

EXERCISE-2

2)a. single dimensional arrays.

```
import numpy as np
a=np.array([1,2,3])
print('My Array is:',a)
```

OUTPUT:

My Array is: [1 2 3]

2)b)addition,subtraction and multiplication of matrices for integers ,floating point and complex numbers

```
arr1=np.arange(1,5).reshape(2,2)
print('arr1:',arr1)
print('Addition:',arr1+4)
print('Subtraction:',arr1-4)
print('Multiplication:',arr1*2)
print('Division:',arr1//2)
a=np.array([[2+3j,1+3j]],[[1+3j,2+3j]])
b=np.array([[1+3j,2+3j]],[[2+3j,1+3j]])
print('Addition of complex matrix:')
print(np.add(a,b))
print('subtraction of matrices for integers:')
c=np.array([[1,2],[3,4]])
d=np.array([[4,5],[6,7]])
print(np.sutraction(c,d))
print(print(np.sutraction(c,d)))
print('multiplication of matrices for integers:')

r=np.dot(c,d)
print(r)
```

OUTPUT:

```
arr1: [[1 2] [3 4]]
```

```
Addition: [[5 6] [7 8]]
```

```
Subtraction: [[-3 -2] [-1 0]]
```

```
Multiplication: [[2 4] [6 8]]
```

```
Division: [[0 1] [1 2]]
```

Addition of complex matrix:

```
[[3+6j 3+6j]
```

```
[3+6j 3+6j]]
```

'subtraction of matrices for integers:

```
[[ -3 -3]
```

```
[-3 -3]]
```

'multiplication of matrices for integers:

```
[[16 19]
```

```
[36 43]]
```

2)c.performing slicing of matrix

#Slicing

```
arr1=np.arange(10,19)
```

```
print('arr1:',arr1)
```

```
print('arr1[1:5]:',arr1[1:5])
```

```
print('arr1[3:]',arr1[3:])
```

```
print('arr1[:3]',arr1[:3])
```

```
print('arr1[1:6:2]:',arr1[1:6:2])
```

```
print('arr1[::2]:',arr1[::2])
```

```
print('arr1[::-1]:',arr1[::-1])
```

OUTPUT: arr1: [10 11 12 13 14 15 16 17 18]

```
arr1[1:5]: [11 12 13 14]
arr1[3:] [13 14 15 16 17 18]
arr1[:3] [10 11 12]
arr1[1:6:2]: [11 13 15]
arr1[::2]: [10 12 14 16 18]
arr1[::-1]: [18 17 16 15 14 13 12 11 10]
```

2)d.transpose of a matrix

```
matrix=[(1,2,3),(4,5,6),(7,8,9),(10,11,12)]
```

```
for row in matrix:
```

```
    print(row)
```

```
print("\n")
```

```
t_matrix = zip(*matrix)
```

```
for row in t_matrix:
```

```
    print(row)
```

OUTPUT:

```
[1, 2]
```

```
[3, 4]
```

```
[5, 6]
```

```
[1, 3, 5]
```

```
[2, 4, 6]
```

EXERCISE-3

#max and min,sum,mean and standard deviation

```
a=np.array([3.3,4.5,1.2,5.7,0.3])
```

```
print('Sum:',a.sum())
```

```
#np.sum(a)
```

```
print('Min:',a.min())
```

```
print('Max:',a.max())
```

```
print('Mean:',a.mean())
```

```
print('Std:',a.std())
```

OUTPUT:

Sum: 15.0

Min: 0.3

Max: 5.7

Mean: 3.0

Std: 2.0079840636817816

EXERCISE-4

#Reading and Writing Array Data on Files #Saving Data in Binary Files

```
data=np.arange(12).reshape(4,3)
```

```
print('Data is:\n',data)
```

```
print('Saving data to saved_data.npy file. ')
```

```
np.save('saved_data',data)
```

```
mydata=np.load('saved_data.npy')
```

```
print('Loaded data is:\n',mydata)
```

OUTPUT:

Data is:

```
[[ 0 1 2]
```

```
 [ 3 4 5]
```

```
 [ 6 7 8]
```

```
 [ 9 10 11]]
```

Saving data to saved_data.npy file.

