copy-of-ppml

September 11, 2024

EXP No:1
NUMPY

Aim : To install Numpy package and do the basic functions

1.Declare the Numpy array

[ ]:
```
import numpy as np
arr=np.array([[1,2,4],[3,5,6]])
print(arr)
```

```
[[1 2 4]
 [3 5 6]]
```

2.Create an array with full of zero values

[ ]:
```
b=np.zeros((3,4))
print(b)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

3.Create an array with a Scalar values filled

[ ]:
```
c=np.full((3,3),4)
print(c)
```

```
[[4 4 4]
 [4 4 4]
 [4 4 4]]
```

4.Create an array with random values

[ ]:
```
d=np.random.random((2,3))
print(d)
```

```
[[0.52306126 0.75986483 0.08595463]
 [0.2955301  0.5941752  0.31490108]]
```

[ ]:
```
e=np.arange(5,40,5)
print(e)
```

```
[ 5 10 15 20 25 30 35]
```

5.Reshape and Flattening the array.

[ ]:
```
narr=arr.reshape(3,2)
print(narr)
farr=narr.flatten()
print(farr)
print(narr.ndim)
print(narr.shape)
print(narr.size)
print(narr.dtype)
```

```
[[1 2]
 [4 3]
 [5 6]]
[1 2 4 3 5 6]
2
(3, 2)
6
int64
[[1. 2.]
 [4. 3.]
 [5. 6.]]
```

6.Convert an array from one type to another.

[ ]:
```
print(narr.astype('f'))
```

```
[[1. 2.]
 [4. 3.]
 [5. 6.]]
```

7.Do slicing operations in an array.

[ ]:
```
b=np.array([[1,2,3],[4,5,6],[7,8,9],[11,22,33]])
print(b)
print(b[0:3:2])
print(b[::1])
print(b[::-1])
print(b[0,1:3])
print(b[2:,2:])
print(b[:,2:])
print(b[2:,2])
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [11 22 33]]
[[1 2 3]
```

```
 [7 8 9]]
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [11 22 33]]
[[11 22 33]
 [ 7  8  9]
 [ 4  5  6]
 [ 1  2  3]]
[2 3]
[[ 9]
 [33]]
[[ 3]
 [ 6]
 [ 9]
 [33]]
[ 9 33]
```

8.Do join functions (join,horizontal join,vertical join and depth join)

[ ]:
```
a=np.array([1,2,3])
b=np.array([4,5,6])
print(np.concatenate([a,b]))
print(np.hstack((a,b)))
print(np.vstack((a,b)))
print(np.dstack((a,b)))
```

```
[1 2 3 4 5 6]
[1 2 3 4 5 6]
[[1 2 3]
 [4 5 6]]
[[[1 4]
  [2 5]
  [3 6]]]
```

9.Do index retrivel and basic operation with respect to index

[ ]:
```
a=np.array([1,2,4,6,5,4])
print(np.where(a==4))
print(np.where(a%2==0))
print(np.where(a%2!=0))
```

```
(array([2, 5]),)
(array([1, 2, 3, 5]),)
(array([0, 4]),)
```

[ ]:
```
a=np.array([[1,2,3,4],[2,3,6,7],[9,6,7,3],[3,6,0,1]])
temp=a[[0,1,2,3]]
print("integer array indexing:")
```

```
print(temp)
b=np.array([1,2,3,4])
c=b>2
temp=b[c]
print("boolean array indexing:",temp)
```

```
integer array indexing:
[[1 2 3 4]
 [2 3 6 7]
 [9 6 7 3]
 [3 6 0 1]]
boolean array indexing: [3 4]
```

[ ]:
```
a=np.array([1,2,3,4,5,6])
na=np.array_split(a,3)
print(a)
print(na)
print("Display in another way")
print(na[0])
print(na[1])
print(na[2])
```

```
[1 2 3 4 5 6]
[array([1, 2]), array([3, 4]), array([5, 6])]
Display in another way
[1 2]
[3 4]
[5 6]
```

10.Sorting operation of an array

[ ]:
```
import numpy as np
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))
arr = np.array([True, False, True])
print(np.sort(arr))
arr = np.array([[3, 2, 4], [5, 0, 1]])
print(np.sort(arr))
```

```
[0 1 2 3]
['apple' 'banana' 'cherry']
[False  True  True]
[[2 3 4]
 [0 1 5]]
```

11.Filtering operation based on array value

[ ]:
```
arr=np.array([41,42,43,44])
x=[True,False,True,False]
narr=arr[x]
print(arr,narr,sep="\n")
na=arr>42
print(na)
print(arr[na])
```

```
[41 42 43 44]
[41 43]
[False False  True  True]
[43 44]
```

12.Vector Operation - Addition ,Subtraction, Multiplication and Division

[ ]:
```
c=np.array([11,22,33])
d=np.array([4,5,6])
print(c+d)
print(c-d)
print(c*d)
print(c/d)
```

```
[15 27 39]
[ 7 17 27]
[ 44 110 198]
[2.75 4.4  5.5 ]
```

13.Scalar Operation and Vectorize operation.

[ ]:
```
a=np.array([2,4,6,8])
d=np.array([4,5,6,9])
print(a+2)
print(a-2)
print(a*2)
print(a/2)
print(a.dot(b))
```

```
[ 4  6  8 10]
[0 2 4 6]
[ 4  8 12 16]
[1. 2. 3. 4.]
60
```

[ ]:
```
import numpy as n
def a(x,y):
  if(x>y):
    return x-y
  else:
    return x+y
```

```
e=n.vectorize(a)
c=n.array([1,2,3])
d=n.array([4,5,6])
print(e(c,d))
```

```
[5 7 9]
```

EXP No:2

PANDAS

Aim:To install pandas and do the DataFrame operations

1.Declare Empty Dataframe

[ ]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame()
print(df)
```

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

2.Declare and print the DataFrame Series

[ ]:
```
e=pd.Series(['Arul','Jack','Justin','Kumar'])
i=pd.Series([102,107,109,114])
c={'Emp':e,'ID':i}
r=pd.DataFrame(c)
print(r)
```

```
      Emp   ID
0    Arul  102
1    Jack  107
2  Justin  109
3   Kumar  114
```

3.Add one column and row.

