MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD WORK

ON

20MCA241 – DATA SCIENCE LAB

Submitted by

____

Regno: VDA21MCA-20___

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING, VADAKARA

(CAPE - GOVT. OF KERALA)

DECEMBER 2022

Department of Computer Applications
College of Engineering, Vadakara
(CAPE - Govt. of Kerala)

CERTIFICATE

This is to certify that this is a bonafide record of the practical work done on the course 20MCA241 Data Science lab done by ____________________________ **(Regno: VDA21MCA-20___)** Third semester, MCA (2021- 2023 batch), Department of Computer Applications at College of Engineering, Vadakara in partial fulfilment for the award of the degree of Master of Computer Application of APJ Abdul Kalam Technological University (KTU).

Faculty-in-Charge **Head of the Department**

Internal Examiner **External Examiner**

INDEX

Experiment No. 1

Aim:

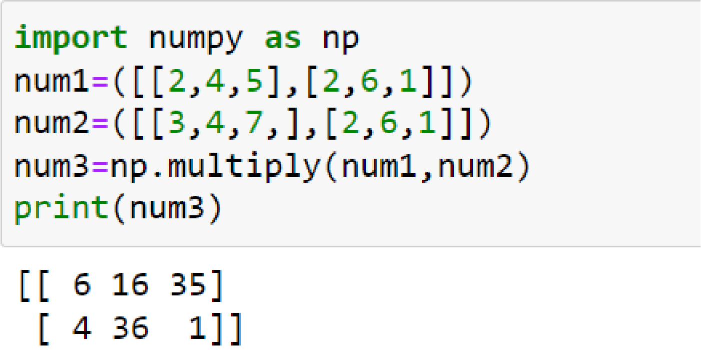
Review of python programming, Programs using NumPy, Programs using matplotlib for data visualisation and programs to handle data using pandas.

Source Code:

Review of python programming:

NumPy

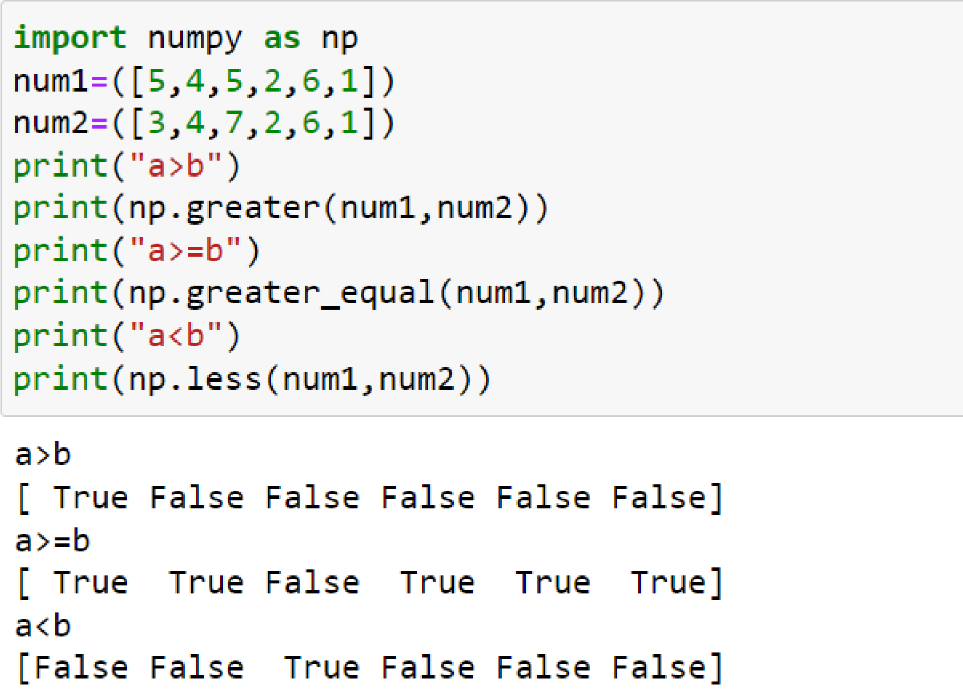
1. Program to multiply two given arrays of same size element-by-element.



```
import numpy as np
num1=([[2,4,5],[2,6,1]])
num2=([[3,4,7,],[2,6,1]])
num3=np.multiply(num1,num2)
print(num3)
```

```
[[ 6 16 35]
 [ 4 36  1]]
```

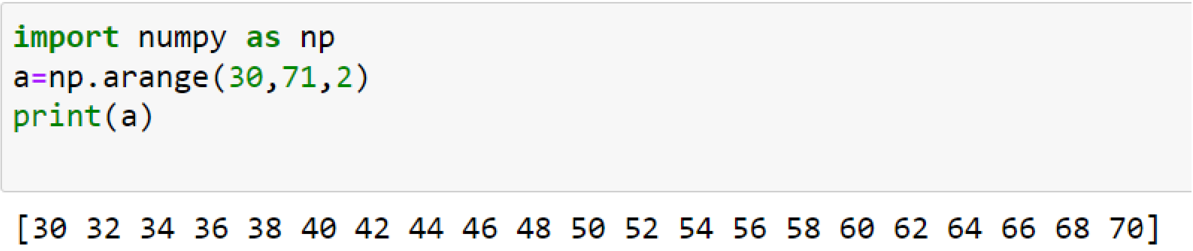
2. Program to create an element-wise comparison (greater, greater equal, lesser).



```
import numpy as np
num1=([5,4,5,2,6,1])
num2=([3,4,7,2,6,1])
print("a>b")
print(np.greater(num1,num2))
print("a>=b")
print(np.greater_equal(num1,num2))
print("a<b")
print(np.less(num1,num2))
```

```
a>b
[ True False False False False False]
a>=b
[ True  True False  True  True  True]
a<b
[False False  True False False False]
```

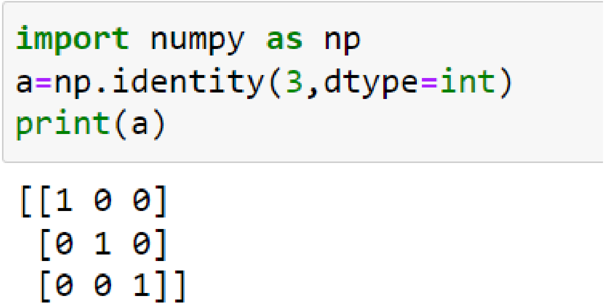
3. Program to create an array of all the even integers from 30 to 70.



```
import numpy as np
a=np.arange(30,71,2)
print(a)
```

```
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
```

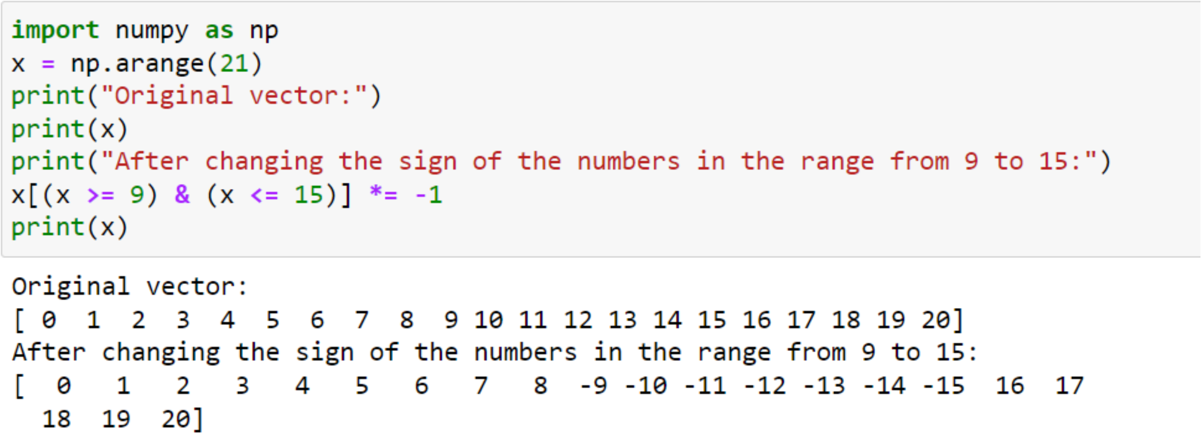
4. Program to create a 3x3 identity matrix.



```
import numpy as np
a=np.identity(3,dtype=int)
print(a)
```

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

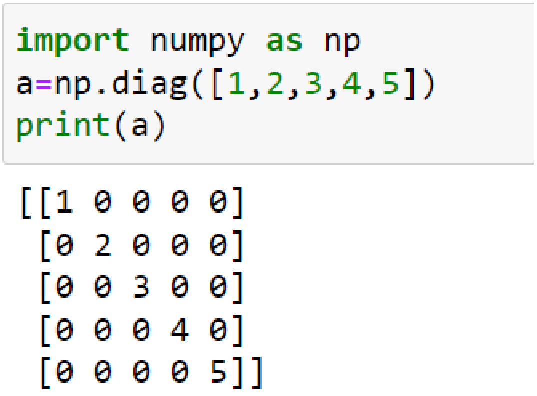
5. Program to create a vector with values from 0 to 20 and change the sign change the sign of the numbers in the range from 9 to 15.



```
import numpy as np
x = np.arange(21)
print("Original vector:")
print(x)
print("After changing the sign of the numbers in the range from 9 to 15:")
x[(x >= 9) & (x <= 15)] *= -1
print(x)
```

```
Original vector:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
After changing the sign of the numbers in the range from 9 to 15:
[  0   1   2   3   4   5   6   7   8  -9 -10 -11 -12 -13 -14 -15  16  17
  18  19  20]
```

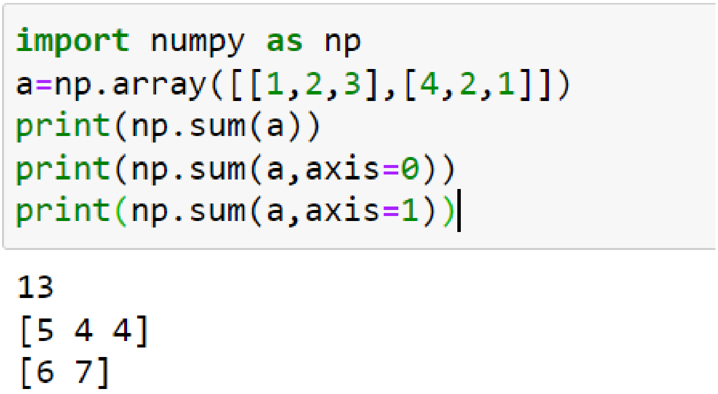
6. Program to create a 5x5 zero matrix with elements on the main diagonal.



```
import numpy as np
a=np.diag([1,2,3,4,5])
print(a)
```

```
[[1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]
 [0 0 0 0 5]]
```

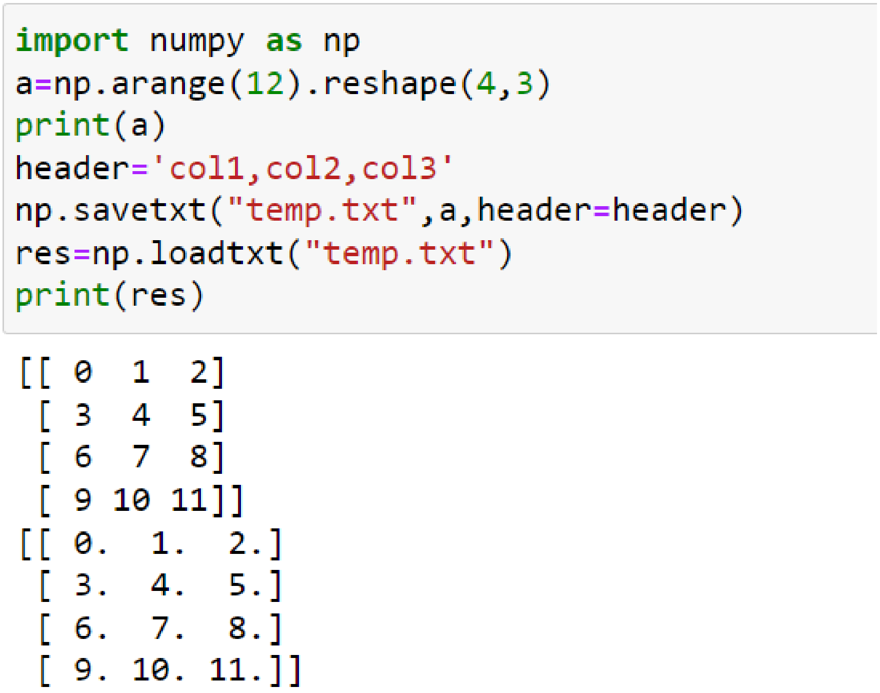
7. Program to compute sum of all elements, sum of each column and sum of each row.



```
import numpy as np
a=np.array([[1,2,3],[4,2,1]])
print(np.sum(a))
print(np.sum(a,axis=0))
print(np.sum(a,axis=1))
```

```
13
[5 4 4]
[6 7]
```

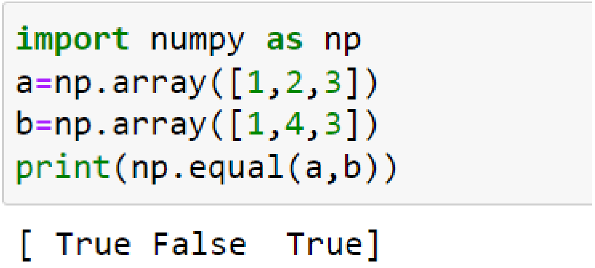
8. Program to save a given array to a text file and load it.



```
import numpy as np
a=np.arange(12).reshape(4,3)
print(a)
header='col1,col2,col3'
np.savetxt("temp.txt",a,header=header)
res=np.loadtxt("temp.txt")
print(res)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[ 0.  1.  2.]
 [ 3.  4.  5.]
 [ 6.  7.  8.]
 [ 9. 10. 11.]]
```

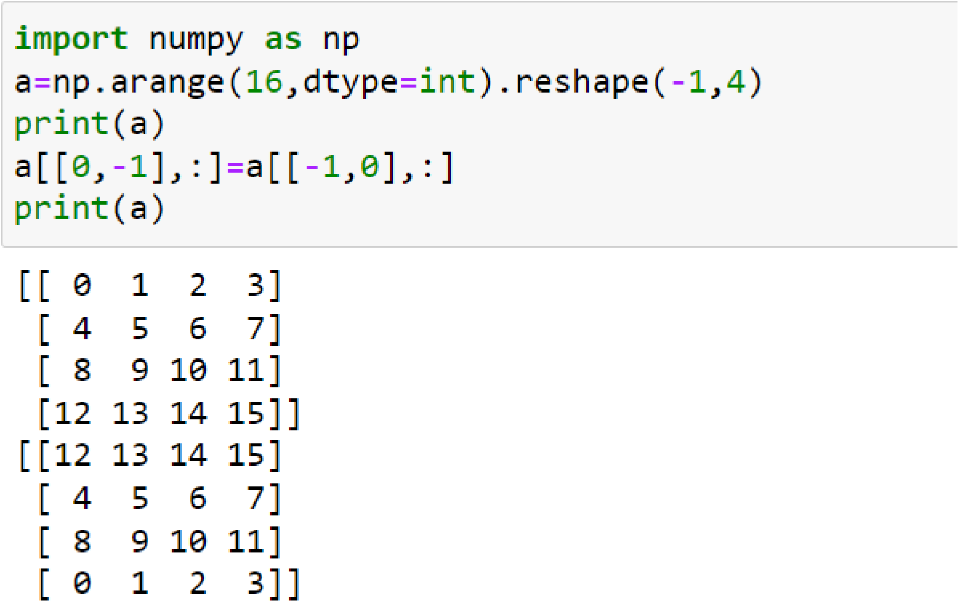
9. Program to check whether two arrays are equal (element wise) or not.



```
import numpy as np
a=np.array([1,2,3])
b=np.array([1,4,3])
print(np.equal(a,b))
```

```
[ True False  True]
```

10. Program to create a 4x4 array with random values, now create a new array after swapping first and last row.

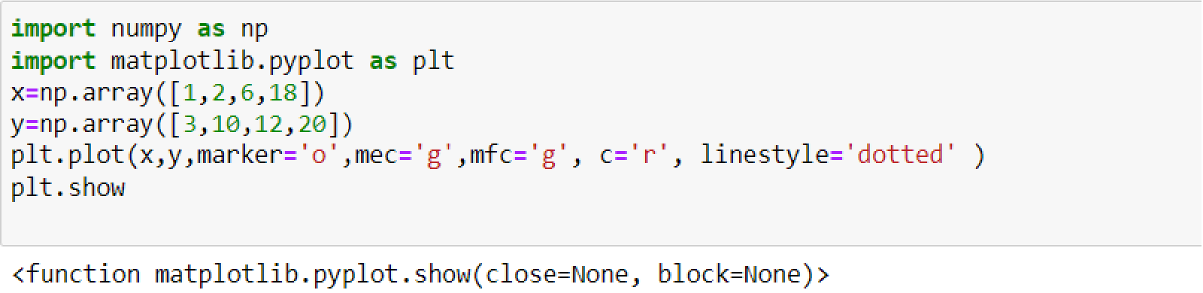


```
import numpy as np
a=np.arange(16,dtype=int).reshape(-1,4)
print(a)
a[[0,-1],:]=a[[-1,0],:]
print(a)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[12 13 14 15]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [ 0  1  2  3]]
```

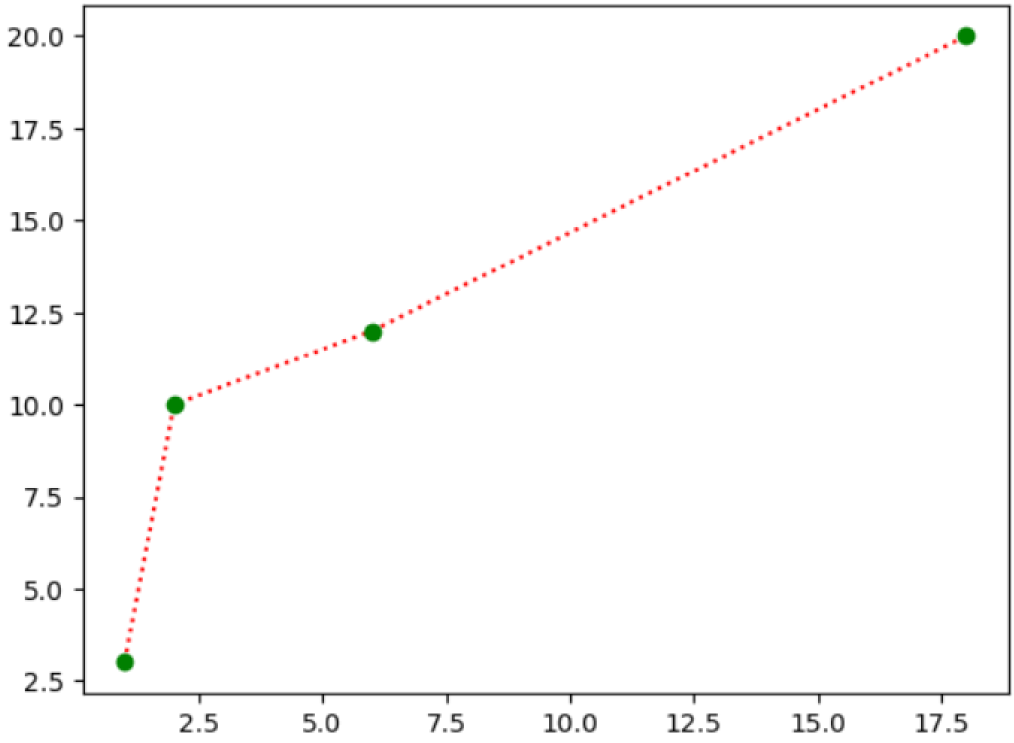
Matplotlib

1. Draw a line in a diagram from position (1, 3) to (2, 10) then to (6, 12) and finally to position (18, 20). (Mark each point with a beautiful green color and set line color to red and line style dotted).