Loaded data is:

```
[[ 0 1 2]
```

```
 [ 3 4 5]
```

```
 [ 6 7 8]
```

```
 [ 9 10 11]]
```

```
data = np.genfromtxt('sample.csv', delimiter=',', names=True)
```

```
print('Data form Sample.csv file is:\n',data)
```

```
data = np.genfromtxt('sample.csv', delimiter=',', names=True, dtype=('i2,S6,S20,i8'))
```

```
print('Data form Sample.csv file is:\n',data)
```

OUTPUT:

Data form Sample.csv file is:

```
[(3001., nan, nan, 1.00000000e+10) (3002., nan, nan, 8.88888889e+09)
(3003., nan, nan, 7.77777778e+09) (1241., nan, nan, 9.96605283e+09)]

Data form Sample.csv file is: [(3001, b'ABC', b'abc@gmail.com', 9999999999)
(3002, b'XYZ', b'xyz@gmail.com', 8888888888)
(3003, b'BBC', b'bbc@gmail.com', 7777777777)
(1241, b'PNR', b'pnr2830@gmail.com', 9966052830)]
```

EXERCISE-5

#integration functions using scipy

Single integration functions

```
from scipy import integrate
```

```
>>> import numpy as np
```

```
>>> x2 = lambda x: x**2
```

```
>>> integrate.quad(x2, 0, 4)
```

```
(21.333333333333332, 2.3684757858670003e-13)
```

```
>>> print(4**3 / 3.)
```

```
21.3333333333 invexp = lambda x: np.exp(-x)
```

```
>>> integrate.quad(invexp, 0, np.inf)
```

```
(1.0, 5.842605999138044e-11)
```

```
f = lambda x, a: a*x
```

```
>>> y, err = integrate.quad(f, 0, 1, args=(1,))
```

```
>>> y
```

```
0.5
```

```
>>> y, err = integrate.quad(f, 0, 1, args=(3,))
```

```
>>> y
```

```
1.5
```

```
from scipy.integrate import quad_vec
```

```
>>> import numpy as np
```

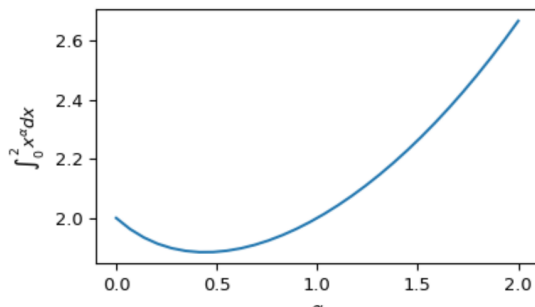
```
>>> import matplotlib.pyplot as plt
```

```
>>> alpha = np.linspace(0.0, 2.0, num=30)
```

```
>>> f = lambda x: x**alpha
```

```
>>> x0, x1 = 0, 2
```

```
>>> y, err = quad_vec(f, x0, x1)
>>> plt.plot(alpha, y)
>>> plt.xlabel(r"$\alpha$")
>>> plt.ylabel(r"$\int_0^2 x^\alpha dx$")
>>> plt.show()
```



#double integration

```
import numpy as np
>>> from scipy import integrate
>>> f = lambda y, x: x*y**2
>>> integrate.dblquad(f, 0, 2, 0, 1)
```

Output:

```
(0.6666666666666667, 7.401486830834377e-15)
```

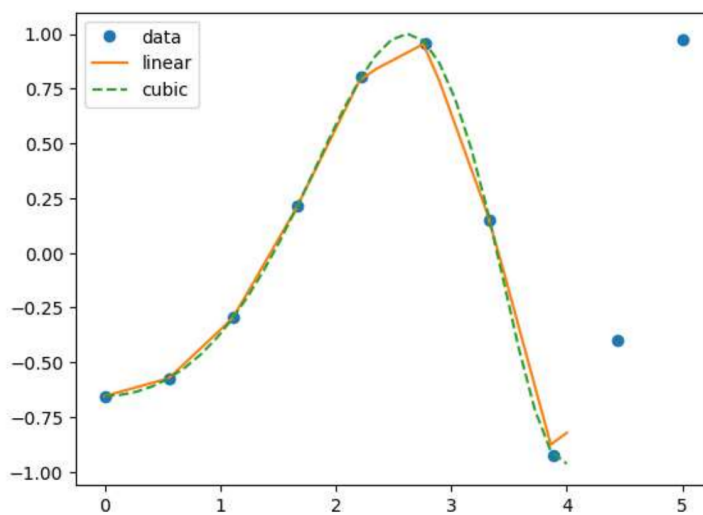
#triple integration

```
import numpy as np
>>> from scipy import integrate
>>> f = lambda z, y, x: x*y*z
>>> integrate.tplquad(f, 1, 2, 2, 3, 0, 1)
(1.8749999999999998, 3.3246447942574074e-14)
f = lambda z, y, x: x*y*z
integrate.tplquad(f, 0, 1, 0, lambda x: 1-2*x, 0, lambda x, y: 1-x-2*y)
(0.05416666666666668, 2.1774196738157757e-14)
f = lambda z, y, x, a: a*x*y*z
>>>
integrate.tplquad(f, 0, 1, 0, 1, 0, 1, args=(1,))
(0.125, 5.527033708952211e-15)
```