[ ]:
```
r['Age']=pd.Series([20,22,18,24])
print("Add colums")
print(r)
a=pd.
  ↪DataFrame([['Suresh',115,25],['William',118,23]],columns=['Emp','ID','Age'])
r=pd.concat([r,a]).reset_index(drop=True) #r=r.append(a)
print("Add rows")
print(r)
```

```
Add colums
      Emp   ID  Age
```

```
0    Arul  102   20
1    Jack  107   22
2  Justin  109   18
3   Kumar  114   24
Add rows
       Emp   ID  Age
0     Arul  102   20
1     Jack  107   22
2   Justin  109   18
3    Kumar  114   24
4   Suresh  115   25
5  William  118   23
```

[ ]:
```
del r['Age']
print("Delete Column")
print(r)
print("Delete Row")
print(r.drop(5))
```

```
Delete Column
       Emp   ID
0     Arul  102
1     Jack  107
2   Justin  109
3    Kumar  114
4   Suresh  115
5  William  118
Delete Row
      Emp   ID
0    Arul  102
1    Jack  107
2  Justin  109
3   Kumar  114
4  Suresh  115
```

4.Extract any one column and row based on condition

[ ]:
```
print("Extract rows")
print(r.loc[2])
print("Extract column")
print(r['Emp'])
```

```
Extract rows
Emp    Justin
ID        109
Name: 2, dtype: object
Extract column
0       Arul
1       Jack
```

```
2     Justin
3      Kumar
4     Suresh
5    William
Name: Emp, dtype: object
```

5.Do the functions like Sum ,square root ,min,max functiion , sort and merge of values .

[ ]:
```
i=pd.DataFrame([[4,8]]*3,columns=['A','B'])
print(i)
print("Sum of values")
print(i.apply(np.sum,axis=0))
print(i.apply(np.sum,axis=1))
print("Square root of values")
print(i.apply(np.sqrt,axis=0))
print(i.apply(np.sqrt,axis=1))
```

```
   A  B
0  4  8
1  4  8
2  4  8
Sum of values
A    12
B    24
dtype: int64
```

[ ]:
```
i=pd.DataFrame([[1,5,3],[8,5,9],[71,28,11]],columns=['A','B','C'])
print(i)
print("Min of values")
print(i.agg(['min'],axis=0))
print(i.agg(['min'],axis=1))
print("Max of values")
print(i.agg(['max'],axis=0))
print(i.agg(['max'],axis=1))
```

```
    A   B   C
0   1   5   3
1   8   5   9
2  71  28  11
Min of values
     A  B  C
min  1  5  3
   min
0    1
1    5
2   11
Max of values
      A   B   C
max  71  28  11
```

```
   max
0    5
1    9
2   71
```

[ ]:
```
a=pd.DataFrame(np.random.randn(4,2),index=['0','3','1','2'],columns=['A','B'])
print(a)
a1=a.sort_index()
print(a1)
a2=a.sort_index(ascending=False)
print(a2)
a3=a.sort_values(by='B')
print(a3)
```

```
          A         B
0 -1.066436 -0.808643
3  0.645098 -1.034834
1 -0.069905 -0.369483
2 -0.499899 -0.045188
          A         B
0 -1.066436 -0.808643
1 -0.069905 -0.369483
2 -0.499899 -0.045188
3  0.645098 -1.034834
          A         B
3  0.645098 -1.034834
2 -0.499899 -0.045188
1 -0.069905 -0.369483
0 -1.066436 -0.808643
          A         B
3  0.645098 -1.034834
0 -1.066436 -0.808643
1 -0.069905 -0.369483
2 -0.499899 -0.045188
```

[ ]:
```
e=pd.DataFrame({'Id':[1,2,3,4,5],'Name':['A','B','C','D','E'],'Age':
 ↪[20,21,22,20,22]})
f=pd.DataFrame({'Id':[1,2,3,4,5],'Mark1':[60,87,78,98,90],'Mark2':
 ↪[77,88,93,97,88]})
print(pd.merge(e,f,on='Id'))
```

```
   Id Name  Age  Mark1  Mark2
0   1    A   20     60     77
1   2    B   21     87     88
2   3    C   22     78     93
3   4    D   20     98     97
4   5    E   22     90     88
```

6..Create series from array , Dictionary

```
[ ]: arr=np.array([1,2,3,4,5])
     s=pd.Series(arr)
     print(s)
     dic={'a':1,'b':2,'c':3,'d':4,'e':5}
     s=pd.Series(dic)
     print(s)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

7.Create Series using Scalar value,index.

```
[ ]: a=pd.Series(5,index=[0,1,2,3,4])
     print(a)
     b=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
     print(b)
```

```
0    5
1    5
2    5
3    5
4    5
dtype: int64
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

EXP No:3

LOAD AND STORE

Aim: To Create and store Excel / CSV Data Series files and store the Same. Do some basic operations

1.Create a dataframe and store the data into specific excel file

[ ]:
```
import numpy as np
import pandas as pd
e=pd.DataFrame({'Id':[1,2,3,4,5],'Name':['A','B','C','D','E'],'Age':
  ↪[20,21,22,20,22]})
f=pd.DataFrame({'Id':[1,2,3,4,5],'Mark1':[60,87,78,98,90],'Mark2':
  ↪[77,88,93,97,88]})
g=pd.merge(e,f,on='Id')
print(g)
g.to_excel('Dataframe.xlsx',index=False)
```

```
   Id Name  Age  Mark1  Mark2
0   1    A   20     60     77
1   2    B   21     87     88
2   3    C   22     78     93
3   4    D   20     98     97
4   5    E   22     90     88
```

2.Read and display the excel file data.

[ ]:
```
a=pd.read_excel('Dataframe.xlsx')
print(a)
display(a)
```

```
   Id Name  Age  Mark1  Mark2
0   1    A   20     60     77
1   2    B   21     87     88
2   3    C   22     78     93
3   4    D   20     98     97
4   5    E   22     90     88
```

```
   Id Name  Age  Mark1  Mark2
0   1    A   20     60     77
1   2    B   21     87     88
2   3    C   22     78     93
3   4    D   20     98     97
4   5    E   22     90     88
```

3.Display the details of Column headings and shape.

[ ]:
```
print(a.columns)
print(a.shape)
```

```
Index(['Id', 'Name', 'Age', 'Mark1', 'Mark2'], dtype='object')
(5, 5)
```

4.Display the particular column values , row values and do slicing operations.