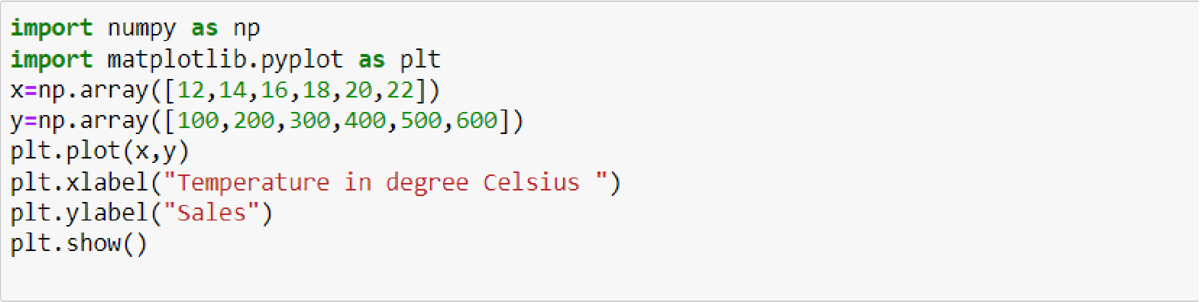
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,6,18])
y=np.array([3,10,12,20])
plt.plot(x,y,marker='o',mec='g',mfc='g', c='r', linestyle='dotted' )
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

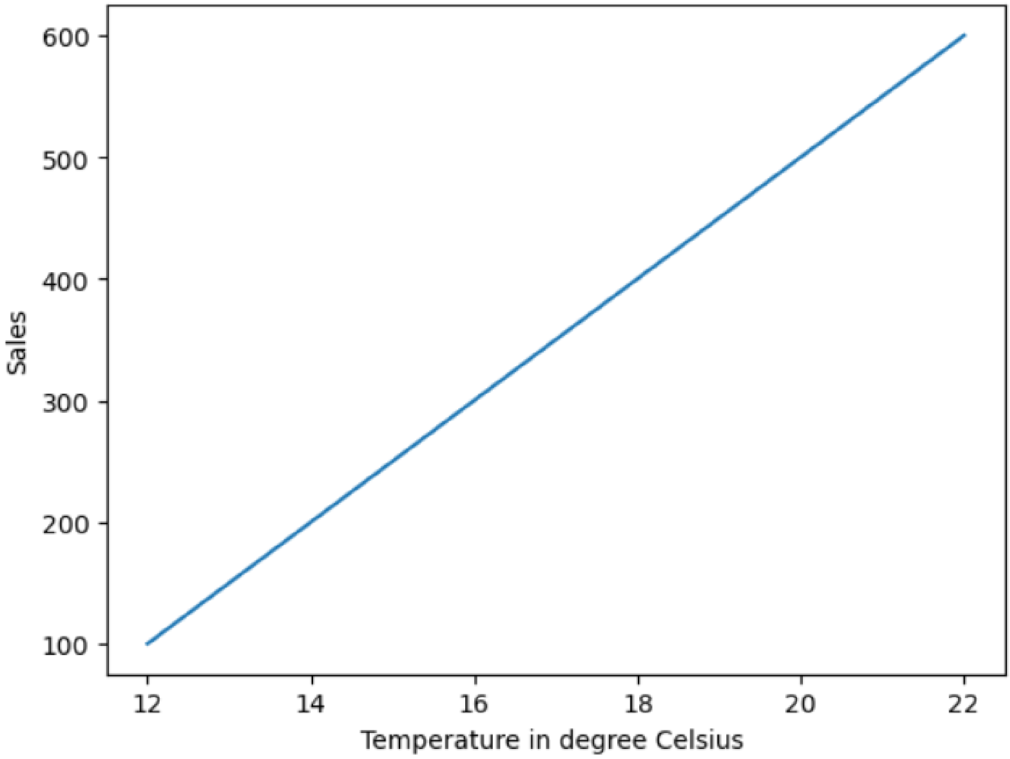


2. Draw a plot for the following data:

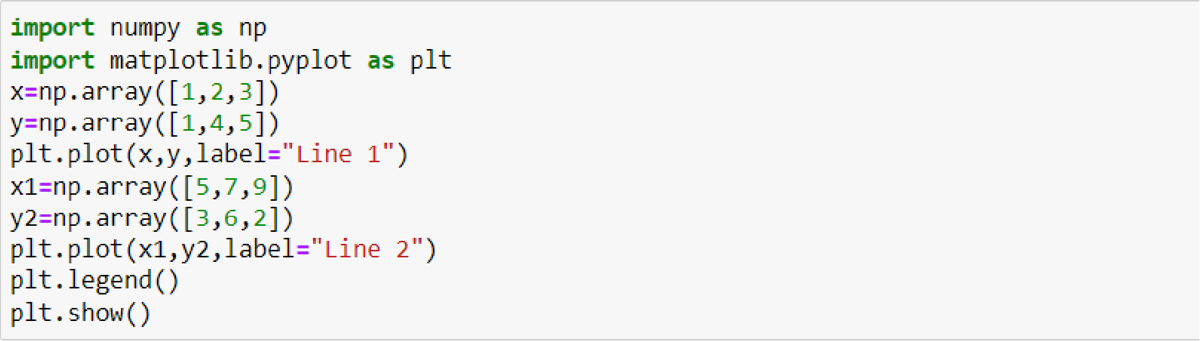
Temperature in degree Celsius,	Sales
12	100
14	200
16	250
18	400
20	300
22	450
24	500



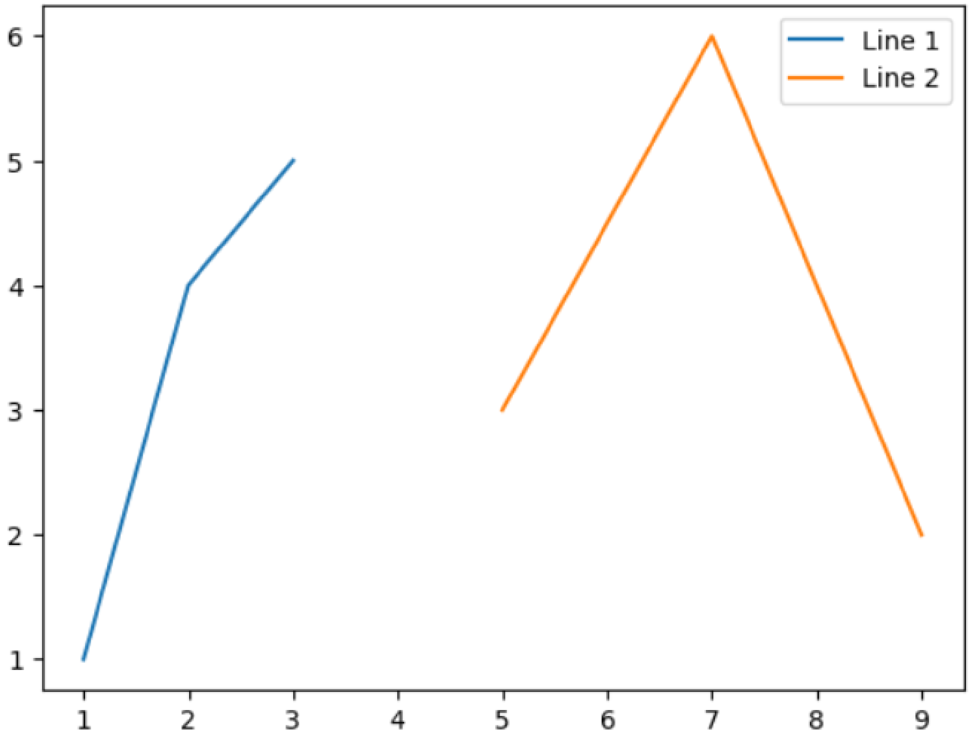
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([12,14,16,18,20,22])
y=np.array([100,200,300,400,500,600])
plt.plot(x,y)
plt.xlabel("Temperature in degree Celsius ")
plt.ylabel("Sales")
plt.show()
```



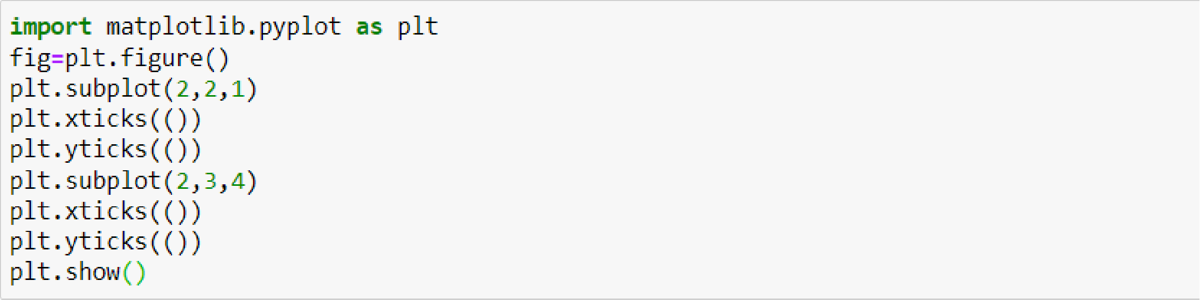
3. Write a Python program to plot two or more lines on same plot with suitable legends of each line.



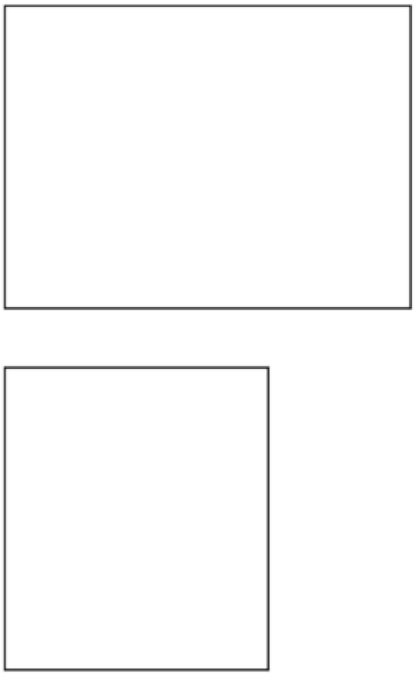
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,3])
y=np.array([1,4,5])
plt.plot(x,y,label="Line 1")
x1=np.array([5,7,9])
y2=np.array([3,6,2])
plt.plot(x1,y2,label="Line 2")
plt.legend()
plt.show()
```



4. Write a Python program to create multiple plots.



```
import matplotlib.pyplot as plt
fig=plt.figure()
plt.subplot(2,2,1)
plt.xticks(())
plt.yticks(())
plt.subplot(2,3,4)
plt.xticks(())
plt.yticks(())
plt.show()
```



5. Consider the following data.

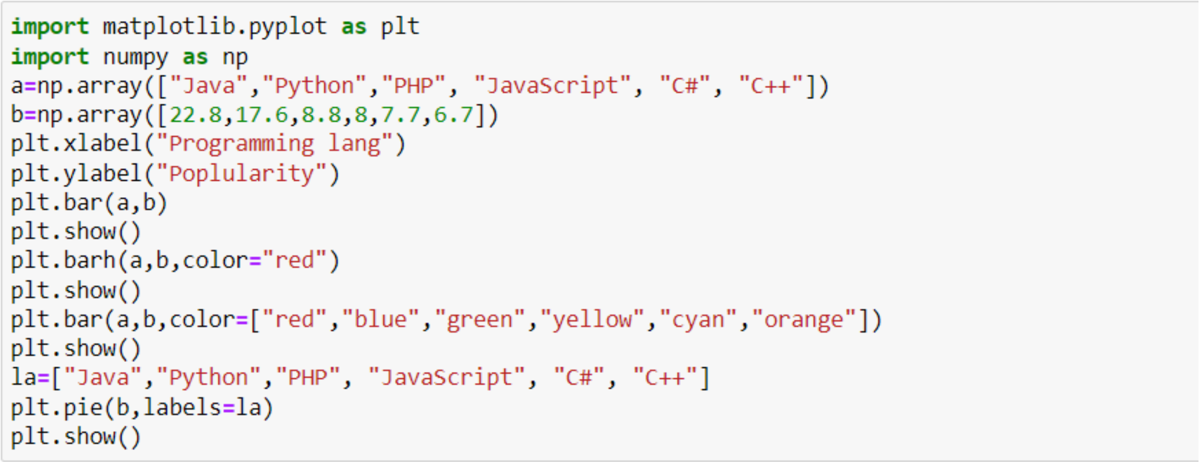
Programming languages:	Java	Python	PHP	JavaScript	C#	C++
Popularity	22.2	17.6	8.8	8	7.7	6.7

(i)Write a Python programming to display a bar chart of the popularity of programming Languages.

(ii)Write a Python programming to display a horizontal bar chart of the popularity of programming Languages (Give Red color to the bar chart)

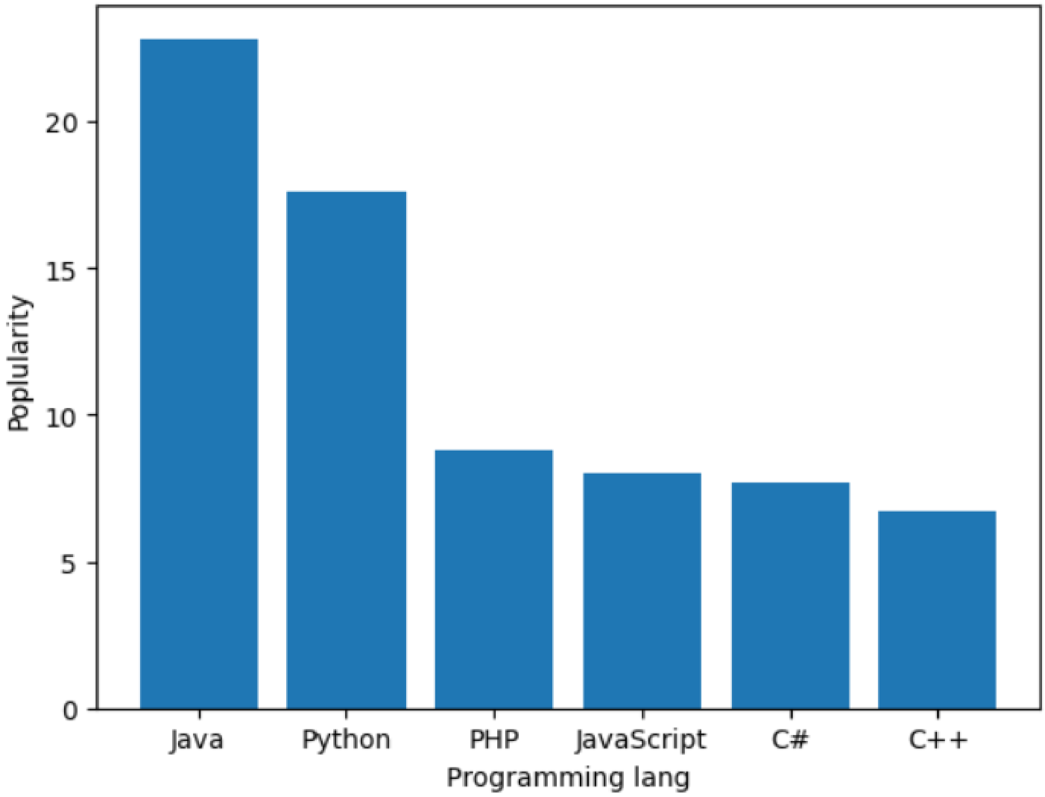
(iii)Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.

