

8 2018-01-02 16:00:00    56

9 2018-01-02 21:00:00    18
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Autonomous)
Department of Information Technology
III B. Tech – I Semester
(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:-	**ROLLNO:-**	**SECTION:-** A

AIM:9b) To use pandas.date_range to generate a DatetimeIndex with an indicated length.

Program:

#Specify start and periods, the number of periods (days)

```
import pandas as pd
from datetime import datetime
print(pd.date_range(start='1/1/2018', periods=8))
```

Output:

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04','2018-01-05', '2018-01-06', '2018-
01-07', '2018-01-08'],dtype='datetime64[ns]', freq='D')
```

#Specify end and periods, the number of periods (days)

```
print(pd.date_range(end='1/1/2018', periods=8))
```

Output:

```
DatetimeIndex(['2017-12-25', '2017-12-26', '2017-12-27', '2017-12-28','2017-12-29', '2017-12-30', '2017-
12-31', '2018-01-01'], dtype='datetime64[ns]', freq='D')
```

#Specify start, end, and periods; the frequency is generated automatically (linearly spaced)

```
print(pd.date_range(start='2018-04-24', end='2018-04-27', periods=3))
```

Output:

```
DatetimeIndex(['2018-04-24 00:00:00', '2018-04-25 12:00:00', '2018-04-27 00:00:00'],
dtype='datetime64[ns]', freq=None)
```

AIM:9c) To generate data ranges by setting time zone, localize time zone and convert to particular time zone using tz_convert and combine two different time zones.

Program:

#SETTING TIME ZONE

```
import pandas as pd

import numpy as np

from datetime import datetime

print(pd.date_range('3/9/2012 9:30', periods=10, freq='D', tz='UTC'))
```

Output:

```
DatetimeIndex(['2012-03-09 09:30:00+00:00', '2012-03-10 09:30:00+00:00',
               '2012-03-11 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00',
               '2012-03-15 09:30:00+00:00', '2012-03-16 09:30:00+00:00',
               '2012-03-17 09:30:00+00:00', '2012-03-18 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='D')
```

#LOCALIZE TIME ZONE

```
rng = pd.date_range('3/9/2012 9:30', periods=6, freq='D')

ts = pd.Series(np.random.randn(len(rng)), index=rng)

print(ts)
```

Output:

```
2012-03-09 09:30:00   -0.326128

2012-03-10 09:30:00   -1.469754

2012-03-11 09:30:00    1.598766

2012-03-12 09:30:00   -0.437444

2012-03-13 09:30:00   -0.150390
```

```
2012-03-14 09:30:00   -2.025113
Freq: D, dtype: float64
```

```
ts_utc = ts.tz_localize('UTC')
print(ts_utc)
```

Output:

```
2012-03-09 09:30:00+00:00    1.368678
2012-03-10 09:30:00+00:00   -1.754311
2012-03-11 09:30:00+00:00    0.479707
2012-03-12 09:30:00+00:00   -1.356843
2012-03-13 09:30:00+00:00   -0.274257
2012-03-14 09:30:00+00:00    1.484583
Freq: D, dtype: float64
```

```
#CONVERTING  TO  PARTICULAR  TIME  ZONE
```

```
ts_utc.tz_convert('America/New_York')
```

Output:

```
2012-03-09 04:30:00-05:00 -0.202469
 2012-03-10 04:30:00-05:00 0.050718
 2012-03-11 05:30:00-04:00 0.639869
 2012-03-12 05:30:00-04:00 0.597594
 2012-03-13 05:30:00-04:00 -0.797246
 2012-03-14 05:30:00-04:00 0.472879
Freq: D, dtype: float64
```

SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Autonomous)
Department of Information Technology
III B. Tech – I Semester
(20BT50532) PYTHON FOR DATA SCIENCE LAB

NAME:- **ROLLNO:-** **SECTION:-** A

AIM:9d) To perform period arithmetic such as adding and subtracting integers from periods and construct range of periods using period_range function.

Program:

```
#ADDING AND SUBRACTING INTEGERS FROM PERIODS
p = pd.Period(2007, freq='A-DEC')
print(p)
```

Output:

```
Period('2007', 'A-DEC')
```

```
print(p + 5)
```

Output:

```
Period('2012', 'A-DEC')
```

```
print(p – 2)
```

```
Period('2005', 'A-DEC')
```

```
#construct range of periods using period_range function
rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')
print(rng)
```

Output:

```
PeriodIndex(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '20 00-06'], dtype='period[M]',
freq='M')
```