[ ]:
```
df = pd.read_excel('Dataframe.xlsx')
print(df['Name'])
print(df.iloc[1])
```

```
0    A
1    B
2    C
3    D
4    E
Name: Name, dtype: object
Id        2
Name      B
Age      21
Mark1    87
Mark2    88
Name: 1, dtype: object
```

[ ]:
```
print(df[2:5])
print(df.iloc[1:4,0:2])
```

```
   Id Name  Age  Mark1  Mark2
2   3    C   22     78     93
3   4    D   20     98     97
4   5    E   22     90     88
   Id Name
1   2    B
2   3    C
3   4    D
```

5.To read two excel file data and merge through the append function and store the merged data into the new Excel file.

[ ]:
```
a=pd.DataFrame({'Id':[1,2,3,4,5],'Name':
  ↪['Arun','Ancy','Anika','Reena','Seema'],'Age':[18,17,18,18,18]})
a.to_excel('Dataframe1.xlsx',index=False) #To avoid unnamed index
```

[ ]:
```
b=pd.DataFrame({'Id':[1,2,3,4,5],'Mark1':[60,87,78,98,90],'Mark2':
  ↪[77,88,93,97,88]})
b.to_excel('Dataframe2.xlsx',index=False)
```

[ ]:
```
df1=pd.read_excel('Dataframe1.xlsx')
df2=pd.read_excel('Dataframe2.xlsx')
c=pd.concat([df1,df2],axis=1)     #c=df1.append(df2)
df3=c.to_excel('Dataframe3.xlsx',index=False)
print(c)
```

```
   Id   Name  Age  Id  Mark1  Mark2
0   1   Arun   18   1     60     77
1   2   Ancy   17   2     87     88
2   3  Anika   18   3     78     93
3   4  Reena   18   4     98     97
4   5  Seema   18   5     90     88
```

6.Using sort function to sort and store the resultant data into a new Excel file

```
[ ]: x=pd.DataFrame({'Id':[1,2,3,4,5],'Mark1':[60,87,78,98,90],'Mark2':
       ↪[77,88,93,97,88]})
     x.to_excel('Dataframe2.xlsx',index=False)
     y=pd.read_excel('Dataframe2.xlsx')
     s=y.sort_values(by='Mark1')                    #s=y.sort(by='Mark1')
     s.to_excel('SortDF.xlsx')
     a=pd.read_excel('SortDF.xlsx')
     print(a)
```

```
   Unnamed: 0  Id  Mark1  Mark2
0           0   1     60     77
1           2   3     78     93
2           1   2     87     88
3           4   5     90     88
4           3   4     98     97
```

EXP No:4

TEXTFILE AND JSON

Aim : To open , read and write the text files and basic JSON operation.

1.Open a text file in write mode

2.Write the content and close the file.

```
[ ]: a=open("ppml.txt",'w')
     a.write("Python is a programming language")
     a.close()
```

3.Open the same text file in read mode and read the contents

```
[ ]: e=open("ppml.txt",'r')
     print(e.read())
     e.close()
```

4.Read the contents through the read function readline() function.

```
[ ]: e = open("ppml.txt", 'r')
     print(e.readline())
     e.close()
```

```
Python is a programming language
```

5.Add some content to the already created file.

```
[ ]: b=open("ppml.txt",'a')
     b.write("\nIt is easy to learn")
     b.close()
```

6.Display the file content through the read function and close the file.

```
[ ]: e=open("ppml.txt",'r')
print(e.read())
e.close()
```

```
Python is a programming language
It is easy to learn
```

7.Use basic JSON loads and dumps functions.

```
[ ]: import json
x='{"Name":"Ancy","Age":20,"City":"Kanyakumari"}'
y=json.loads(x)
z=json.dumps(x)
print(y)
print(z)
```

```
{'Name': 'Ancy', 'Age': 20, 'City': 'Kanyakumari'}
"{\"Name\":\"Ancy\",\"Age\":20,\"City\":\"Kanyakumari\"}"
```

EXP No:5

DATA CLEANING AND PREPARATION

```
[ ]: import pandas as pd
import numpy as np
a=pd.DataFrame(np.random.
  ↪randn(5,3),index=['a','c','e','f','h'],columns=['One','Two','Three'])
print(a)
a=a.reindex(['a','b','c','d','e','f','g','h'])
print(a)
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
c  0.292735  0.865780 -0.471783
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
h  0.110259 -1.922232  0.253248
        One       Two     Three
a  0.599844  1.445202 -1.223061
b       NaN       NaN       NaN
c  0.292735  0.865780 -0.471783
d       NaN       NaN       NaN
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
g       NaN       NaN       NaN
h  0.110259 -1.922232  0.253248
```

[ ]:
```
a1=a
print(a.dropna())
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
c  0.292735  0.865780 -0.471783
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
h  0.110259 -1.922232  0.253248
```

[ ]:
```
a2=a1
print(a1.fillna(0))
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
b  0.000000  0.000000  0.000000
c  0.292735  0.865780 -0.471783
d  0.000000  0.000000  0.000000
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
g  0.000000  0.000000  0.000000
h  0.110259 -1.922232  0.253248
```

[ ]:
```
a3=a2
print(a2.fillna(method='pad'))
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
b  0.599844  1.445202 -1.223061
c  0.292735  0.865780 -0.471783
d  0.292735  0.865780 -0.471783
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
g -1.123648  1.540607 -0.347031
h  0.110259 -1.922232  0.253248
<ipython-input-5-b927352de9fc>:2: FutureWarning: DataFrame.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  print(a2.fillna(method='pad'))
```

[ ]:
```
a4=a3
print(a3.fillna(method='bfill'))
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
b  0.292735  0.865780 -0.471783
c  0.292735  0.865780 -0.471783
```

```
d -0.014980  1.944982  1.774031
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
g  0.110259 -1.922232  0.253248
h  0.110259 -1.922232  0.253248
```

```
<ipython-input-6-1f554bed7946>:2: FutureWarning: DataFrame.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  print(a3.fillna(method='bfill'))
```

[ ]:
```
a5=a4
print(a4.bfill())
```

```
        One       Two     Three
a  0.599844  1.445202 -1.223061
b  0.292735  0.865780 -0.471783
c  0.292735  0.865780 -0.471783
d -0.014980  1.944982  1.774031
e -0.014980  1.944982  1.774031
f -1.123648  1.540607 -0.347031
g  0.110259 -1.922232  0.253248
h  0.110259 -1.922232  0.253248
```