(iv)Write a Python programming to display a pie chart of the popularity of programming Languages.



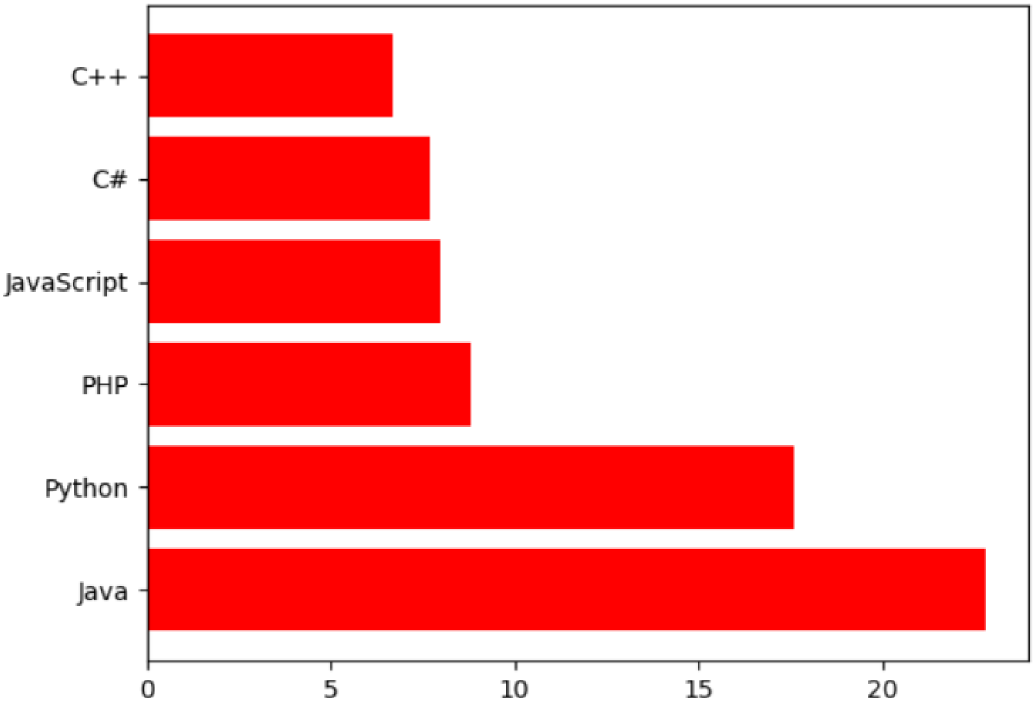
```
import matplotlib.pyplot as plt
import numpy as np
a=np.array(["Java","Python","PHP", "JavaScript", "C#", "C++"])
b=np.array([22.8,17.6,8.8,8,7.7,6.7])
plt.xlabel("Programming lang")
plt.ylabel("Poplularity")
plt.bar(a,b)
plt.show()
plt.barh(a,b,color="red")
plt.show()
plt.bar(a,b,color=["red","blue","green","yellow","cyan","orange"])
plt.show()
la=["Java","Python","PHP", "JavaScript", "C#", "C++"]
plt.pie(b,labels=la)
plt.show()
```

(i)



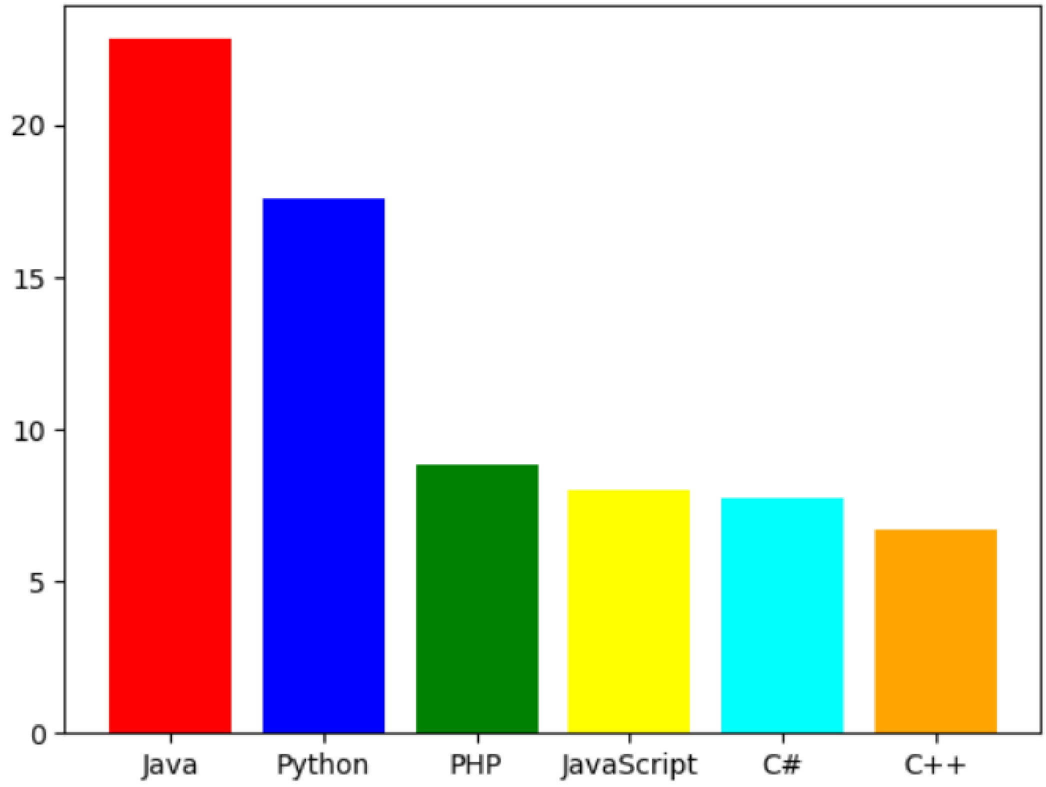
Poplularity
0
5
10
15
20
Java
Python
PHP
JavaScript
C#
C++
Programming lang

(ii)

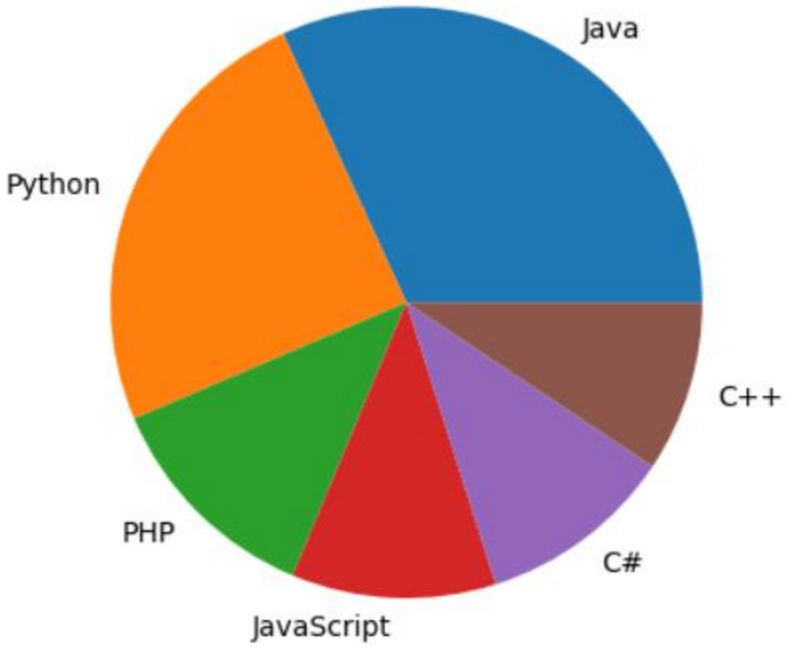


C++
C#
JavaScript
PHP
Python
Java
0
5
10
15
20

(iii)


20
15
10
5
0
Java
Python
PHP
JavaScript
C#
C++

(iv)

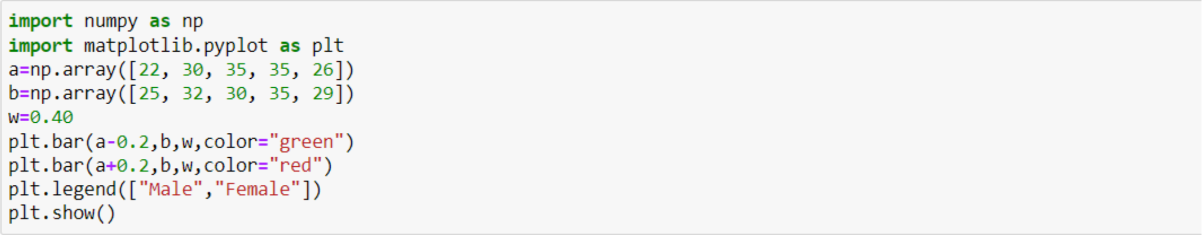

Java
Python
C++
PHP
C#
JavaScript

6. Write a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.

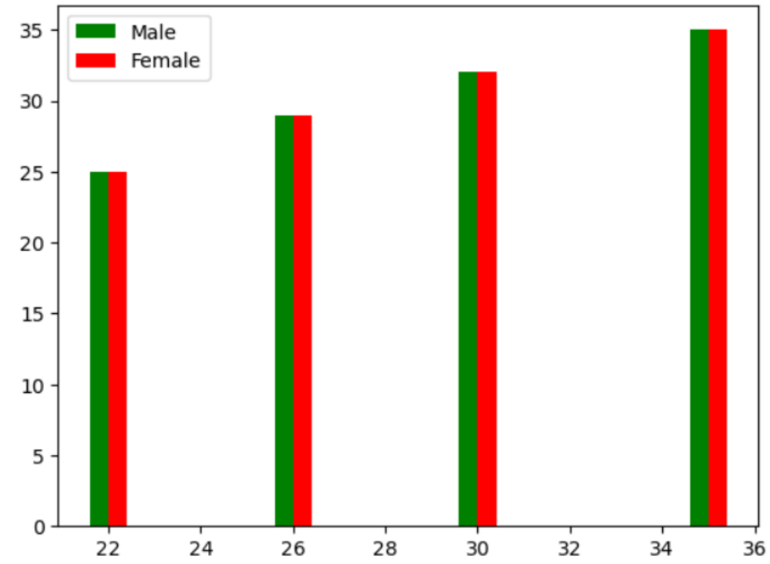
Sample Data:

Means (men) = (22, 30, 35, 35, 26)

Means (women) = (25, 32, 30, 35, 29)

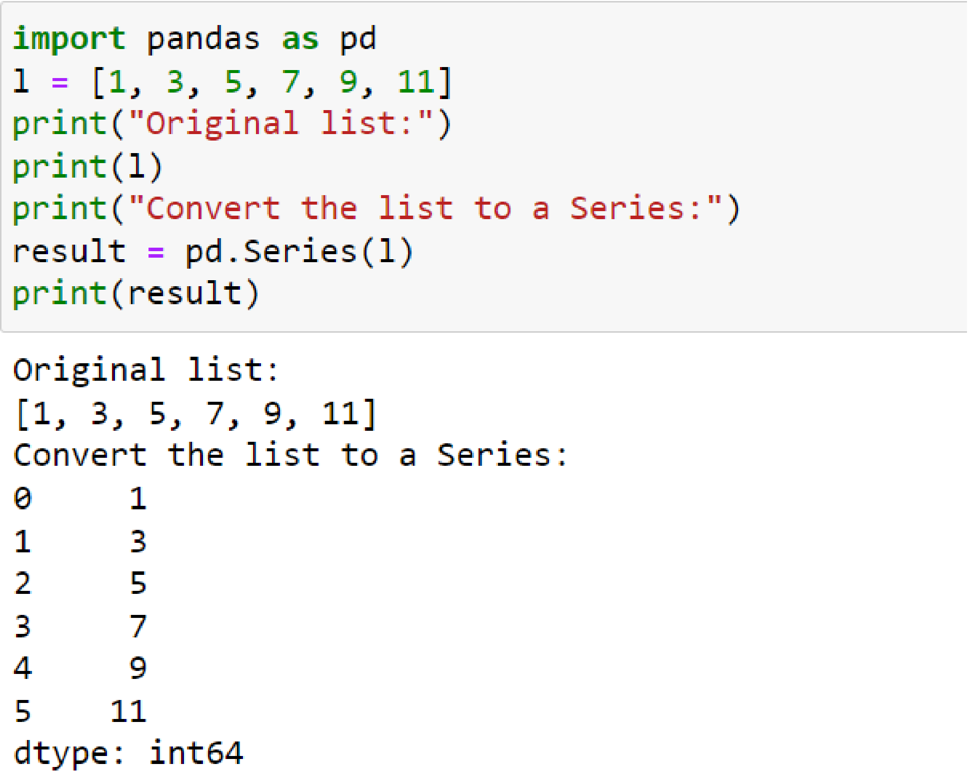


```
import numpy as np
import matplotlib.pyplot as plt
a=np.array([22, 30, 35, 35, 26])
b=np.array([25, 32, 30, 35, 29])
w=0.40
plt.bar(a-0.2,b,w,color="green")
plt.bar(a+0.2,b,w,color="red")
plt.legend(["Male","Female"])
plt.show()
```



Pandas

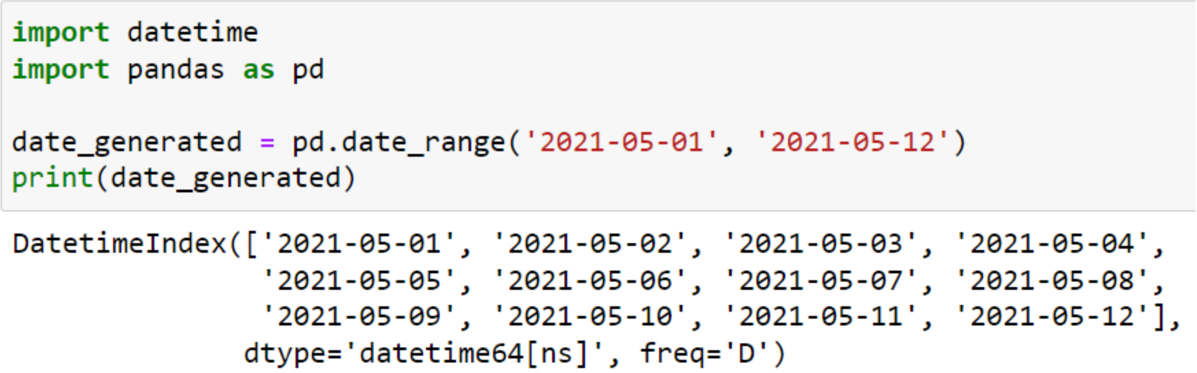
1. Write a python program to implement List-to-Series Conversion.



```
import pandas as pd
l = [1, 3, 5, 7, 9, 11]
print("Original list:")
print(l)
print("Convert the list to a Series:")
result = pd.Series(l)
print(result)
```

```
Original list:
[1, 3, 5, 7, 9, 11]
Convert the list to a Series:
0      1
1      3
2      5
3      7
4      9
5     11
dtype: int64
```

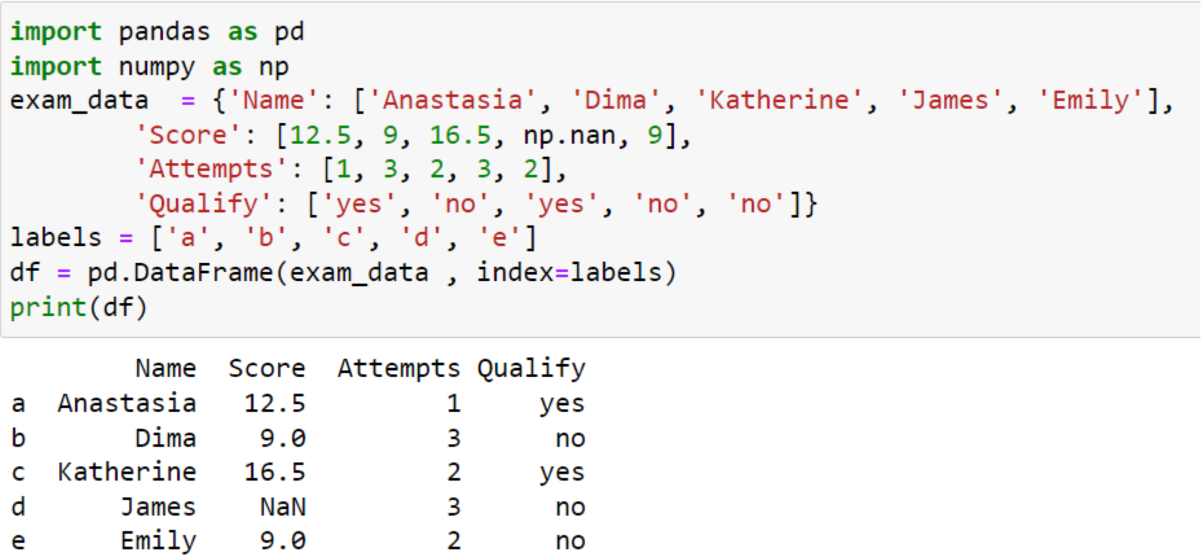
2. Write a python program to Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive).



```
import datetime
import pandas as pd

date_generated = pd.date_range('2021-05-01', '2021-05-12')
print(date_generated)
```

```
DatetimeIndex(['2021-05-01', '2021-05-02', '2021-05-03', '2021-05-04',
               '2021-05-05', '2021-05-06', '2021-05-07', '2021-05-08',
               '2021-05-09', '2021-05-10', '2021-05-11', '2021-05-12'],
              dtype='datetime64[ns]', freq='D')
```