EXERCISE-6

#interpolation function using scipy

```
In [3]: import numpy as np
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
x = np.linspace(0, 5, 10)
y = np.cos(x**2/3+4)
fun1 = interp1d(x, y, kind = 'linear')
fun2 = interp1d(x, y, kind = 'cubic')
xnew = np.linspace(0, 4, 30)
plt.plot(x, y, 'o', xnew, fun1(xnew), '-', xnew, fun2(xnew), '--')
plt.legend(['data', 'linear', 'cubic', 'nearest'], loc = 'best')
plt.show()
```



```
In [9]: import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x = np.arange(-5.01, 5.01, 0.25)
y = np.arange(-5.01, 5.01, 0.25)
xx, yy = np.meshgrid(x, y)
z = np.sin(xx**2+yy**2)
f = interpolate.interp2d(x, y, z, kind='cubic')
xnew = np.arange(-5.01, 5.01, 1e-2)
ynew = np.arange(-5.01, 5.01, 1e-2)
znew = f(xnew, ynew)
plt.plot(x, z[0, :], 'ro-', xnew, znew[0, :], 'b-')
plt.show()
```

```

C:\Users\JKLabSys\AppData\Local\Temp\ipykernel_13536\2495500341.py:7: Deprecation
Warning: `interp2d` is deprecated!
`interp2d` is deprecated in SciPy 1.10 and will be removed in SciPy 1.12.0.

For legacy code, nearly bug-for-bug compatible replacements are
`RectBivariateSpline` on regular grids, and `bisplep`/`bisplev` for
scattered 2D data.

In new code, for regular grids use `RegularGridInterpolator` instead.
For scattered data, prefer `LinearNDInterpolator` or
`CloughTocher2DInterpolator`.

For more details see
`https://gist.github.com/ev-br/8544371b40f414b7eaf3fe6217209bfff`

f = interpolate.interp2d(x, y, z, kind='cubic')
C:\Users\JKLabSys\AppData\Local\Temp\ipykernel_13536\2495500341.py:10: Deprecatio
nWarning: `interp2d` is deprecated!
`interp2d` is deprecated in SciPy 1.10 and will be removed in SciPy 1.12.
0.

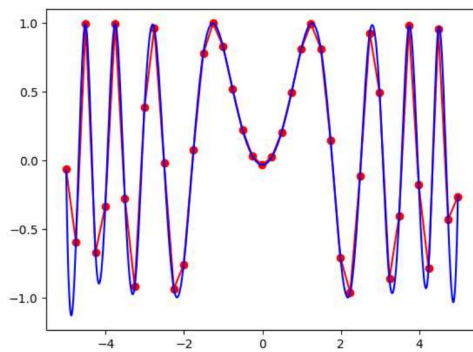
For legacy code, nearly bug-for-bug compatible replacements are
`RectBivariateSpline` on regular grids, and `bisplep`/`bisplev` for
scattered 2D data.

In new code, for regular grids use `RegularGridInterpolator` instead.
For scattered data, prefer `LinearNDInterpolator` or
`CloughTocher2DInterpolator`.

For more details see
`https://gist.github.com/ev-br/8544371b40f414b7eaf3fe6217209bfff`

znew = f(xnew, ynew)