[ ]:
```
print(a['One'].isnull())
print(a['One'].notnull())
```

```
a    False
b     True
c    False
d     True
e    False
f    False
g     True
h    False
Name: One, dtype: bool
a     True
b    False
c     True
d    False
e     True
f     True
g    False
h     True
Name: One, dtype: bool
```

[ ]:
```
b=pd.
  ↪DataFrame([[11,'a'],[12,'b'],[13,'c'],[14,'d'],[15,'e'],[103,'f'],[101,'g'],[18,'h']],colum
```

```
print(b)
```

```
   Age Name
0   11    a
1   12    b
2   13    c
3   14    d
4   15    e
5  103    f
6  101    g
7   18    h
```

[ ]:
```
print(b.replace({103:16,101:17}))
```

```
   Age Name
0   11    a
1   12    b
2   13    c
3   14    d
4   15    e
5   16    f
6   17    g
7   18    h
```

EXP NO:6

DATA WRANGLING

[ ]:
```
import pandas as pd
import numpy as np
d1={"name":["salini","Mary","Johncy"],"age":[40,60,38]}
d2={"Qualified":[True,False,True]}
df1=pd.DataFrame(d1)
df2=pd.DataFrame(d2)
nd=df1.join(df2)
print(nd)
```

```
     name  age  Qualified
0  salini   40       True
1    Mary   60      False
2  Johncy   38       True
```

[ ]:
```
df=pd.DataFrame({"team":['A','B','C','D'],"points":[88,89,99,98],"assist":
↪[17,14,16,12],"rebounds":[22,21,25,38]})
print(df)
df1=pd.melt(df,id_vars=['team'],value_vars=['points','assist','rebounds'])
print(df1)
```

```
  team  points  assist  rebounds
```

```
0    A      88      17        22
1    B      89      14        21
2    C      99      16        25
3    D      98      12        38
   team  variable  value
0     A    points     88
1     B    points     89
2     C    points     99
3     D    points     98
4     A    assist     17
5     B    assist     14
6     C    assist     16
7     D    assist     12
8     A  rebounds     22
9     B  rebounds     21
10    C  rebounds     25
11    D  rebounds     38
```

Data Grouping Function

[ ]:
```
b=pd.
  ↪DataFrame([['Hen',80],['Hen',100],['Parrot',40],['Parrot',30],['Finges',10],['Finges',15]],
print(b)
```

```
     Name  Speed
0     Hen     80
1     Hen    100
2  Parrot     40
3  Parrot     30
4  Finges     10
5  Finges     15
```