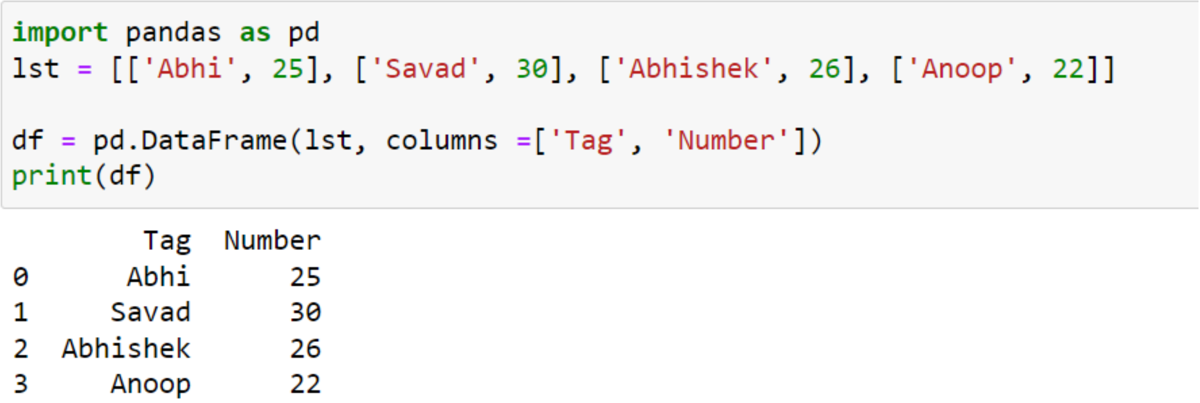
3. Given a dictionary, convert it into corresponding data frame and display it.



```
import pandas as pd
import numpy as np
exam_data  = {'Name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily'],
        'Score': [12.5, 9, 16.5, np.nan, 9],
        'Attempts': [1, 3, 2, 3, 2],
        'Qualify': ['yes', 'no', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e']
df = pd.DataFrame(exam_data , index=labels)
print(df)
```

```
        Name  Score  Attempts Qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
```

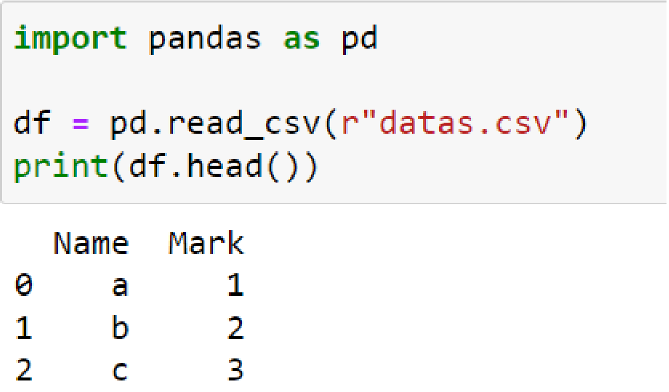
4. Given a 2D List, convert it into corresponding data frame and display it.



```
import pandas as pd
lst = [['Abhi', 25], ['Savad', 30], ['Abhishek', 26], ['Anoop', 22]]

df = pd.DataFrame(lst, columns =['Tag', 'Number'])
print(df)
```

```
        Tag  Number
0      Abhi      25
1     Savad      30
2  Abhishek      26
3     Anoop      22
```

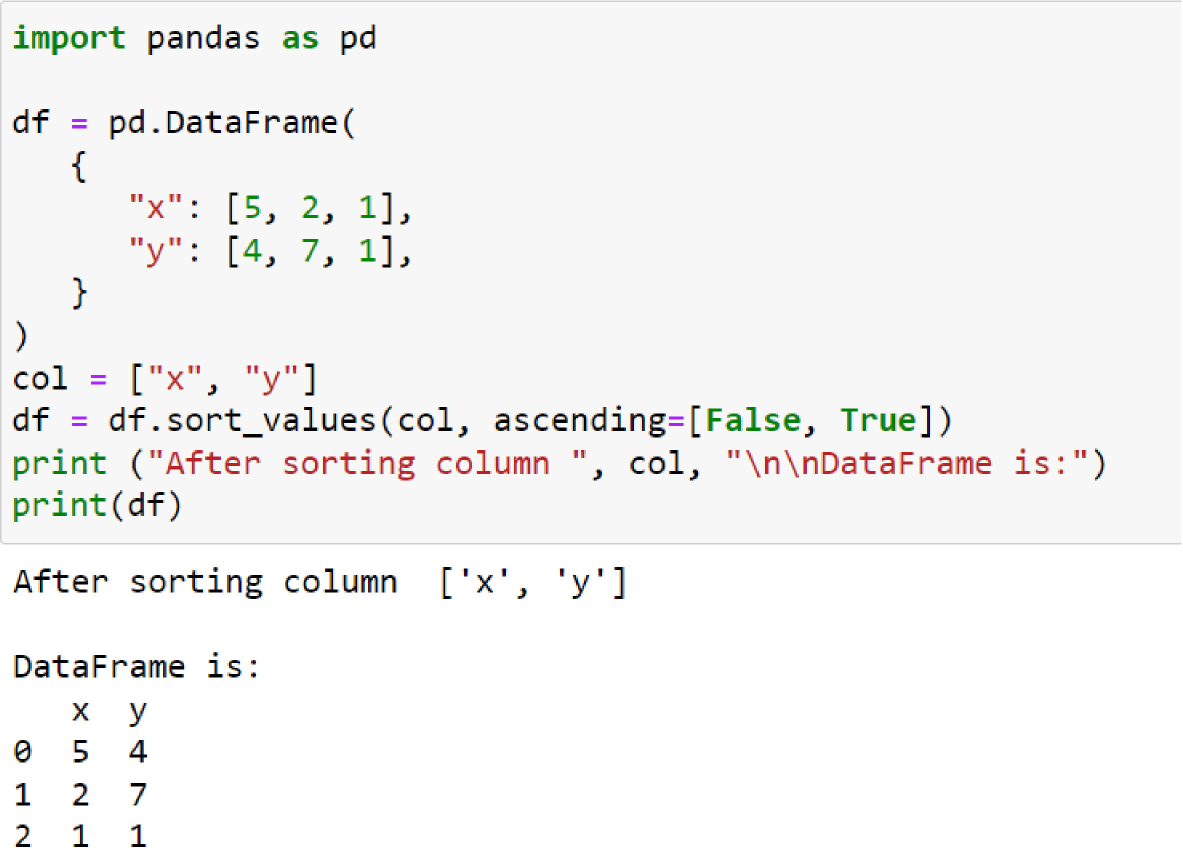
5. Given a CSV file, read it into a data frame and display it.
CSV file: Name, mark
a,1
b,2
c,3



```
import pandas as pd

df = pd.read_csv(r"datas.csv")
print(df.head())
```

```
  Name  Mark
0    a     1
1    b     2
2    c     3
```

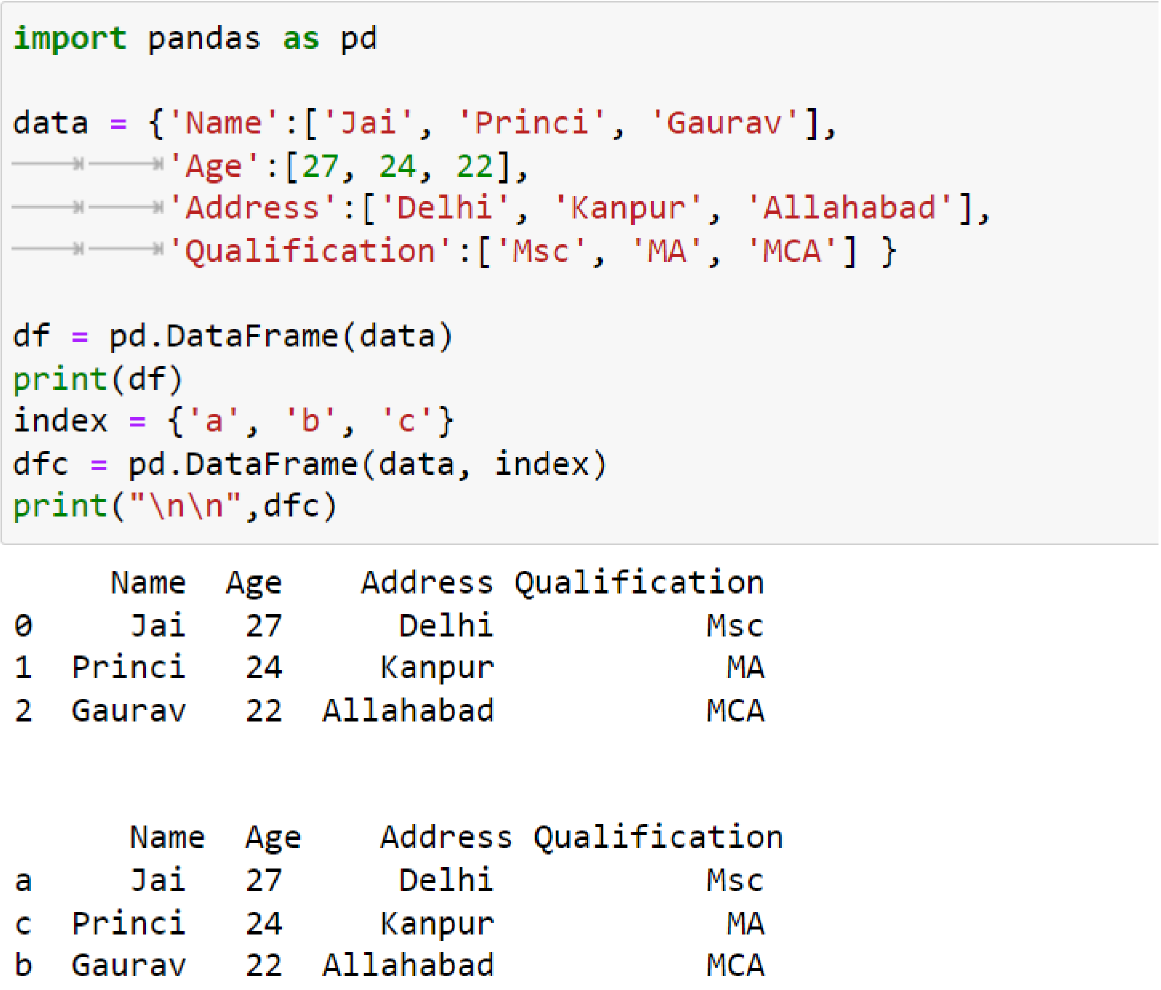
6. Given a data frame, sort it by multiple columns.



```
import pandas as pd

df = pd.DataFrame(
   {
      "x": [5, 2, 1],
      "y": [4, 7, 1],
   }
)
col = ["x", "y"]
df = df.sort_values(col, ascending=[False, True])
print ("After sorting column ", col, "\n\nDataFrame is:")
print(df)
```

```
After sorting column  ['x', 'y']

DataFrame is:
   x  y
0  5  4
1  2  7
2  1  1
```

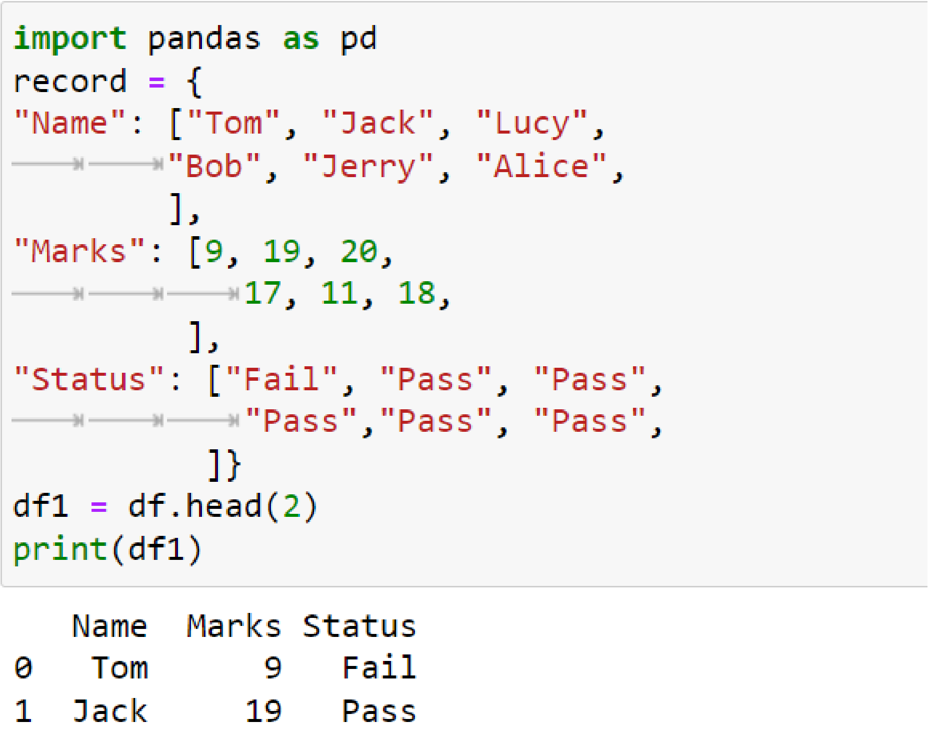
7. Given a data frame with custom indexing, convert it to default indexing and display it.



```
import pandas as pd

data = {'Name':['Jai', 'Princi', 'Gaurav'],
		'Age':[27, 24, 22],
		'Address':['Delhi', 'Kanpur', 'Allahabad'],
		'Qualification':['Msc', 'MA', 'MCA'] }

df = pd.DataFrame(data)
print(df)
index = {'a', 'b', 'c'}
dfc = pd.DataFrame(data, index)
print("\n\n",dfc)
```

```
     Name  Age    Address Qualification
0     Jai   27      Delhi           Msc
1  Princi   24     Kanpur            MA
2  Gaurav   22  Allahabad           MCA


      Name  Age    Address Qualification
a     Jai   27      Delhi           Msc
c  Princi   24     Kanpur            MA
b  Gaurav   22  Allahabad           MCA
```

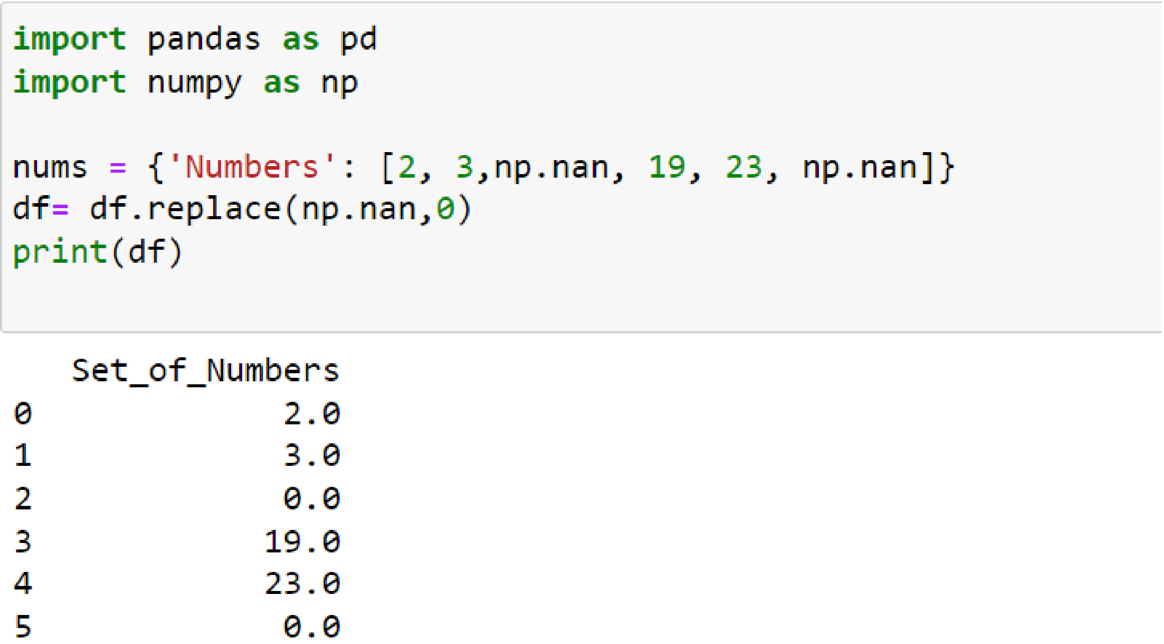
8. Given a dataframe, select first 2 rows and output them.



```
import pandas as pd
record = {
"Name": ["Tom", "Jack", "Lucy",
		"Bob", "Jerry", "Alice",
        ],
"Marks": [9, 19, 20,
			17, 11, 18,
         ],
"Status": ["Fail", "Pass", "Pass",
			"Pass","Pass", "Pass",
          ]}
df1 = df.head(2)
print(df1)
```

```
   Name  Marks Status
0   Tom      9   Fail
1  Jack     19   Pass
```

9. Given a data frame with NaN Values, fill the NaN values with 0.



```
import pandas as pd
import numpy as np

nums = {'Numbers': [2, 3,np.nan, 19, 23, np.nan]}
df= df.replace(np.nan,0)
print(df)
```