```



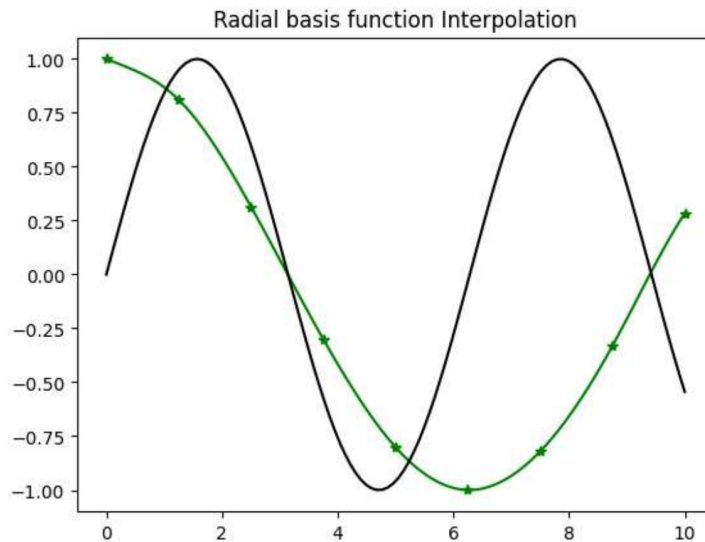
```

In [6]: import numpy as np
        from scipy.interpolate import Rbf
        import matplotlib.pyplot as plt

        # setup the data values
        x = np.linspace(0, 10, 9)
        y = np.cos(x/2)
        xi = np.linspace(0, 10, 110)

        # Interpolation using RBF
        rbf = Rbf(x, y)
        fi = rbf(xi)
        plt.plot(x, y, '*', color="green")
        plt.plot(xi, fi, 'green')
        plt.plot(xi, np.sin(xi), 'black')
        plt.title('Radial basis function Interpolation')
        plt.show()

```



EXERCISE-7

#linear systems of equations using scipy

```
In [7]: import numpy as np
        from scipy import linalg

        # The Linear algebra system which is given as

        # 3x + 2y = 2

        # x - y = 4

        # 5y + z = -1

        #We need to find values of x,y and z for which all these equations are zero

        # Creating the input array
        a = np.array([[3, 2, 0], [1, -1, 0], [0, 5, 1]])

        # Providing the solution Array
        b = np.array([2, 4, -1])

        # Solve the Linear algebra
        x = linalg.solve(a, b)

        # Printing the result
        print(x)

        # Checking the result
        np.dot(a, x) == b
```

```
[[ 2.]
 [-2.]
 [ 9.]]

Out[7]: array([[ True],
               [ True],
               [ True]])
```

```
In [8]: import numpy as np
        from scipy import linalg

        #x + 2y - 3z = -3

        #2x - 5y + 4z = 13

        #5x + 4y - z = 5

        # Creating the input array
        a = np.array([[1, 2, -3], [2, -5, 4], [5, 4, -1]])

        # Providing the solution Array
        b = np.array([-3, 13, 5])

        # Solve the linear algebra
        x = linalg.solve(a, b)

        # Printing the result
        print(x)

        # Checking the result
        np.dot(a, x) == b

[[ 2.]
 [-1.]
 [ 1.]]

Out[8]: array([[ True],
               [ True],
               [ True]])
```

EXERCISE-8

#pandas#pandas series#pandas data frames

```
import pandas as pd
s=pd.Series([15,-2,3,1],index=['x','y','z','w'])
print(s[0:2]) print(s[['y','w']])
```

OUTPUT:

```
x  15
y  -2
dtype: int64
y  -2
w   1
dtype: int64
```

```
import pandas as pd
ser=pd.Series([5,-2,3,4])
print(ser>2)
print(ser[ser>2])
```

OUTPUT:

```
0    True
1    False
2     True
3     True
dtype: bool
0 5 2 3 3 4
dtype: int64
```

#operations between series

```
mydict={'red':2000,'yellow':500,'blue':300}
mydict1={'red':500,'yellow':700,'blue':500,'green':400}
ser=pd.Series(mydict)
ser1=pd.Series(mydict1)
print(ser)
print(ser1)
print("ser1+ser",ser1+ser,sep='\n')
```

OUTPUT:

red 2000

yellow 500

blue 300

dtype: int64

red 500

yellow 700

blue 500

#pandas dataframe

```
mydict={'color':['red','yellow','green','blue','orange'],
```

```
'object':['ball','pen','book','scale','mug'],
```

```
'price':[1.5,2.4,1.3,3.6,4.8]}
```

```
frame=pd.DataFrame(mydict,index=['one','two','three','four','five'])
```

```
print(frame)
```

```
print()
```

```
print(frame['color'],end="\n")
```

```
print(frame.price)
```

```
print(frame.columns,frame.index,sep="\n")
```

```
print(frame.values)
```

OUTPUT:

	Color	object	price
One	red	ball	1.5
two	yellow	pen	2.4
three	green	book	1.3
four	blue	scale	3.6
five	orange	mug	4.8
one	red		
two	yellow		
three	green		
four	blue		
five	orange		

Name: color, dtype: object

One	1.5
two	2.4
three	1.3
four	3.6
five	4.8

Name: price, dtype: float64

Index(['color', 'object', 'price'], dtype='object') Index(['one', 'two', 'three', 'four', 'five'], dtype='object')

[['red' 'ball' 1.5]