[ ]:
```
b.groupby(['Name']).mean()
```

[ ]:
```
         Speed
Name
Finges   12.5
Hen      90.0
Parrot   35.0
```

[ ]:
```
b.groupby(['Name']).sum()
```

[ ]:
```
         Speed
Name
Finges     25
Hen       180
Parrot     70
```

[ ]:
```
b.groupby(['Name']).count()
```

[ ]:
```
         Speed
Name
Finges       2
Hen          2
Parrot       2
```

[ ]:
```
b.groupby(['Name']).first()
```

[ ]:
```
         Speed
Name
Finges      10
Hen         80
Parrot      40
```

[ ]:
```
.groupby(['Name']).last()
```

[ ]:
```
         Speed
Name
Finges      15
Hen        100
Parrot      30
```

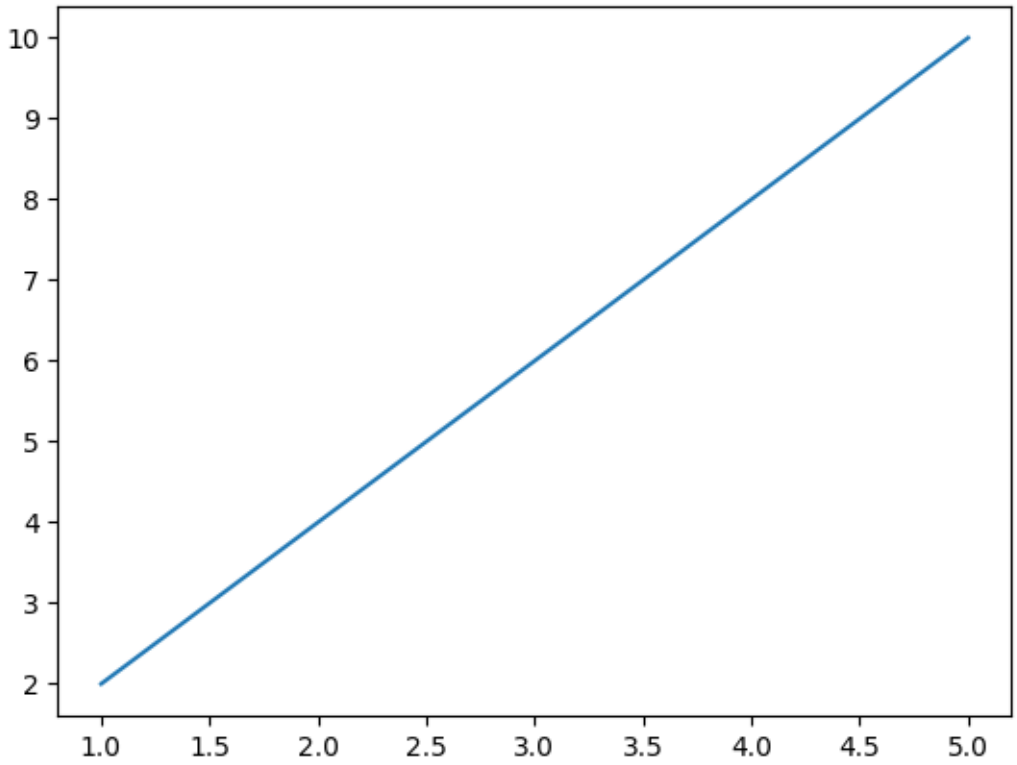
[ ]:
```
b.groupby(['Name']).size()
```

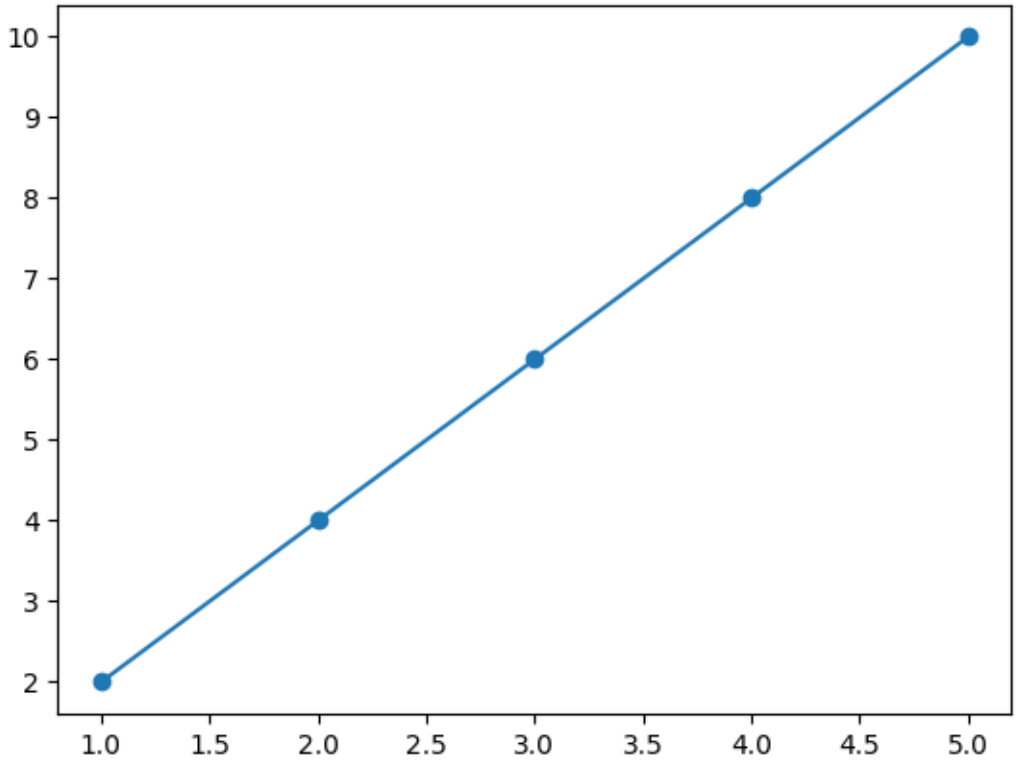
[ ]:
```
Name
Finges    2
Hen       2
Parrot    2
dtype: int64
```