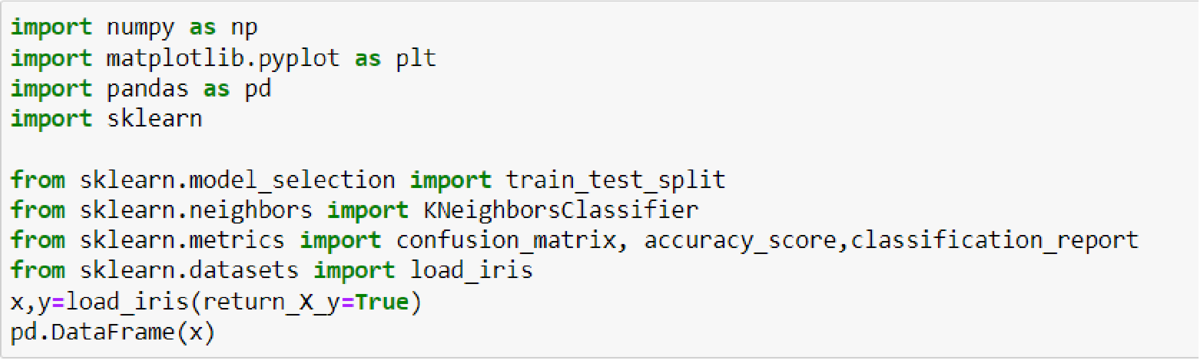
```
   Set_of_Numbers
0             2.0
1             3.0
2             0.0
3            19.0
4            23.0
5             0.0
```

Experiment No 2:

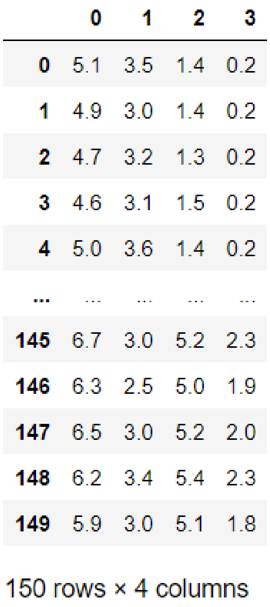
Aim:

Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

Source Code:

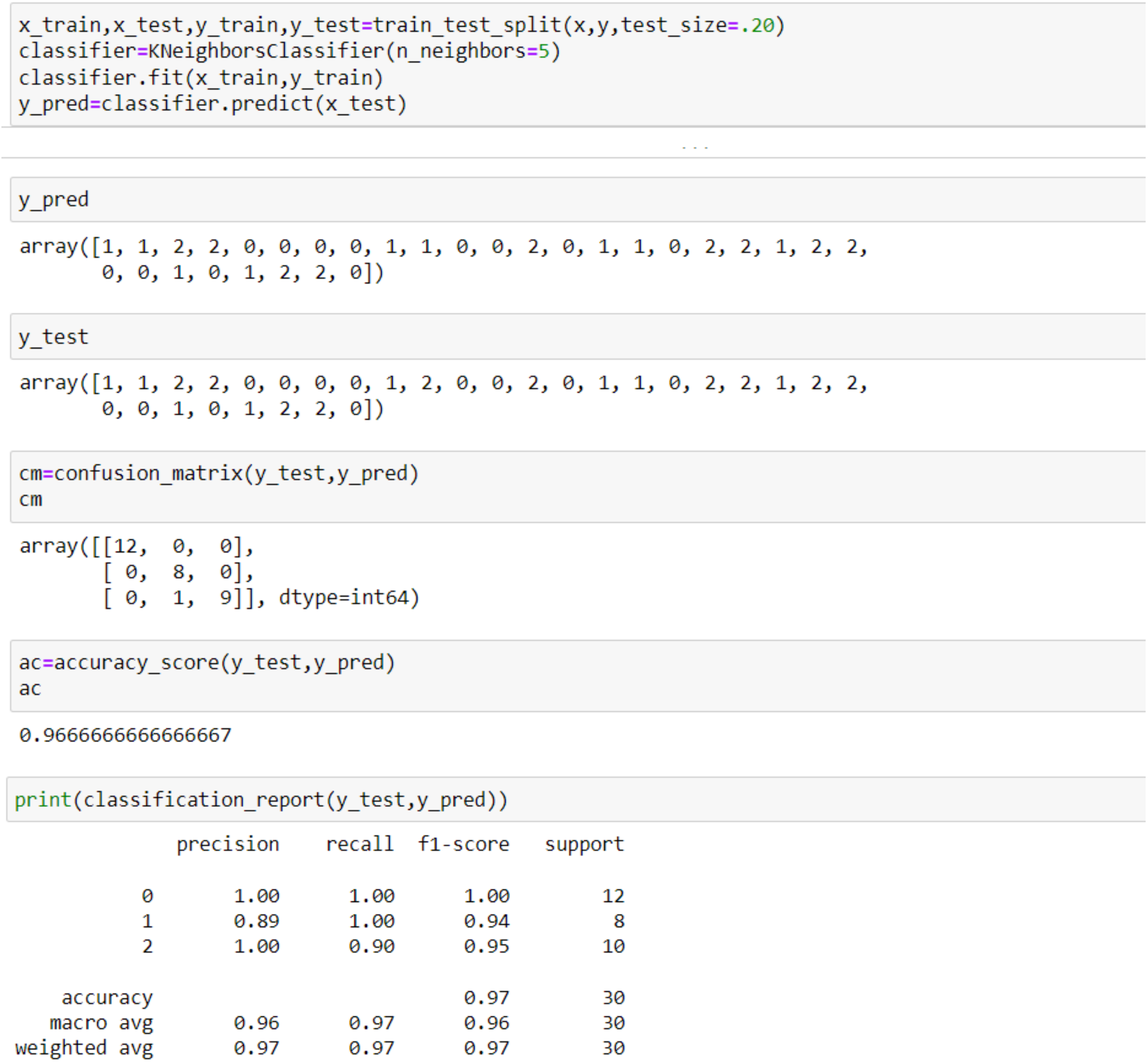


```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.datasets import load_iris
x,y=load_iris(return_X_y=True)
pd.DataFrame(x)
```



	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns



```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
```

```
y_pred
```

```
array([1, 1, 2, 2, 0, 0, 0, 0, 1, 1, 0, 0, 2, 0, 1, 1, 0, 2, 2, 1, 2, 2,
       0, 0, 1, 0, 1, 2, 2, 0])
```

```
y_test
```

```
array([1, 1, 2, 2, 0, 0, 0, 0, 1, 2, 0, 0, 2, 0, 1, 1, 0, 2, 2, 1, 2, 2,
       0, 0, 1, 0, 1, 2, 2, 0])
```

```
cm=confusion_matrix(y_test,y_pred)
cm
```

```
array([[12,  0,  0],
       [ 0,  8,  0],
       [ 0,  1,  9]], dtype=int64)
```

```
ac=accuracy_score(y_test,y_pred)
ac
```

```
0.9666666666666667
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.89      1.00      0.94         8
           2       1.00      0.90      0.95        10

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Result:

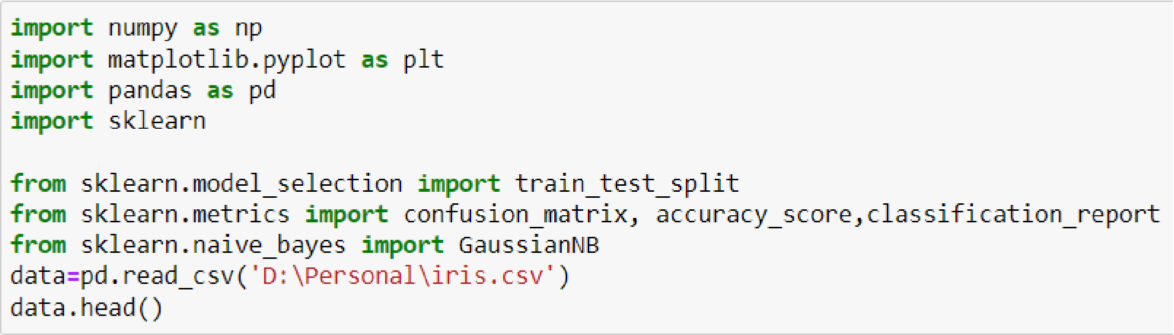
Successfully implemented k-NN classification using **iris** dataset and the accuracy of the algorithm is **0.9666666666666667**

Experiment No 3:

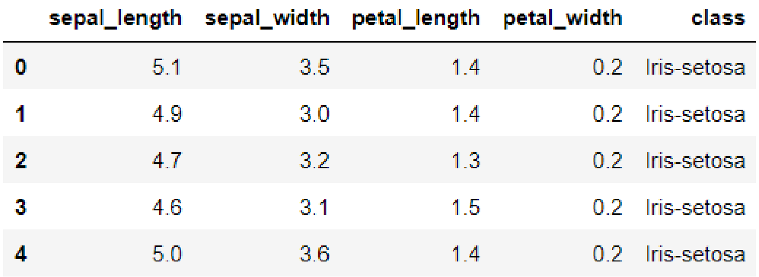
Aim:

Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

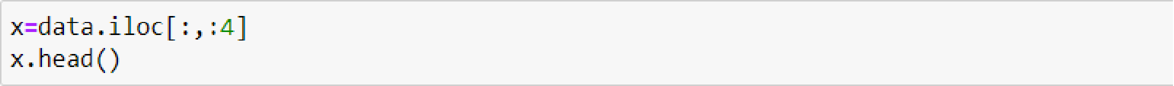
Source Code:



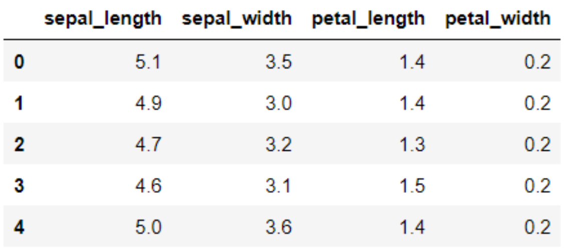
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv('D:\Personal\iris.csv')
data.head()
```



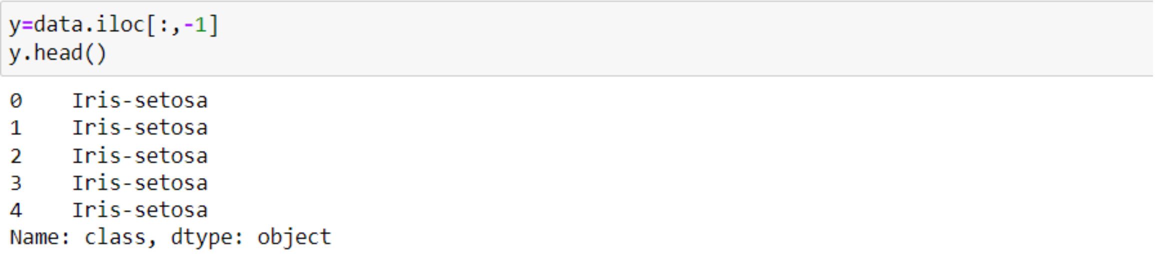
	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa



```
x=data.iloc[:,:4]
x.head()
```

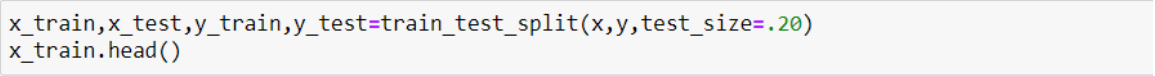


	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

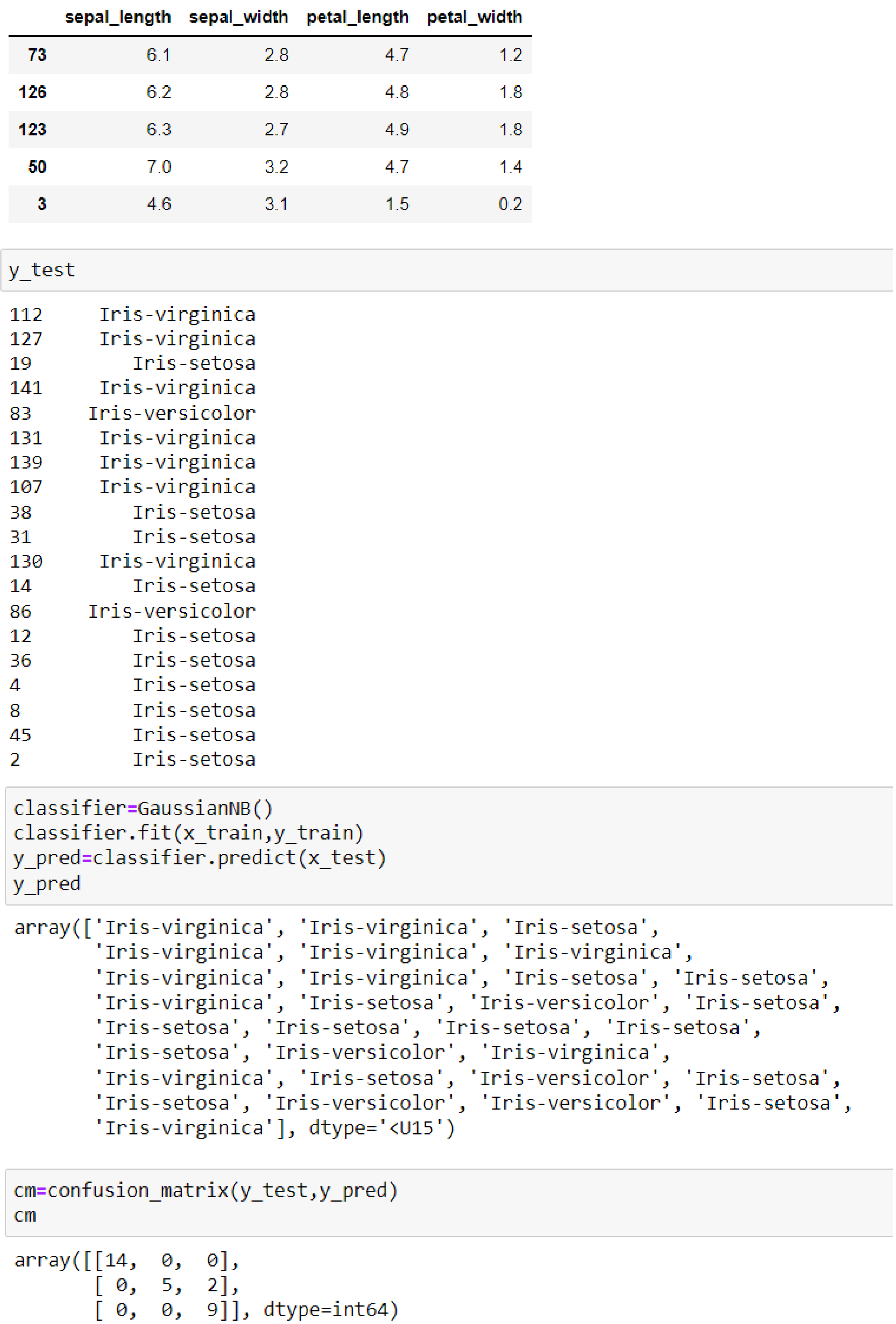


```
y=data.iloc[:,-1]
y.head()
```

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: class, dtype: object
```



```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
x_train.head()
```



	sepal_length	sepal_width	petal_length	petal_width
73	6.1	2.8	4.7	1.2
126	6.2	2.8	4.8	1.8
123	6.3	2.7	4.9	1.8
50	7.0	3.2	4.7	1.4
3	4.6	3.1	1.5	0.2

```
y_test
```

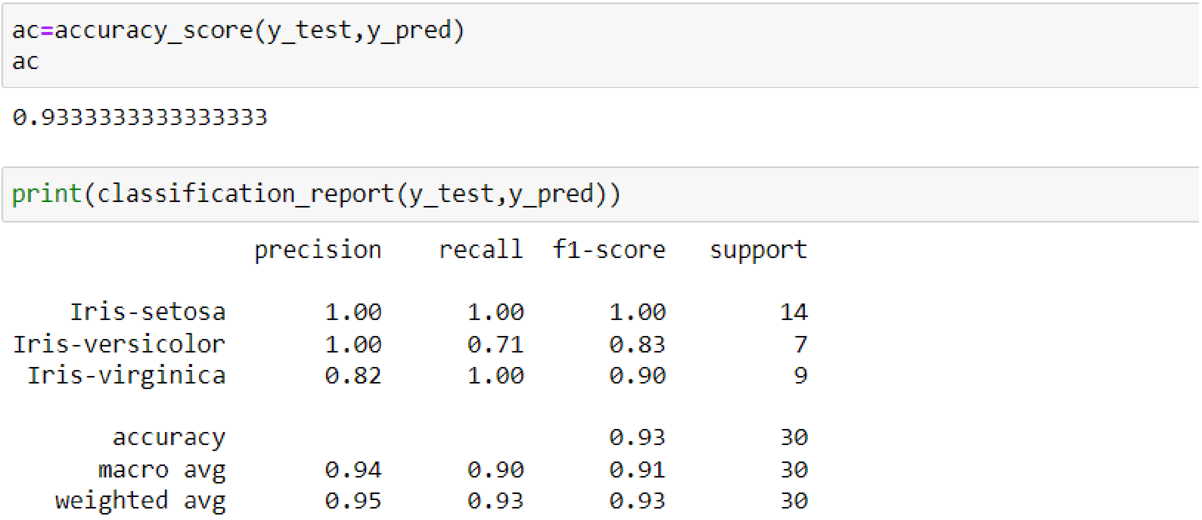
```
112       Iris-virginica
127       Iris-virginica
19           Iris-setosa
141       Iris-virginica
83       Iris-versicolor
131       Iris-virginica
139       Iris-virginica
107       Iris-virginica
38           Iris-setosa
31           Iris-setosa
130       Iris-virginica
14           Iris-setosa
86       Iris-versicolor
12           Iris-setosa
36           Iris-setosa
4            Iris-setosa
8            Iris-setosa
45           Iris-setosa
2            Iris-setosa
```

```
classifier=GaussianNB()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred
```

```
array(['Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica'], dtype='<U15')
```