['yellow' 'pen' 2.4]

['green' 'book' 1.3]

['blue' 'scale' 3.6]

['orange' 'mug' 4.8]]

EXERCISE-9

#reindexing

```
ser3=pd.Series([2,5,7,4],index=['one','two','three','four'])
```

```
print(ser3)
```

```
ser3.reindex(['three','four','five','one'])
```

one 2

two 5

three 7

four 4

dtype: int64

Out[12]:

three 7.0

four 4.0

five NaN

one 2.0

dtype: float6

#dropping

import pandas as pd

```
import numpy as np

ser5=pd.Series(np.arange(4),index=['red','green','blue','white'])

print(ser5)

print('dropping index green')

print(ser5.drop('green'))

print(ser5.drop(['red','blue']))

red    0
green   1
blue    2
white   3
dtype: int32
```

#dropping in dataframes

```
frame4=pd.DataFrame(np.arange(16).reshape(4,4),
index=['r','b','g','w'], columns=['ball','pen','pencil','book'])

print(frame4)

print('dropping indexes blue , green')

print(frame4.drop(['b','g']))

print('dropping columns')

print(frame4.drop(['pen','pencil'],axis=1))
```

	ball	pen	pencil	book
r	0	1	2	3
b	4	5	6	7
g	8	9	10	11
w	12	13	14	15

dropping indexes blue , green

	ball	pen	pencil	book
r	0	1	2	3
w	12	13	14	15

dropping columns

	ball	book
r	0	3

```
b      4      7
g      8     11
w     12     15
```

EXERCISE-10

#CSV data processing pandas

```
# Import pandas
import pandas as pd

# reading csv file
pd.read_csv("example1.csv")
output:
```

```
:
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2
10	10.27	1.71	Male	No	Sun	Dinner	2
11	35.26	5.00	Female	No	Sun	Dinner	4

Using usecols in read_csv()

```
df = pd.read_csv('example1.csv',

                 header=0,

                 usecols=["tip", "sex", "time"])

df
```

OUTPUT:

9]:

	tip	sex	time
0	1.01	Female	Dinner
1	1.66	Male	Dinner
2	3.50	Male	Dinner
3	3.31	Male	Dinner
4	3.61	Female	Dinner
5	4.71	Male	Dinner
6	2.00	Male	Dinner
7	3.12	Male	Dinner
8	1.96	Male	Dinner
9	3.23	Male	Dinner
10	1.71	Male	Dinner
11	5.00	Female	Dinner

Using `index_col` in `read_csv()`

Here, we use the “**sex**” index first and then the “**tip**” index, we can simply reindex the header with **`index_col`** parameter.

```
df = pd.read_csv('example1.csv',  
                 header=0,  
                 index_col=["sex", "tip"],  
                 usecols=["tip", "sex", "time"])
```

df**Output:**

[13]:

time		
sex	tip	
Female	1.01	Dinner
Male	1.66	Dinner
	3.50	Dinner
	3.31	Dinner
Female	3.61	Dinner
Male	4.71	Dinner
	2.00	Dinner
	3.12	Dinner
	1.96	Dinner
	3.23	Dinner
	1.71	Dinner
Female	5.00	Dinner

Using nrows in read_csv()

```
df = pd.read_csv('example1.csv',  
  
                 header=0,  
  
                 index_col=["tip", "sex"],  
  
                 usecols=["tip", "sex", "time"],  
  
                 nrows=5)
```

df

Output:

			time
tip	sex		
1.01	Female	Dinner	
1.66	Male	Dinner	
3.50	Male	Dinner	
3.31	Male	Dinner	
3.61	Female	Dinner	

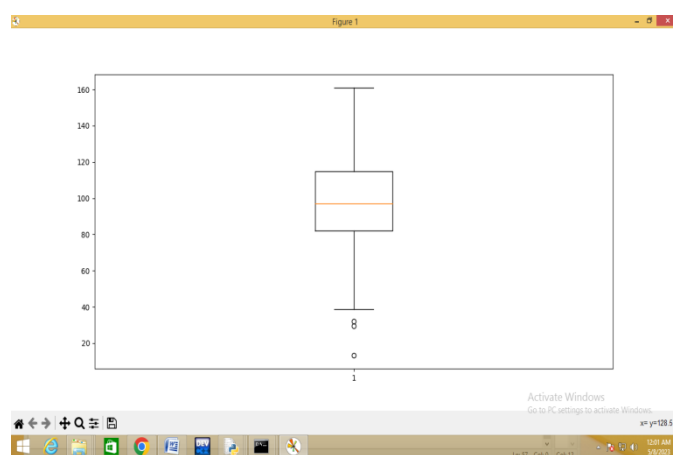
EXERCISE-11

#data visyalization using matplotlib lib

BOX PLOT:

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(15)
dataSet = np.random.normal(100, 25, 200)
print(dataSet)
figure = plt.figure(figsize =(10, 8))
plt.boxplot(dataSet)
plt.show()
```