EXP NO: 7

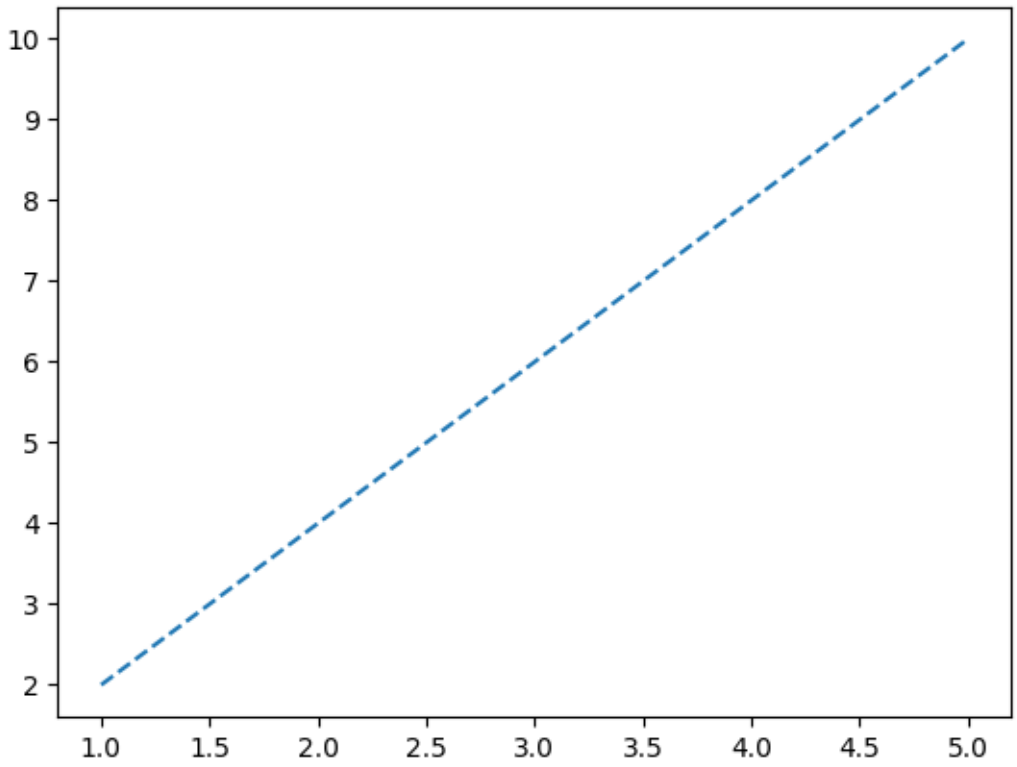
DATA VISUALIZATION

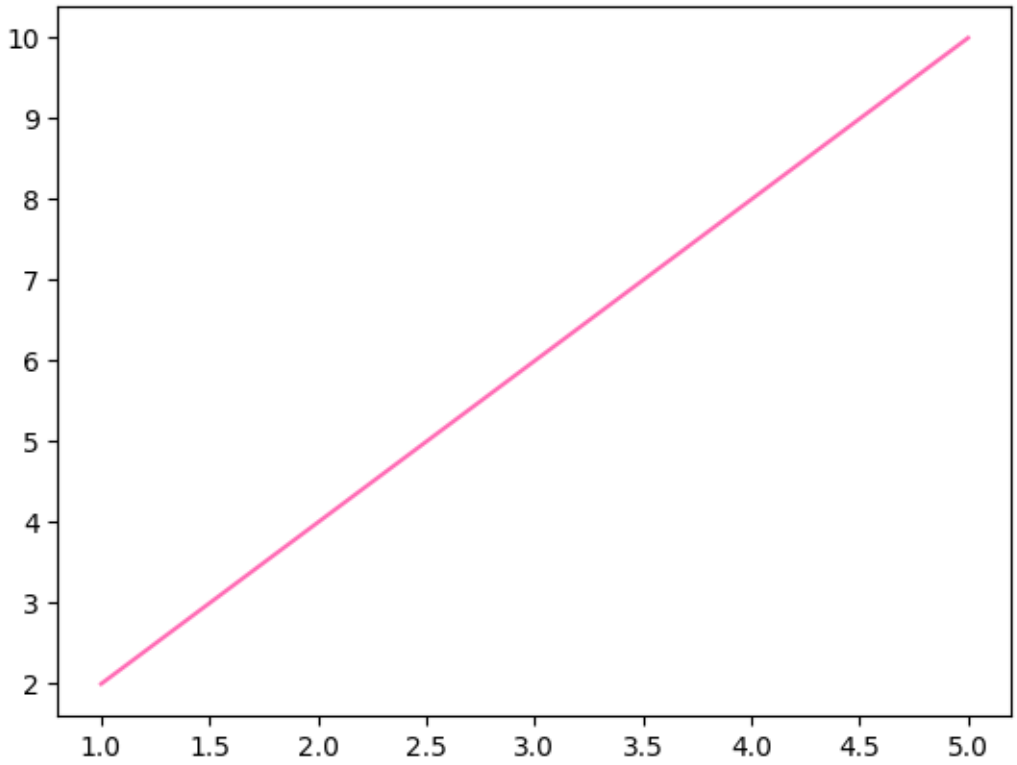
[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([1,2,3,4,5])
y=np.array([2,4,6,8,10])
plt.plot(x,y)
plt.show()
plt.plot(x,y,marker='o')
plt.show()
```

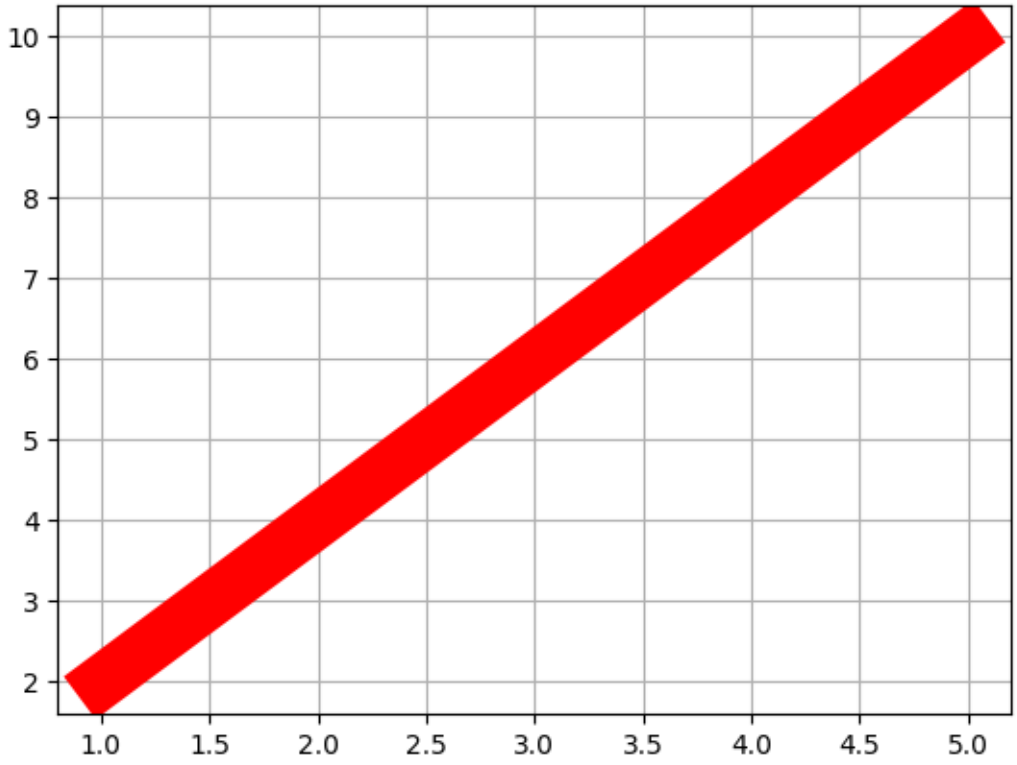

10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

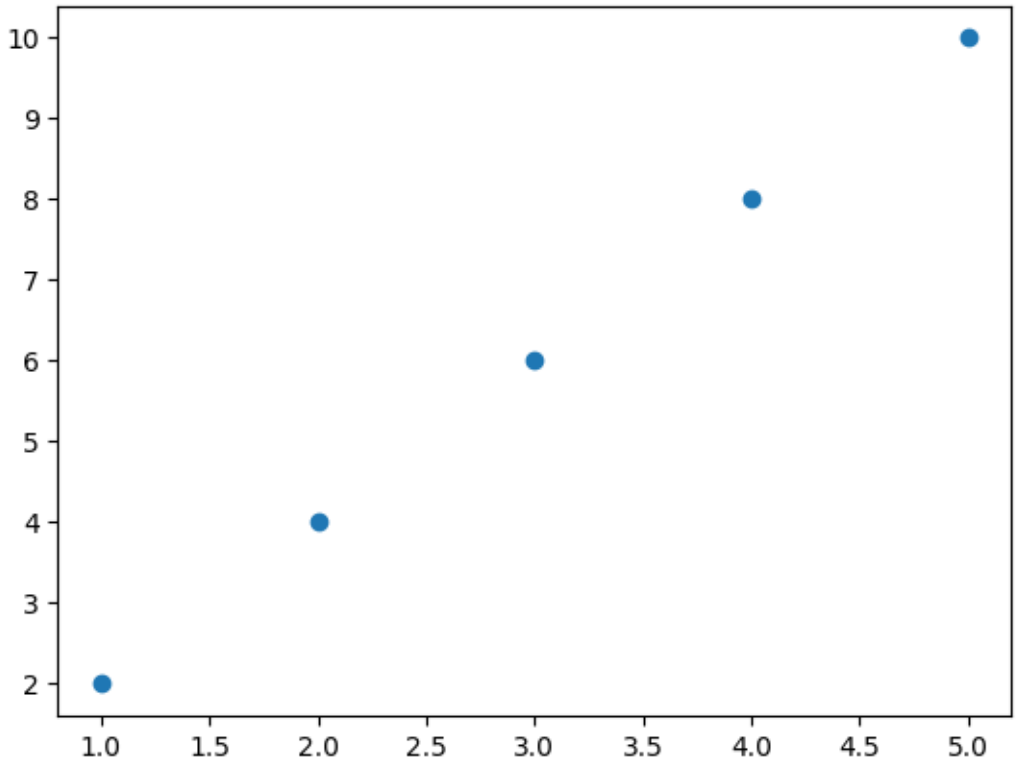

10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

[ ]:
```
plt.plot(x,y,linestyle='dashed')
plt.show()
plt.plot(x,y,'hotpink')
plt.show()
plt.plot(x,y,linewidth='20',color='r')
plt.grid()
plt.show()