```
cm=confusion_matrix(y_test,y_pred)
cm
```

```
array([[14,  0,  0],
       [ 0,  5,  2],
       [ 0,  0,  9]], dtype=int64)
```



```
ac=accuracy_score(y_test,y_pred)
ac
```

```
0.9333333333333333
```

```
print(classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.71      0.83         7
 Iris-virginica       0.82      1.00      0.90         9

       accuracy                           0.93        30
      macro avg       0.94      0.90      0.91        30
   weighted avg       0.95      0.93      0.93        30
```

Result:

Successfully implemented Naïve Bayes Algorithm using **iris** dataset and the accuracy of the algorithm is **0.9333333333333333**

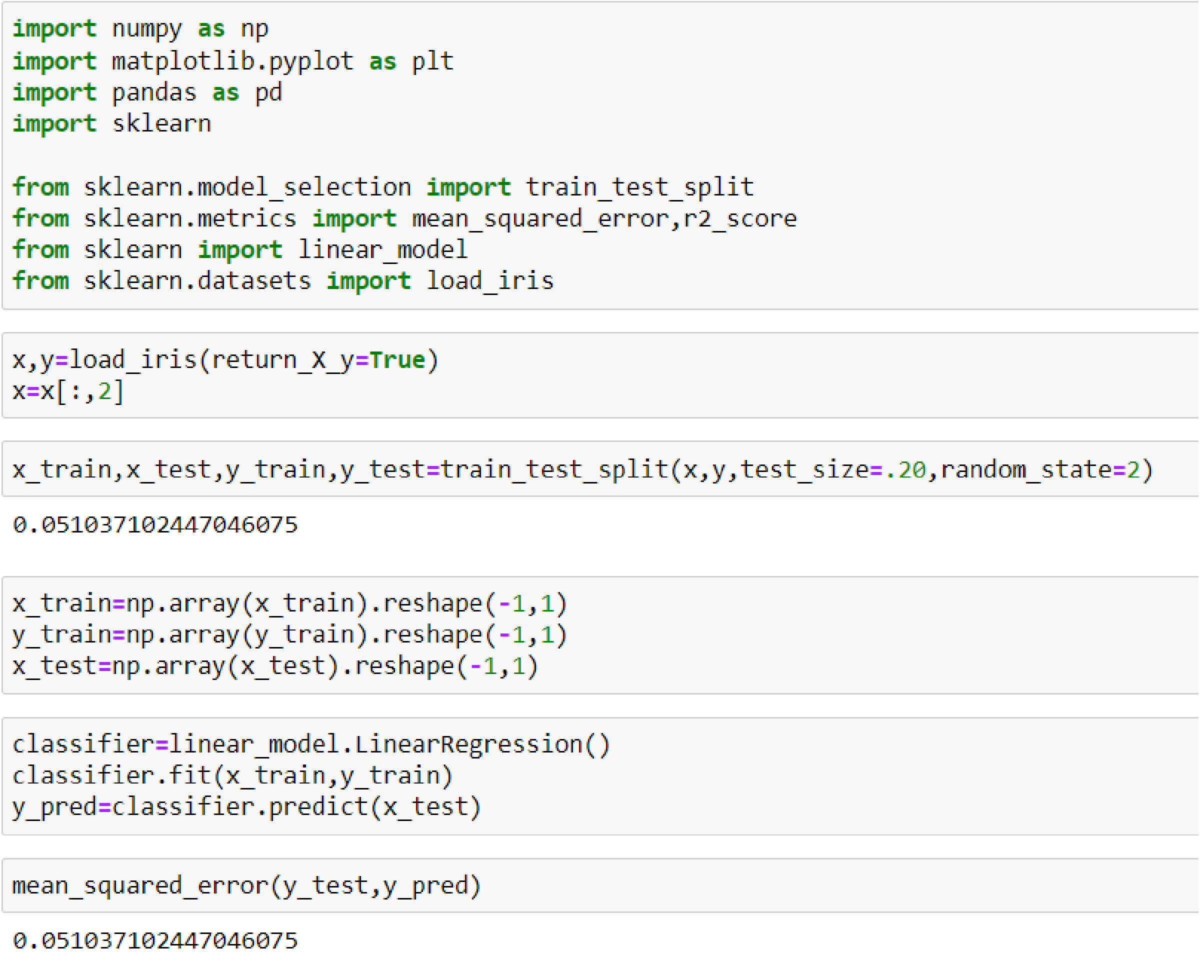
Experiment No 4:

Aim:

Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Source Code:

Linear Regression:



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
from sklearn import linear_model
from sklearn.datasets import load_iris
```

```
x,y=load_iris(return_X_y=True)
x=x[:,2]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=2)
```

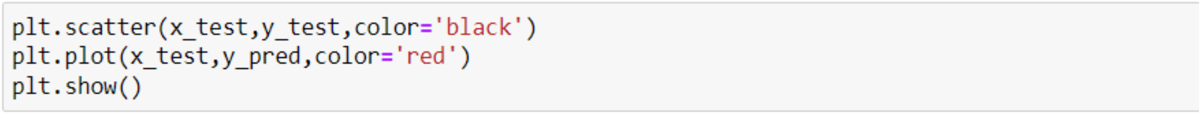
```
0.051037102447046075
```

```
x_train=np.array(x_train).reshape(-1,1)
y_train=np.array(y_train).reshape(-1,1)
x_test=np.array(x_test).reshape(-1,1)
```

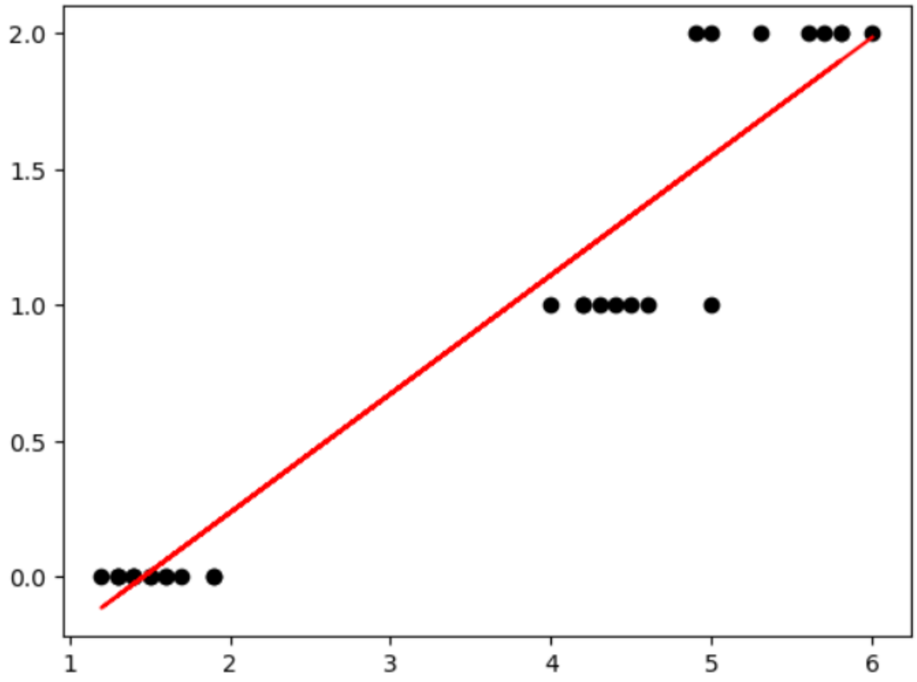
```
classifier=linear_model.LinearRegression()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
```

```
mean_squared_error(y_test,y_pred)
```

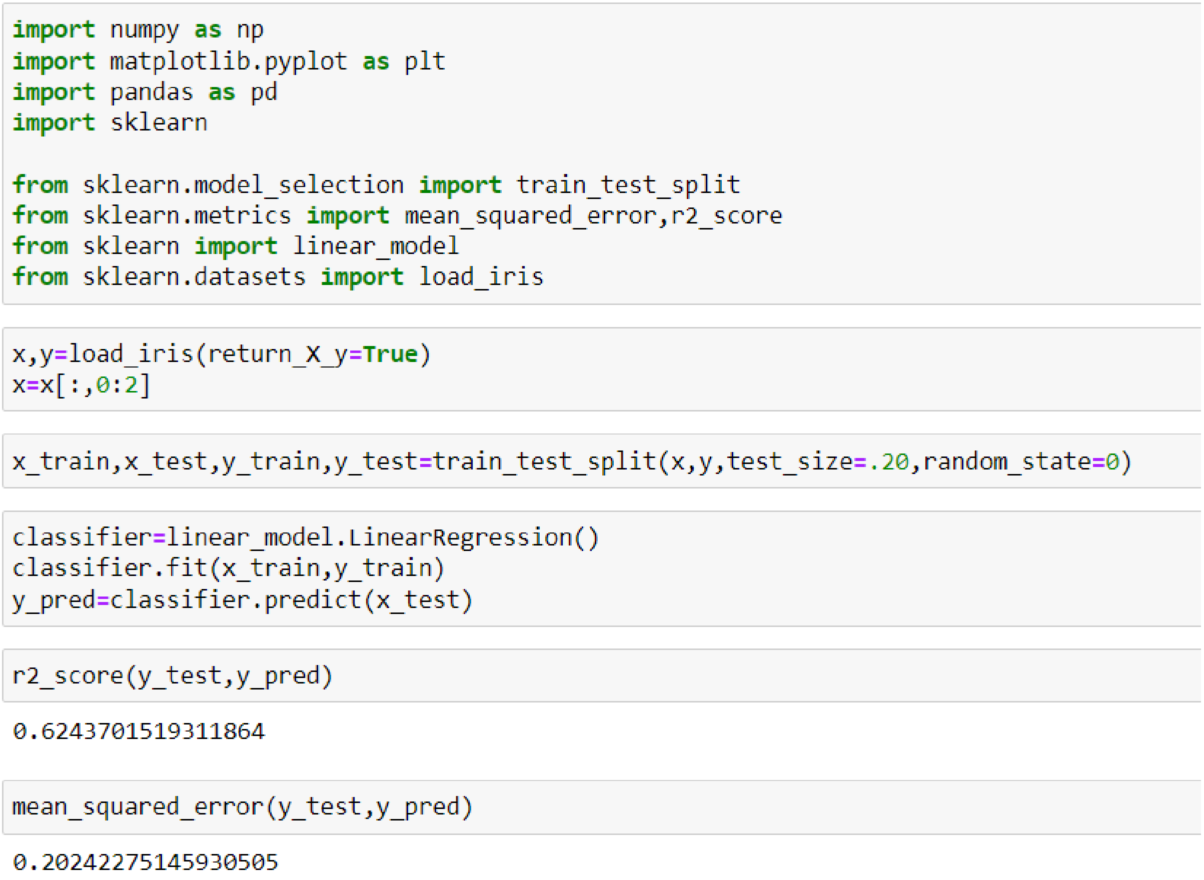
```
0.051037102447046075
```



```
plt.scatter(x_test,y_test,color='black')
plt.plot(x_test,y_pred,color='red')
plt.show()
```



Multiple Regression:



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
from sklearn import linear_model
from sklearn.datasets import load_iris
```

```
x,y=load_iris(return_X_y=True)
x=x[:,0:2]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=0)
```

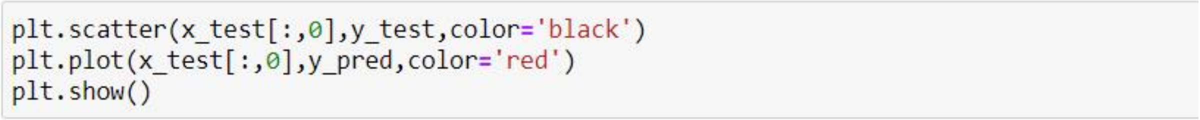
```
classifier=linear_model.LinearRegression()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
```

```
r2_score(y_test,y_pred)
```

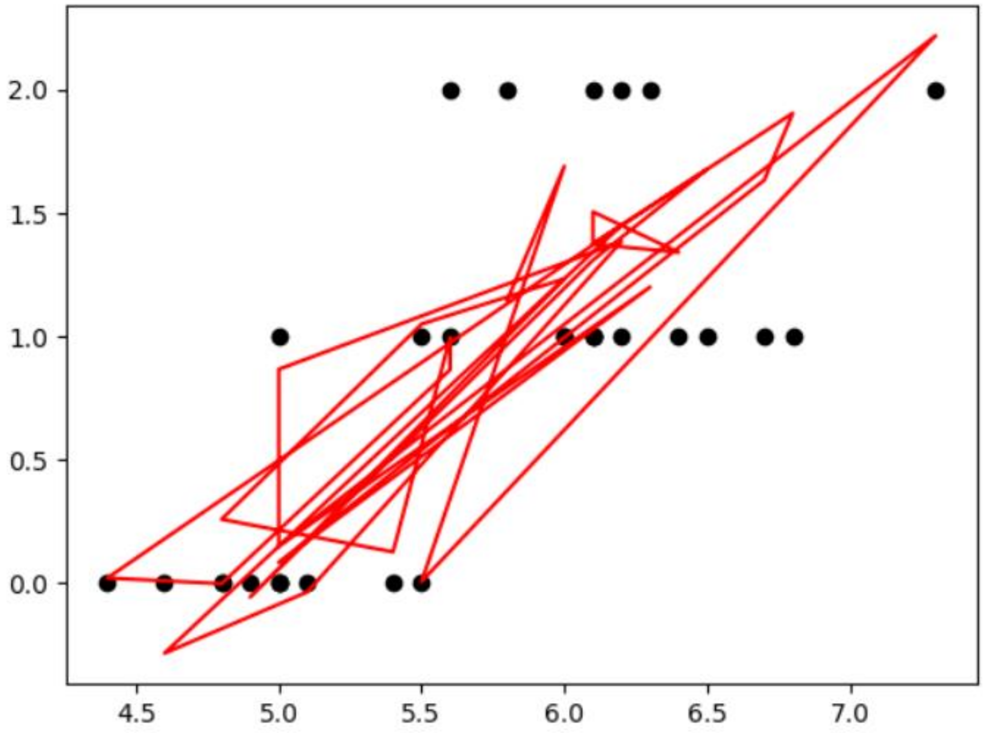
```
0.6243701519311864
```