OUTPUT:



SCATTER PLOT:

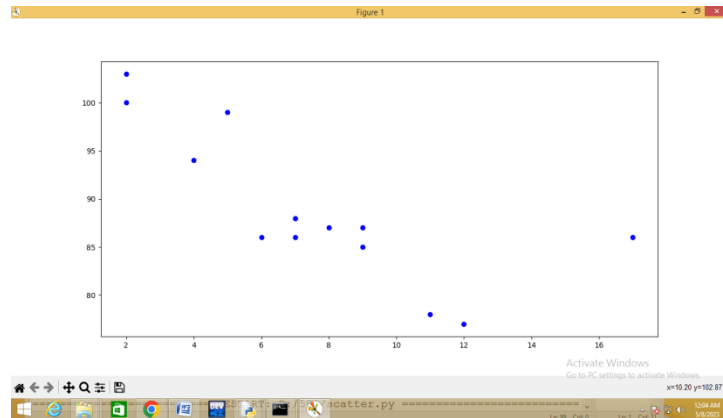
```
import matplotlib.pyplot as plt
x =[5, 7, 8, 7, 2, 17, 2, 9,
    4, 11, 12, 9, 6]
y =[99, 86, 87, 88, 100, 86,
```

```

103, 87, 94, 78, 77, 85, 86]
plt.scatter(x, y, c="blue")
plt.show()

```

OUTPUT:

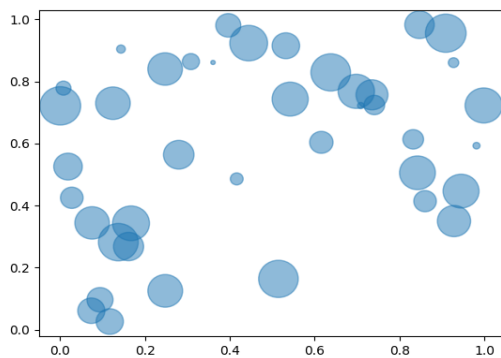


BUBBLE CHART:

```

import matplotlib.pyplot as plt
import numpy as np
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
plt.scatter(x, y, s=z*1000, alpha=0.5)
plt.show()

```



3D PLOTS:

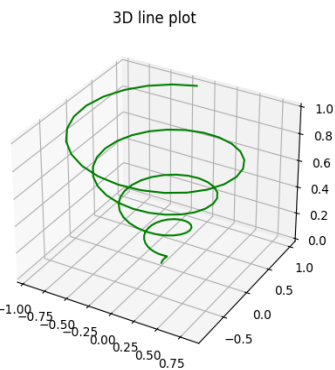
```

from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)
ax.plot3D(x, y, z, 'green')
ax.set_title('3D line plot')

```

plt.show()

OUTPUT:



EXERCISE-12

#decision tree induction algorithm

```
In [4]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = np.array(
    [['Asset Flip', 100, 1000],
    ['Text Based', 500, 3000],
    ['Visual Novel', 1500, 5000],
    ['2D Pixel Art', 3500, 8000],
    ['2D Vector Art', 5000, 6500],
    ['Strategy', 6000, 7000],
    ['First Person Shooter', 8000, 15000],
    ['Simulator', 9500, 20000],
    ['Racing', 12000, 21000],
    ['RPG', 14000, 25000],
    ['Sandbox', 15500, 27000],
    ['Open-World', 16500, 30000],
    ['MMORPG', 25000, 52000],
    ['MMORPG', 30000, 80000]
])
print(dataset)
X = dataset[:, 1:2].astype(int)
print(X)
y = dataset[:, 2].astype(int)
print(y)
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
y_pred = regressor.predict([[1500]])
print("Predicted price: % d\n" % y_pred)

X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Profit to Production Cost (Decision Tree Regression)')
plt.xlabel('Production Cost')
plt.ylabel('Profit')
plt.show()

# import export_graphviz
from sklearn.tree import export_graphviz
export_graphviz(regressor, out_file = 'tree.DOT', feature_names = ['Production Cost'])
```

```
[[['Asset Flip' '100' '1000']
[['Text Based' '500' '3000']
[['Visual Novel' '1500' '5000']
[['2D Pixel Art' '3500' '8000']
[['2D Vector Art' '5000' '6500']
[['Strategy' '6000' '7000']
[['First Person Shooter' '8000' '15000']
[['Simulator' '9500' '20000']
[['Racing' '12000' '21000']
[['RPG' '14000' '25000']
[['Sandbox' '15500' '27000']
[['Open-World' '16500' '30000']
[['MMORPG' '25000' '52000']
[['MMORPG' '30000' '80000']]

[[
100
500
1500
3500
5000
6000
8000
9500
12000
14000
15500
16500
25000
30000]]

[ 1000 3000 5000 8000 6500 7000 15000 20000 21000 25000 27000 30000
52000 80000]
Predicted price: 5000
```