plt.scatter(x,y)
plt.show()
```

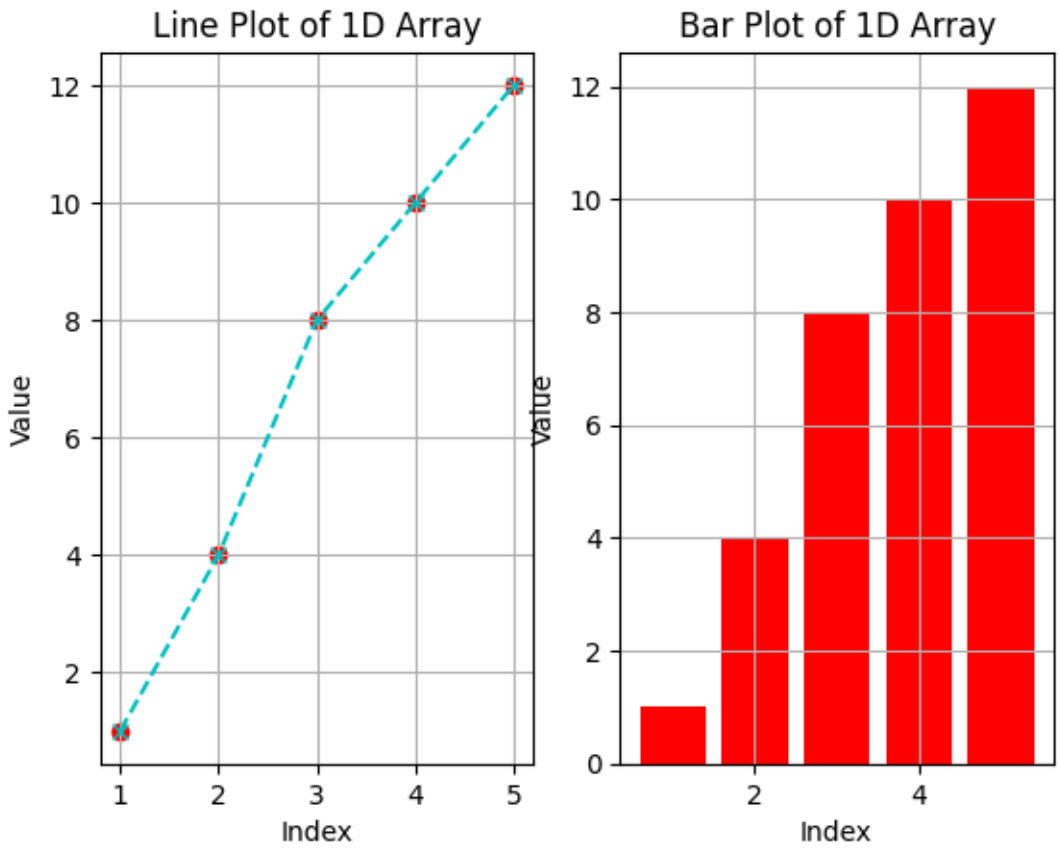



10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

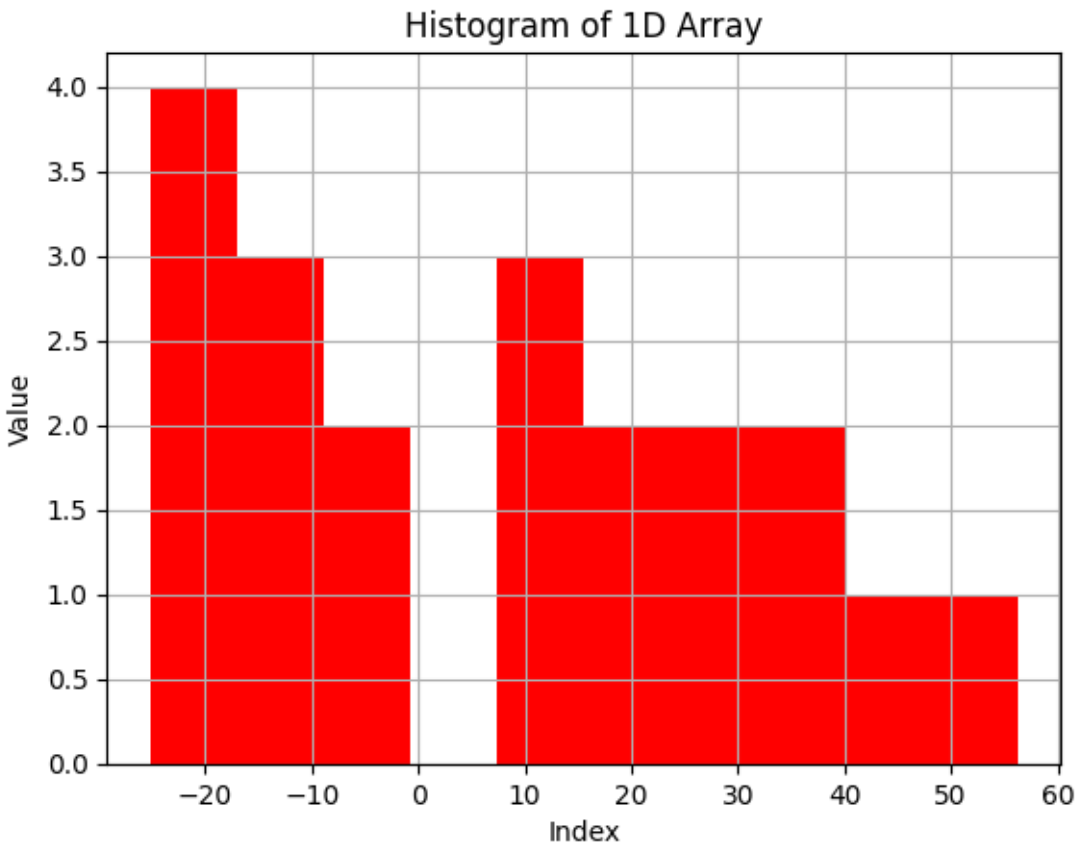


[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
a=np.array([1,2,3,4,5])
b=np.array([1,4,8,10,12])
plt.subplot(1,2,1)
plt.plot(a,b,marker='x',linestyle='dashed',color='c')
plt.grid(True)
plt.scatter(a,b,color='r')
plt.xlabel("Index")
plt.ylabel("Value")
plt.title("Line Plot of 1D Array")
plt.subplot(1,2,2)
plt.bar(a,b,color='r')
plt.grid(True)
plt.xlabel("Index")
plt.ylabel("Value")
plt.title("Bar Plot of 1D Array")
plt.show()
```