```
mean_squared_error(y_test,y_pred)
```

```
0.20242275145930505
```



```
plt.scatter(x_test[:,0],y_test,color='black')
plt.plot(x_test[:,0],y_pred,color='red')
plt.show()
```



Result:

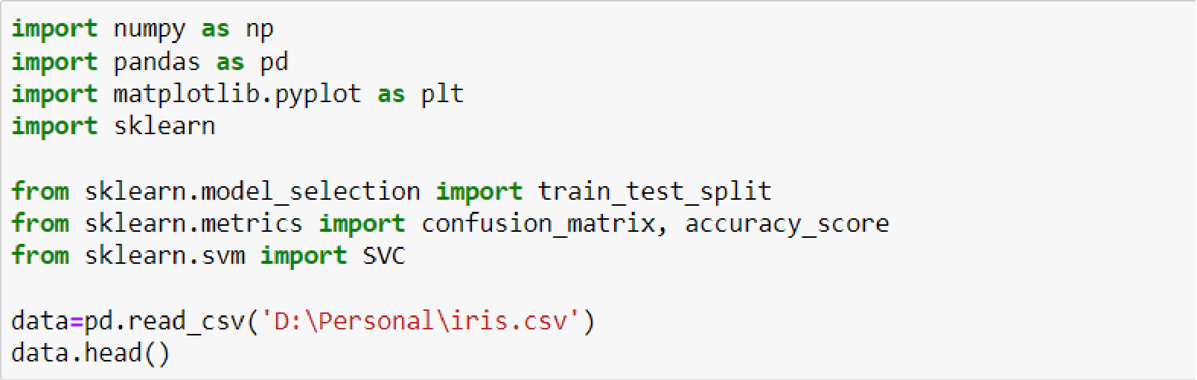
Successfully implemented linear and multiple regression techniques using **iris** dataset

Experiment No 5:

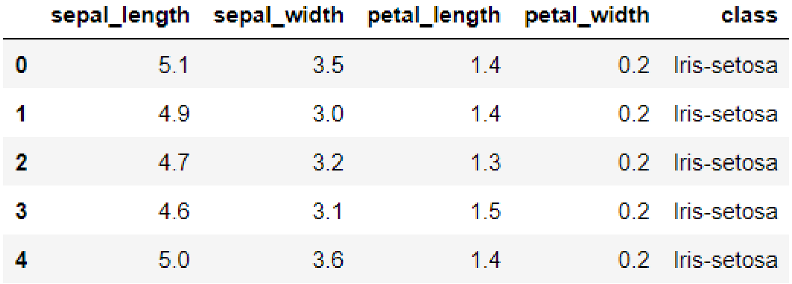
Aim:

Program to implement text classification using Support vector machine.

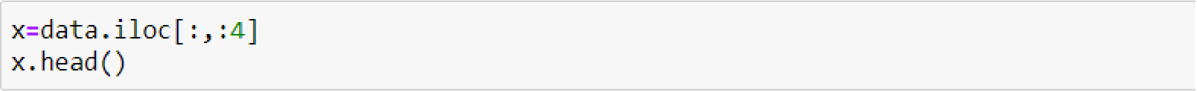
Source Code:



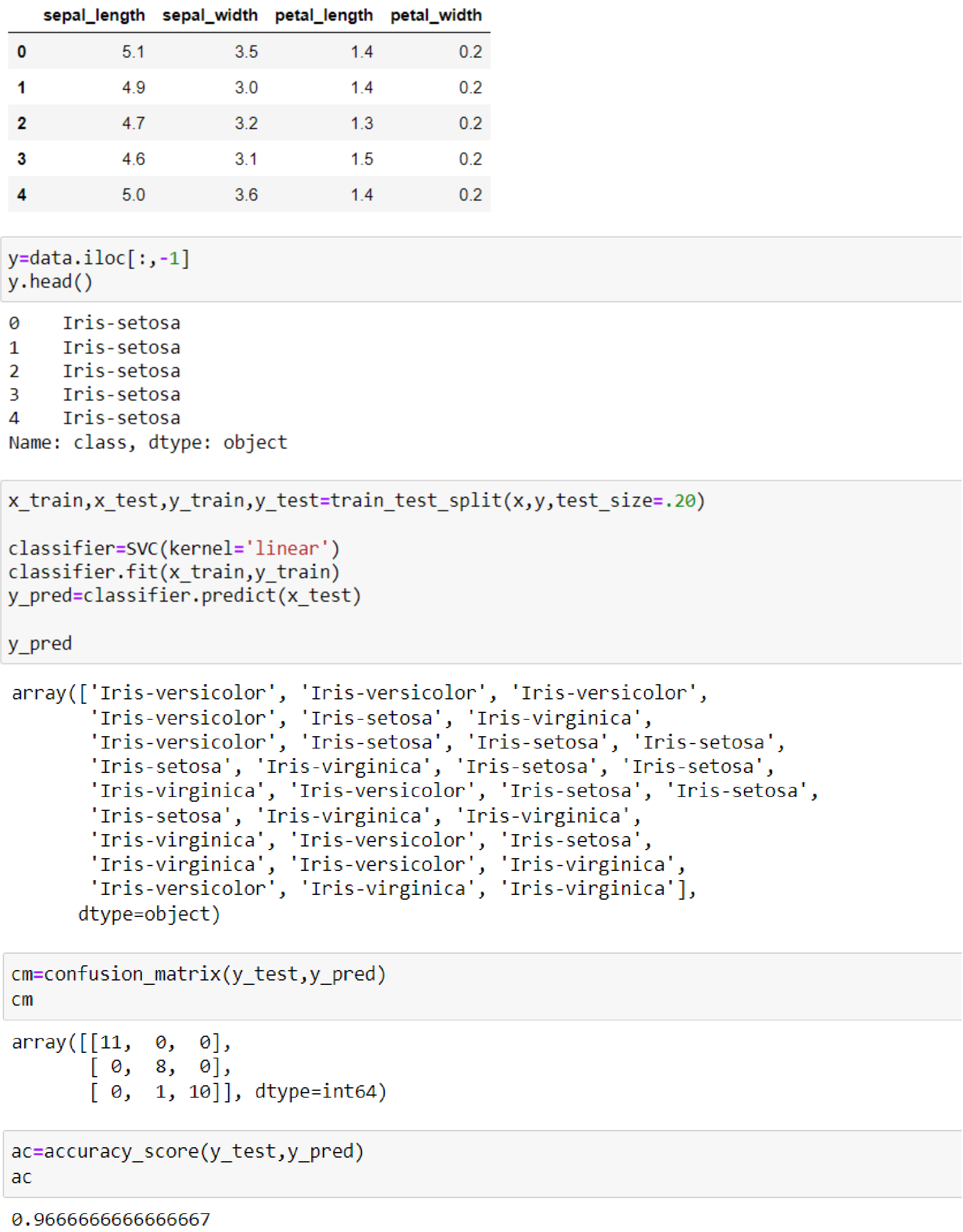
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.svm import SVC

data=pd.read_csv('D:\Personal\iris.csv')
data.head()
```



	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa



```
x=data.iloc[:,:4]
x.head()
```



	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
y=data.iloc[:,-1]
y.head()
```

```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: class, dtype: object
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)

classifier=SVC(kernel='linear')
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)

y_pred
```

```
array(['Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica'],
      dtype=object)
```

```
cm=confusion_matrix(y_test,y_pred)
cm
```

```
array([[11,  0,  0],
       [ 0,  8,  0],
       [ 0,  1, 10]], dtype=int64)
```

```
ac=accuracy_score(y_test,y_pred)
ac
```

```
0.9666666666666667
```

Result:

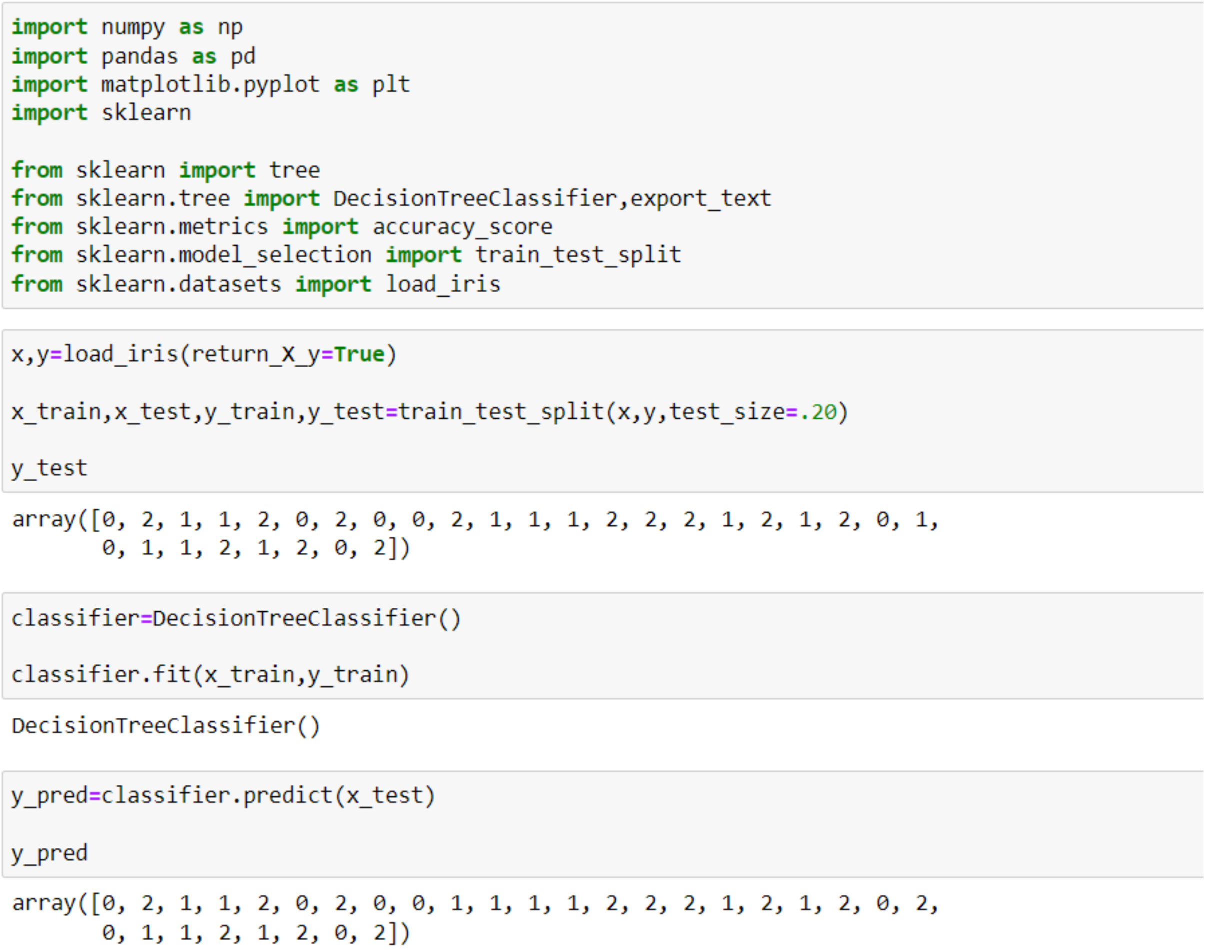
Successfully implemented text classification using Support vector machine using **iris** dataset and the accuracy of the algorithm is **0.9666666666666667**

Experiment No 6:

Aim:

Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

Source Code:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier,export_text
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
```

```
x,y=load_iris(return_X_y=True)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)

y_test
```

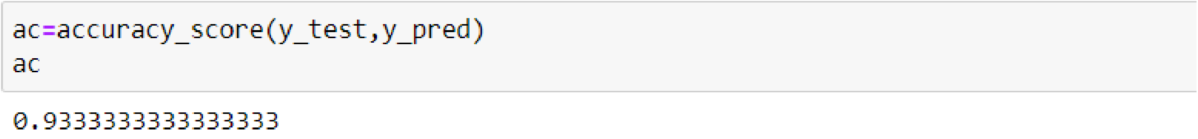
```
array([0, 2, 1, 1, 2, 0, 2, 0, 0, 2, 1, 1, 1, 2, 2, 2, 1, 2, 1, 2, 0, 1,
       0, 1, 1, 2, 1, 2, 0, 2])
```

```
classifier=DecisionTreeClassifier()

classifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

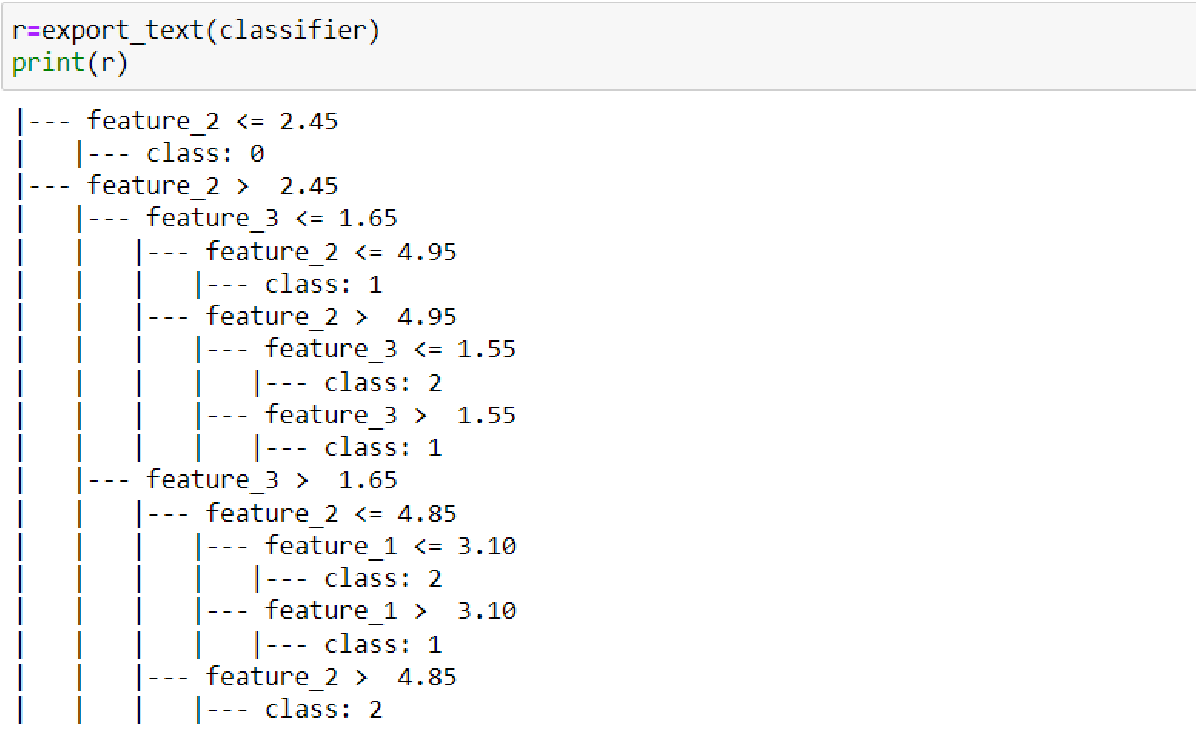
```
y_pred=classifier.predict(x_test)

y_pred
```

```
array([0, 2, 1, 1, 2, 0, 2, 0, 0, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 2, 0, 2,
       0, 1, 1, 2, 1, 2, 0, 2])
```



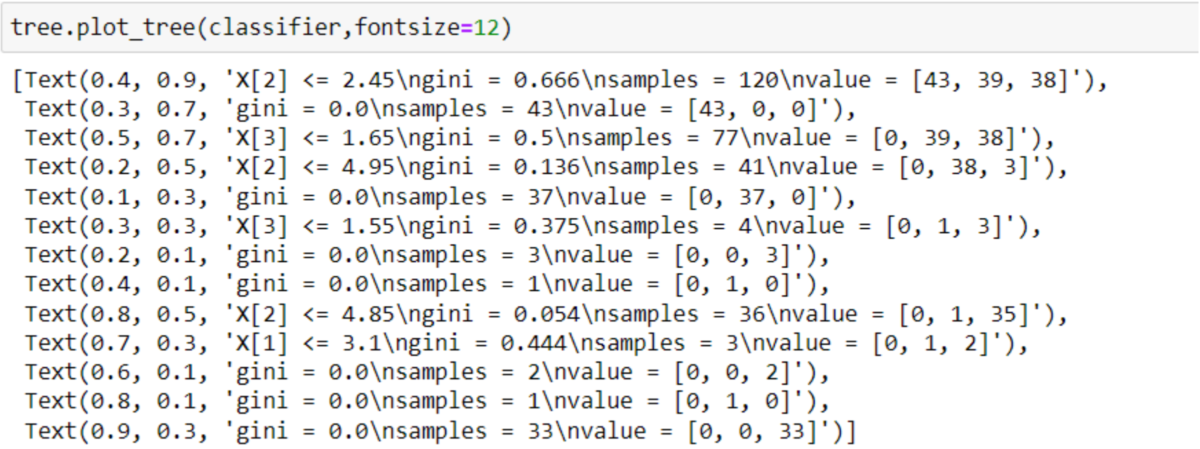
```
ac=accuracy_score(y_test,y_pred)
ac
```

```
0.9333333333333333
```



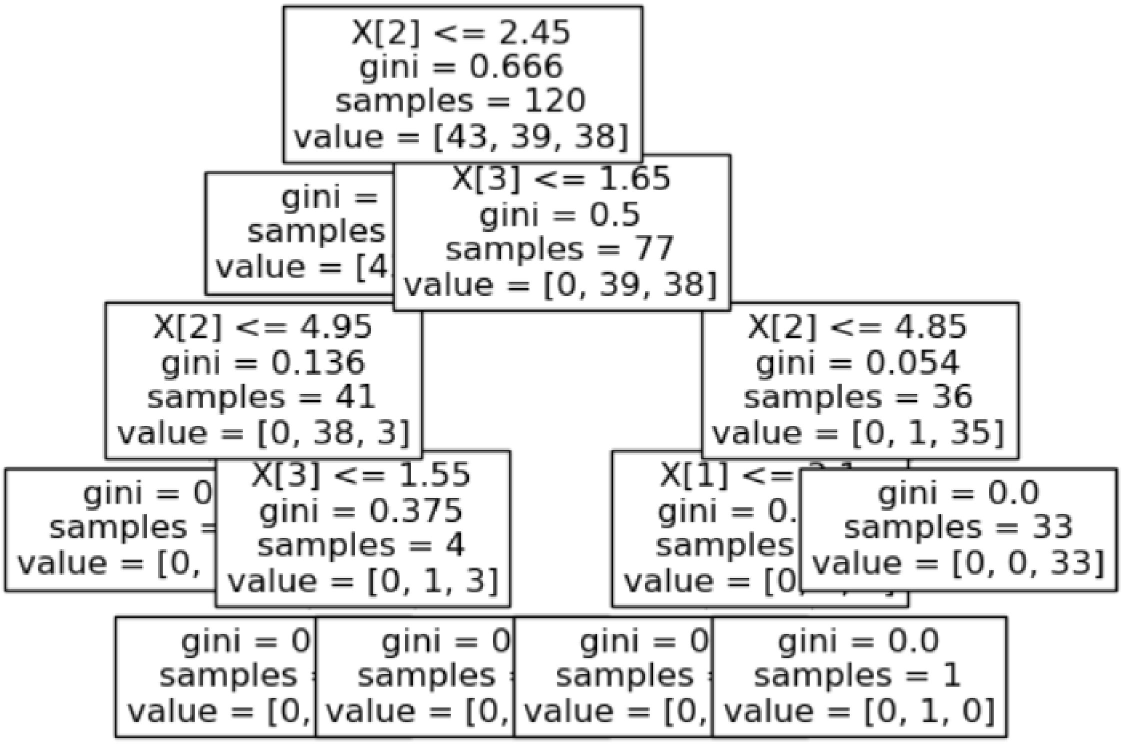
```
r=export_text(classifier)
print(r)
```

```
|--- feature_2 <= 2.45
|   |--- class: 0
|--- feature_2 >  2.45
|   |--- feature_3 <= 1.65
|   |   |--- feature_2 <= 4.95
|   |   |   |--- class: 1
|   |   |--- feature_2 >  4.95
|   |   |   |--- feature_3 <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- feature_3 >  1.55
|   |   |   |   |--- class: 1
|   |--- feature_3 >  1.65
|   |   |--- feature_2 <= 4.85
|   |   |   |--- feature_1 <= 3.10
|   |   |   |   |--- class: 2
|   |   |   |--- feature_1 >  3.10
|   |   |   |   |--- class: 1
|   |   |--- feature_2 >  4.85
|   |   |   |--- class: 2
```