EXERCISE-13

```
In [2]: import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
x = [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
plt.scatter(x, y, c=classes)
plt.show()

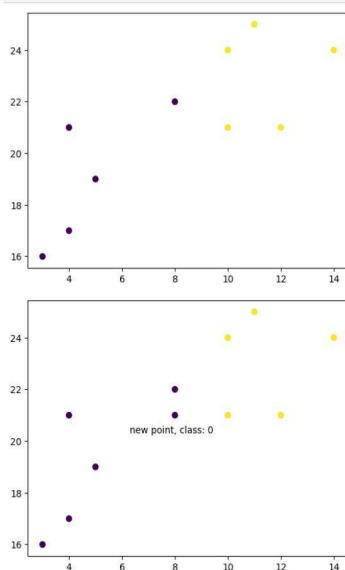
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(data, classes)

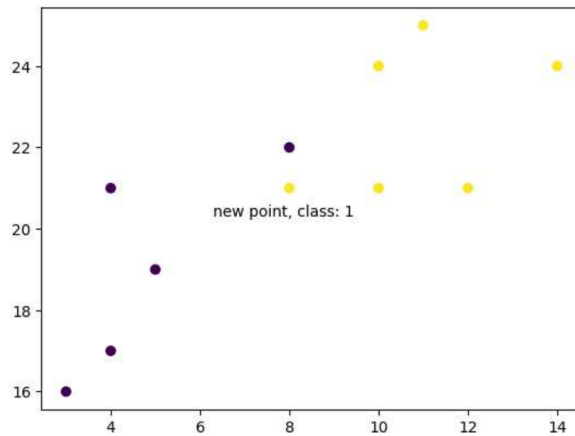
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(data, classes)
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
```





EXERCISE-14

#printing machine name and ipv4 adress

```
import socket

def print_machine_info():
    host_name = socket.gethostname()
    ip_address = socket.gethostbyname(host_name)
    print ("Host name: %s" %host_name)
    print ("IP address: %s" %ip_address)

if __name__ == '__main__':
    print_machine_info
```

output:

127.0.1.1

#creating an ipv4 adress to different formates

```
import socket

from binascii import hexlify

def convert_ip4_address():
    for ip_addr in ['127.0.0.1', '192.168.0.1']:
        packed_ip_addr = socket.
            inet_aton(ip_addr)
```

```

unpacked_ip_addr = socket.inet_ntoa
    (packed_ip_addr)
print ("IP Address: %s => Packed: %s,
    Unpacked: %s" %(ip_addr,
    hexlify(packed_ip_addr),
    unpacked_ip_addr))

```

```

if __name__ == '__main__':
    convert_ip4_address()

```

output:

```

IP Address: 127.0.0.1 => Packed: 7f000001, Unpacked:
127.0.0.1

```

```

IP Address: 192.168.0.1 => Packed: c0a80001, Unpacked: 192.168.0.1

```

#finding service name ,given the port and protoicol

```

def find_service_name():
    protocolname = 'tcp'
    for port in [80, 25]:
        print ("Port: %s => service name: %s" %(port, socket.getservbyport(port,
        protocolname)))

```

```

    print ("Port: %s => service name: %s" %(53, socket.getservbyport(53,
    'udp')))

```

```

if __name__ == '__main__':
    find_service_name()

```

output:

```

Port: 80 => service name: http

```

```

Port: 25 => service name: smtp

```

Port: 53 => service name: domain

#convert integers to from host to network:

```
import socket
```

```
def convert_integer():
```

```
    data = 1234
```

```
    # 32-bit
```

```
    print ("Original: %s => Long  host byte order: %s, Network byte order: %s"
%(data, socket.ntohl(data), socket.htonl(data)))
```

```
    # 16-bit
```

```
    print ("Original: %s => Short  host byte order: %s, Network byte order: %s"
%(data, socket.ntohs(data), socket.htons(data)))
```

```
if __name__ == '__main__':
```

```
    convert_integer()
```

output:

Original: 1234 => Long host byte order: 3523477504,

Network byte order: 3523477504

Original: 1234 => Short host byte order: 53764,

Network byte order: 53764

EXERCISE-15

#simple echo client /server application

Server.py:

```
import socket
```

```
def server_program():
```

```
    # get the hostname
```

```
    host = socket.gethostname()
```

```
    port = 5000
```

```

# initiate port no above 1024

server_socket = socket.socket()

# get instance

# look closely.The bind() function takes tuple as argument server_socket.bind((host, port)) # bind
host address and port together # configure how many client the server can listen simultaneously
server_socket.listen()

conn, address = server_socket.accept()

# accept new connection

print("Connection from: " + str(address))

while True:

    # receive data stream. it won't accept data packet greater than 1024 bytes data =
conn.recv(1024).decode() if not data: # if data is not received break break

    print("from connected user: " + str(data))

    data = input(' -> ')

    conn.send(data.encode())

# send data to the client

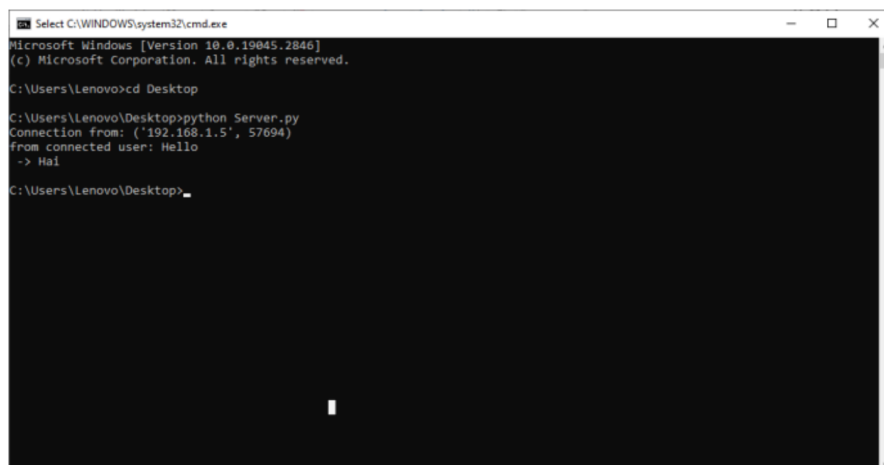
conn.close()

# close the connection

if __name__ == '__main__':

    server_program()

```



```

Select C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd Desktop

C:\Users\Lenovo\Desktop>python Server.py
Connection from: ('192.168.1.5', 57694)
from connected user: Hello
-> Hai
C:\Users\Lenovo\Desktop>_

```

Client.py:

```
import socket
```

```

def client_program():
    host = socket.gethostname()

    # as both code is running on same pc

    port = 5000

    # socket server port number

    client_socket = socket.socket()

# instantiate
client_socket.connect((host, port))

# connect to the server

message = input(" -> ")

# take input while
message.lower().strip() != 'bye':
    client_socket.send(message.encode())

# send message

data = client_socket.recv(1024).decode()

# receive response

print('Received from server: ' + data)

# show in terminal

message = input(" -> ")

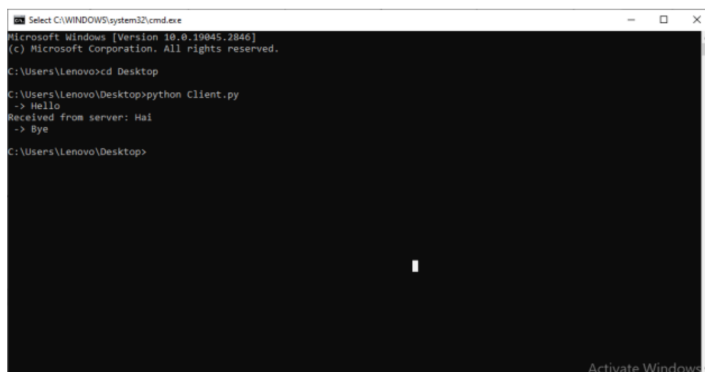
# again take input

client_socket.close()

# close the connection

if __name__ == '__main__':
    client_program()

```



```

Select C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>cd Desktop
C:\Users\Lenovo\Desktop>python Client.py
-> Hello
Received from server: Hai
-> Bye
C:\Users\Lenovo\Desktop>

```