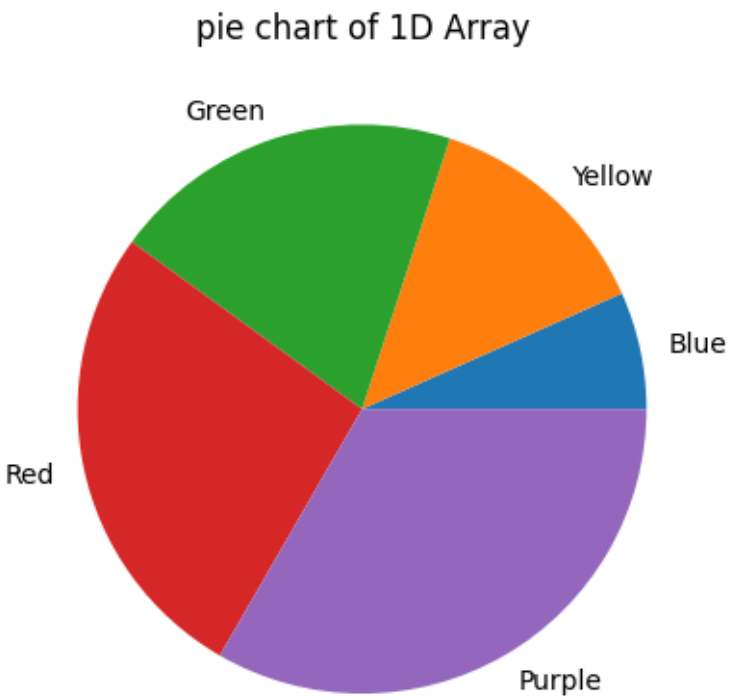


[ ]:
```
e=np.random.normal(10,30,20)
print(e)
plt.hist(e,color='r')
plt.grid(True)
plt.xlabel("Index")
plt.ylabel("Value")
plt.title("Histogram of 1D Array")
plt.show()
```

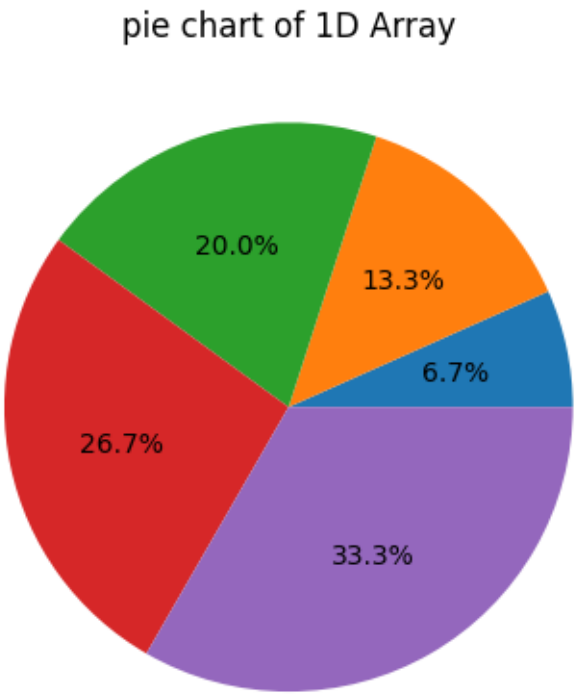
```
[ 20.09932999  25.43011562  56.24606001 -14.12311315  18.15699551
  -9.04304705  24.64913647  -7.37388312   7.74658115  37.94521752
  36.34503627 -19.07623019 -18.94282536 -11.76623775  -3.08199169
 -25.17000888   9.0193486   44.90876525  13.92293299 -24.25308911]
```



[ ]:
```
c=np.array(['Blue','Yellow','Green','Red','Purple'])
plt.pie(a, labels=c)
plt.grid(True)
plt.title("pie chart of 1D Array")
plt.show()
```



[ ]:
```
plt.pie(a,autopct='%1.1f%%')
plt.title("pie chart of 1D Array")
plt.show()
```



EXP NO:8

TIMESERIES

[ ]:
```
import datetime as dt
r=dt.datetime.now()
s=dt.datetime.today()
print(r)
print(s)
```

```
2024-08-22 17:16:43.014071
2024-08-22 17:16:43.014132
```

[ ]:
```
t=r+dt.timedelta(days=1)
o=r-dt.timedelta(days=2)
print(t)
print(o)
```

```
2024-08-22 05:09:47.689781
2024-08-19 05:09:47.689781
```

[ ]:
```
a=dt.datetime(2020,6,8,23,10,25,7264)
print(a)
```

```
2020-06-08 23:10:25.007264
```

[ ]:
```
print(a.replace(day=26))
print(a.replace(month=12))
```

```
2020-06-26 23:10:25.007264
2020-12-08 23:10:25.007264
```

[ ]:
```
print(dt.date(2004,10,1).ctime())
```

```
Fri Oct  1 00:00:00 2004
```

[ ]:
```
print(r.strftime("%Y"))
print(r.strftime("%M"))
print(r.strftime("%b"))
print(r.strftime("%B"))
print(r.strftime("%j"))
print(r.strftime("%D"))
print(r.strftime("%d"))
print(r.strftime("%a"))
print(r.strftime("%A"))
print(r.strftime("%H"))
print(r.strftime("%S"))
print(r.strftime("%F"))
print(r.strftime("%p"))
print(r.strftime("%x"))
print(r.strftime("%X"))
#print(r.strftime("%c"))
#print(r.strftime("%I"))
#print(r.strftime("%m"))
```

```
2024
16
Aug
August
235
08/22/24
22
Thu
Thursday
17
43
2024-08-22
PM
08/22/24
17:16:43
```