```
tree.plot_tree(classifier,fontsize=12)
```

```
[Text(0.4, 0.9, 'X[2] <= 2.45\ngini = 0.666\nsamples = 120\nvalue = [43, 39, 38]'),
 Text(0.3, 0.7, 'gini = 0.0\nsamples = 43\nvalue = [43, 0, 0]'),
 Text(0.5, 0.7, 'X[3] <= 1.65\ngini = 0.5\nsamples = 77\nvalue = [0, 39, 38]'),
 Text(0.2, 0.5, 'X[2] <= 4.95\ngini = 0.136\nsamples = 41\nvalue = [0, 38, 3]'),
 Text(0.1, 0.3, 'gini = 0.0\nsamples = 37\nvalue = [0, 37, 0]'),
 Text(0.3, 0.3, 'X[3] <= 1.55\ngini = 0.375\nsamples = 4\nvalue = [0, 1, 3]'),
 Text(0.2, 0.1, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(0.4, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8, 0.5, 'X[2] <= 4.85\ngini = 0.054\nsamples = 36\nvalue = [0, 1, 35]'),
 Text(0.7, 0.3, 'X[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.6, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.8, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.9, 0.3, 'gini = 0.0\nsamples = 33\nvalue = [0, 0, 33]')]
```



Result:

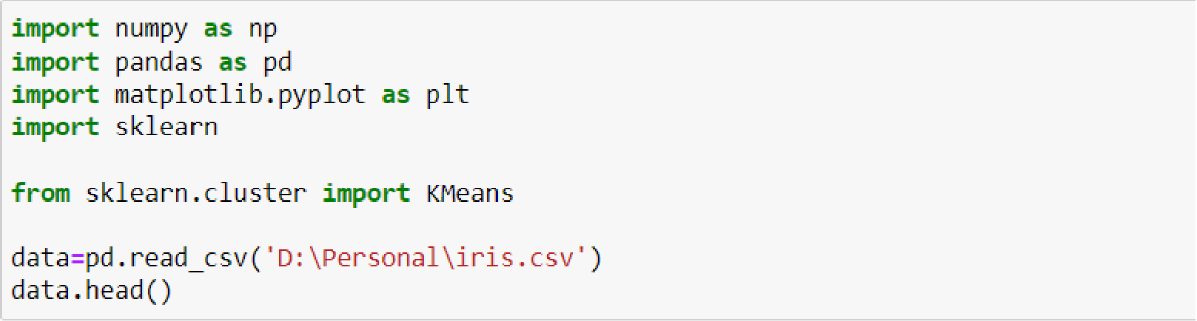
Successfully implemented decision trees using **iris** dataset and the accuracy of the algorithm is **0.9333333333333333**

Experiment No 7:

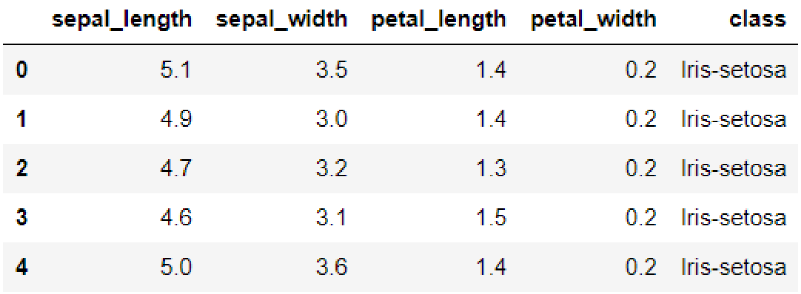
Aim:

Program to implement k-means clustering technique using any standard dataset available in the public domain

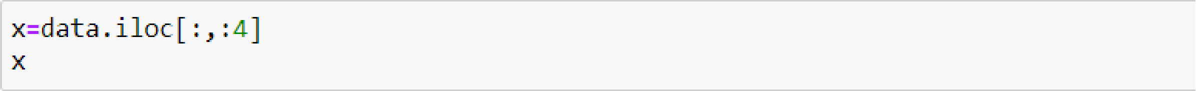
Source Code:



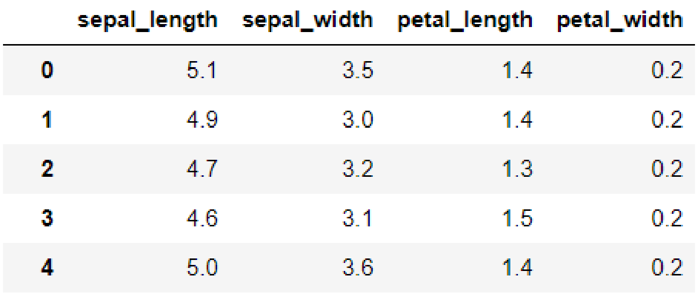
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn

from sklearn.cluster import KMeans

data=pd.read_csv('D:\Personal\iris.csv')
data.head()
```



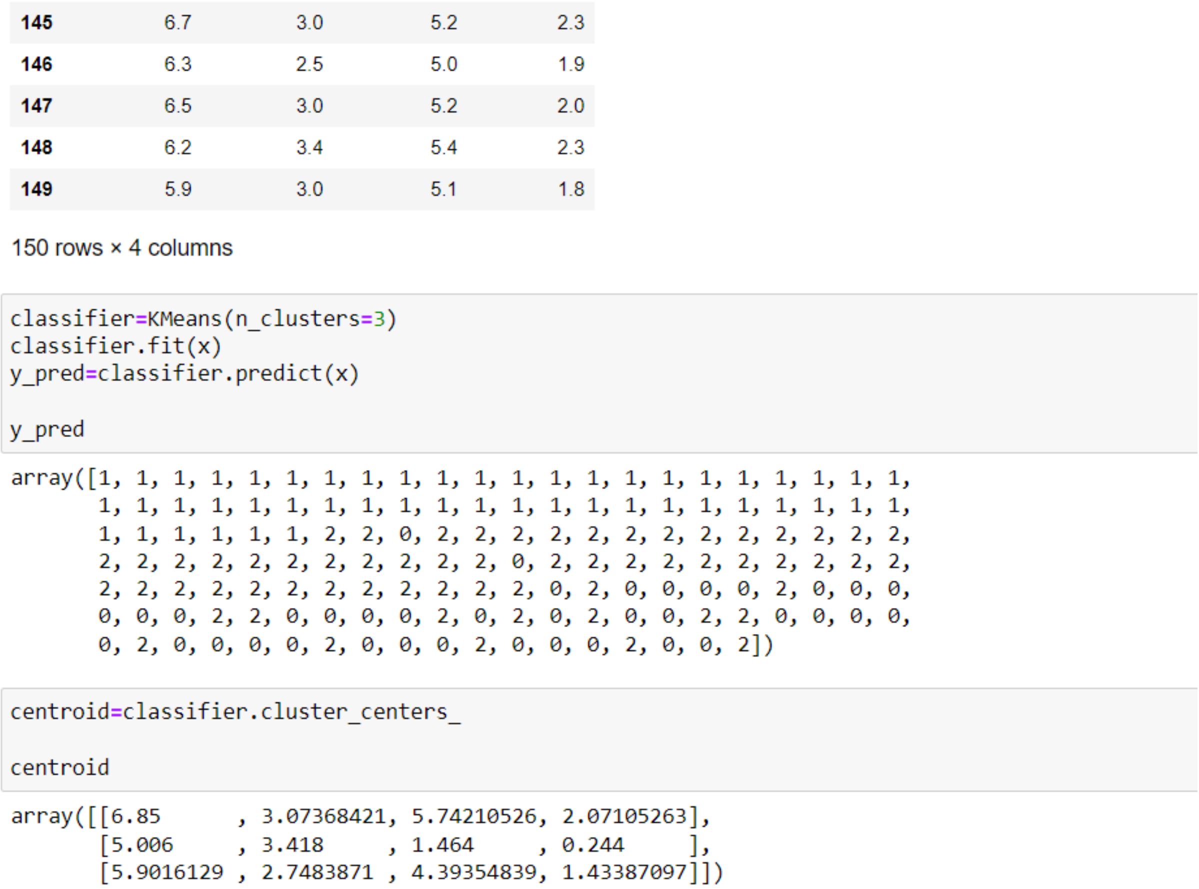
	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa



```
x=data.iloc[:,:4]
x
```



	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2



145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
classifier=KMeans(n_clusters=3)
classifier.fit(x)
y_pred=classifier.predict(x)

y_pred
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0,
       0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0, 0, 0,
       0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2])
```

```
centroid=classifier.cluster_centers_

centroid
```

```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.418     , 1.464     , 0.244     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

Result:

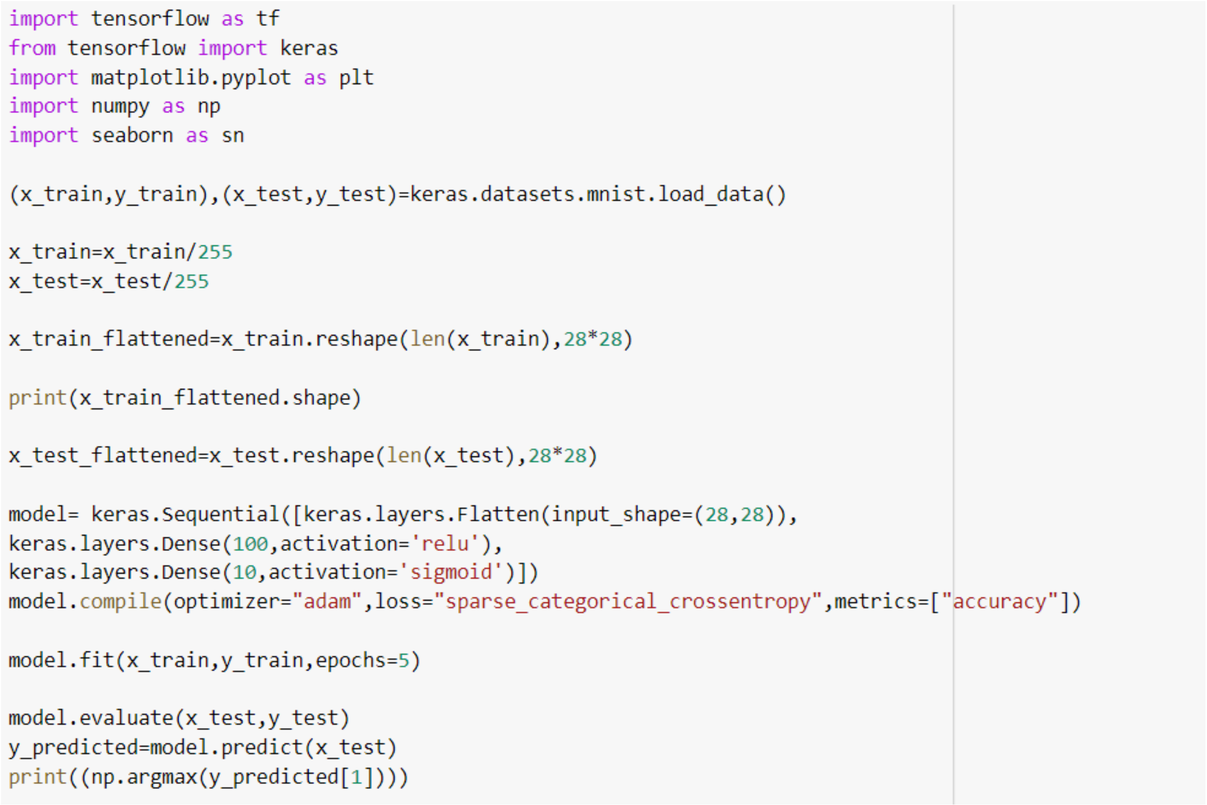
Successfully implemented k-means clustering technique using **iris** dataset.

Experiment No 8:

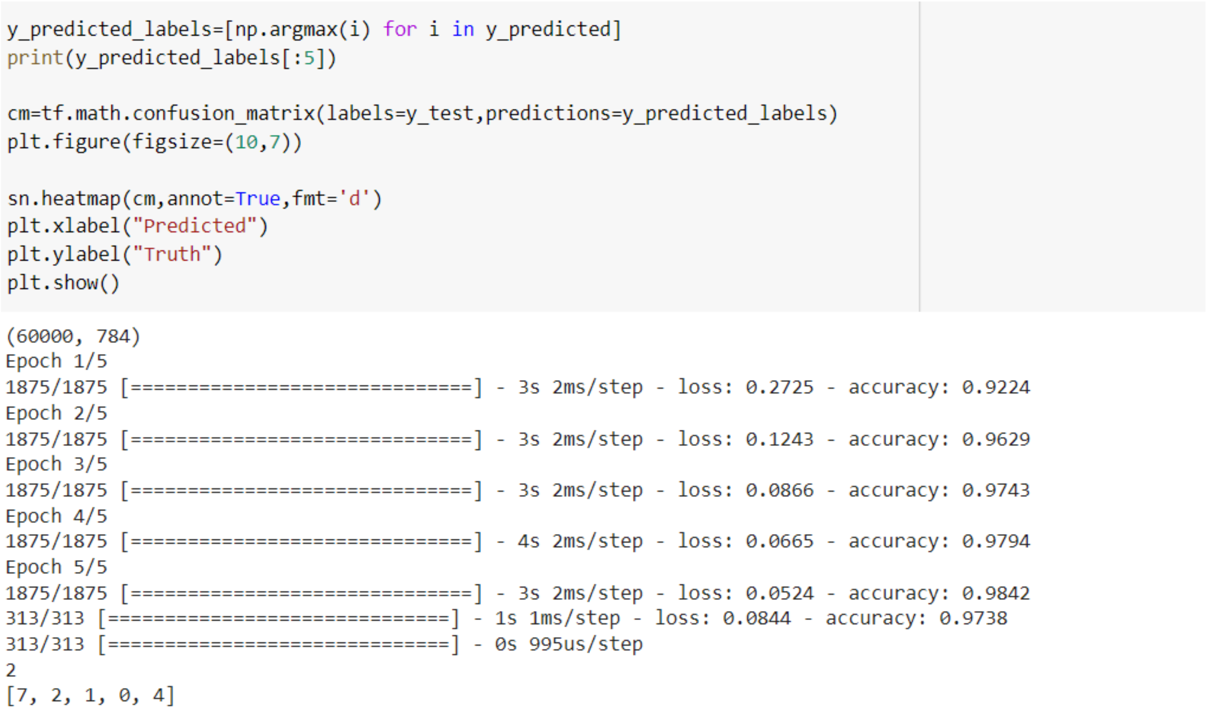
Aim:

Program on convolutional neural network to classify images from any standard dataset in the public domain using Keras framework.

Source Code:

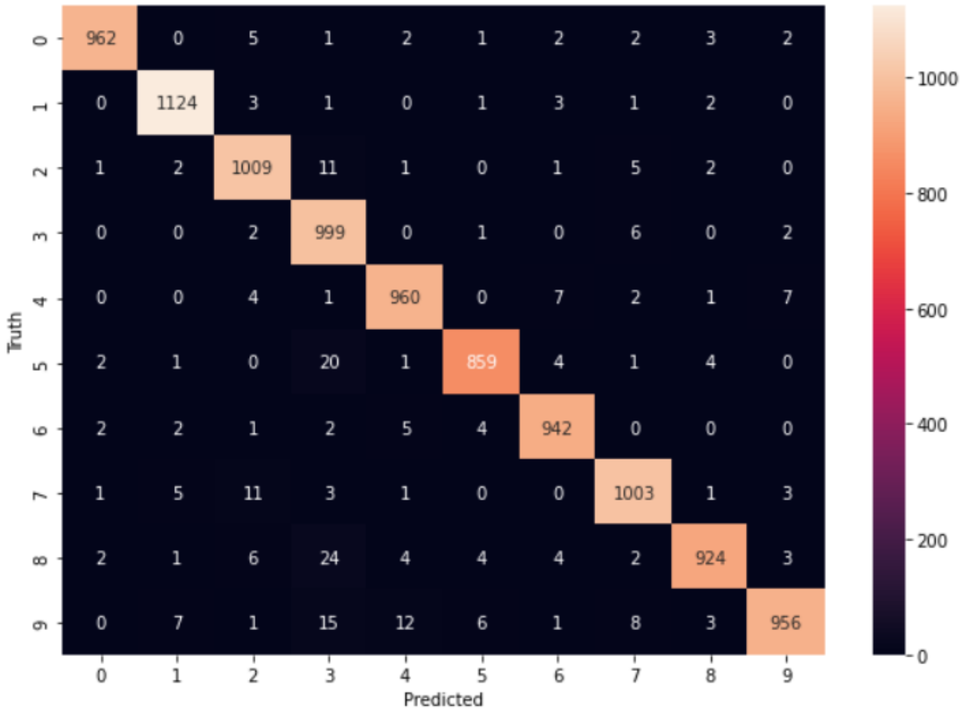


```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sn

(x_train,y_train),(x_test,y_test)=keras.datasets.mnist.load_data()

x_train=x_train/255
x_test=x_test/255

x_train_flattened=x_train.reshape(len(x_train),28*28)

print(x_train_flattened.shape)

x_test_flattened=x_test.reshape(len(x_test),28*28)

model= keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
keras.layers.Dense(100,activation='relu'),
keras.layers.Dense(10,activation='sigmoid')])
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=["accuracy"])

model.fit(x_train,y_train,epochs=5)

model.evaluate(x_test,y_test)
y_predicted=model.predict(x_test)
print((np.argmax(y_predicted[1])))
```



```
y_predicted_labels=[np.argmax(i) for i in y_predicted]
print(y_predicted_labels[:5])

cm=tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
plt.figure(figsize=(10,7))

sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Truth")
plt.show()
```

```
(60000, 784)
Epoch 1/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2725 - accuracy: 0.9224
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1243 - accuracy: 0.9629
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0866 - accuracy: 0.9743
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0665 - accuracy: 0.9794
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0524 - accuracy: 0.9842
313/313 [==============================] - 1s 1ms/step - loss: 0.0844 - accuracy: 0.9738
313/313 [==============================] - 0s 995us/step
2
[7, 2, 1, 0, 4]
```



Result:

Successfully implemented convolutional neural network to classify images from mnist dataset using Keras framework.