1 . Write a python program to find mean, mode , median

In [1]:
```
#1

import numpy as np
from scipy import stats

list1=[12,11,13,14,15,12,11,4]
x=np.mean(list1)

print("mean=",x)

x=np.median(list1)
print("midean=",x)

x=stats.mode(list1)

print(x)
```

```
mean= 11.5
midean= 12.0
ModeResult(mode=11, count=2)
```

In [2]:
```
#2

list1=[12,11,13,14,15,12,11,4]
mean=sum(list1)/len(list1)

print("mean=",mean)

list1.sort()

if len(list1)%2==0:
    m1=list1[len(list1)//2]
    m2=list1[len(list1)//2-1]
    median=(m1+m2)/2
else:
    median=list1[len(list1)//2]
print("median=",median)

freq={}
for i in list1:
    freq.setdefault(i,0)
    freq[i]+=1

fre=max(freq.values())

for i,j in freq.items():
    if j==fre:
        mode=i
print("Mode=",mode)
```

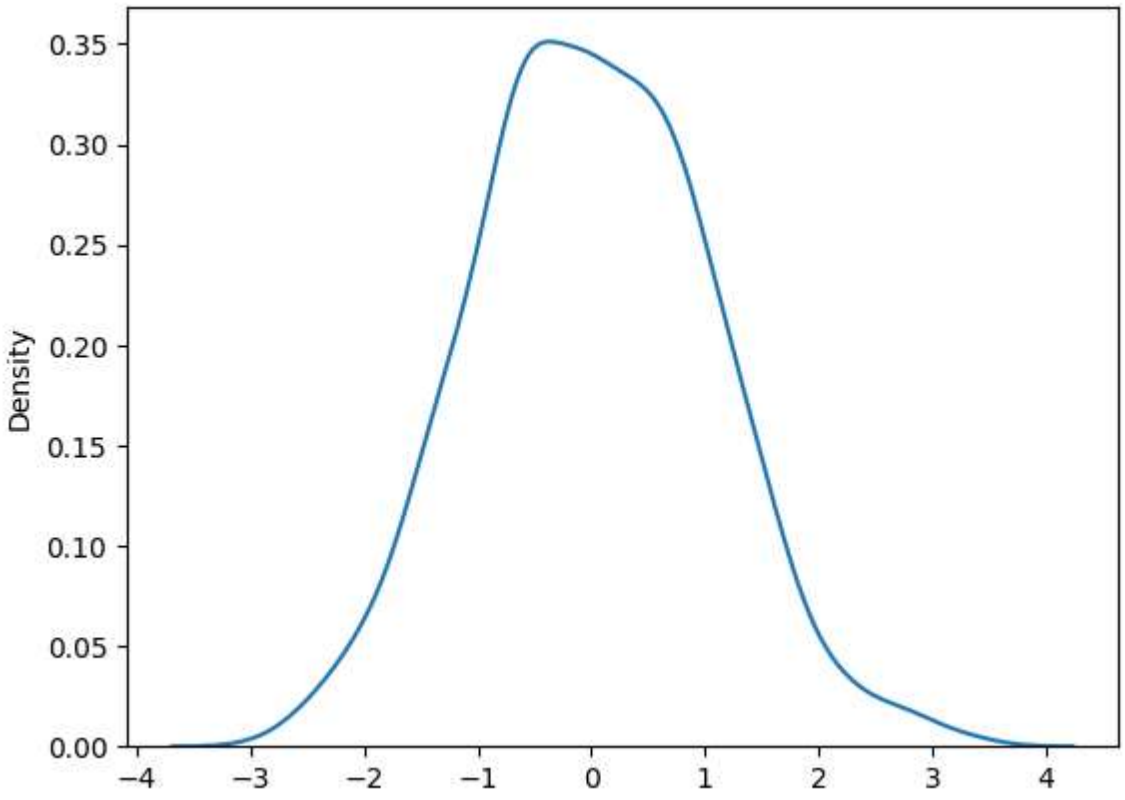
```
mean= 11.5
median= 12.0
Mode= 12
```

In [ ]:

2. Write a python program to typical normal data distribution .

In [3]:
```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.normal(size=1000),hist=False)
plt.show()
```

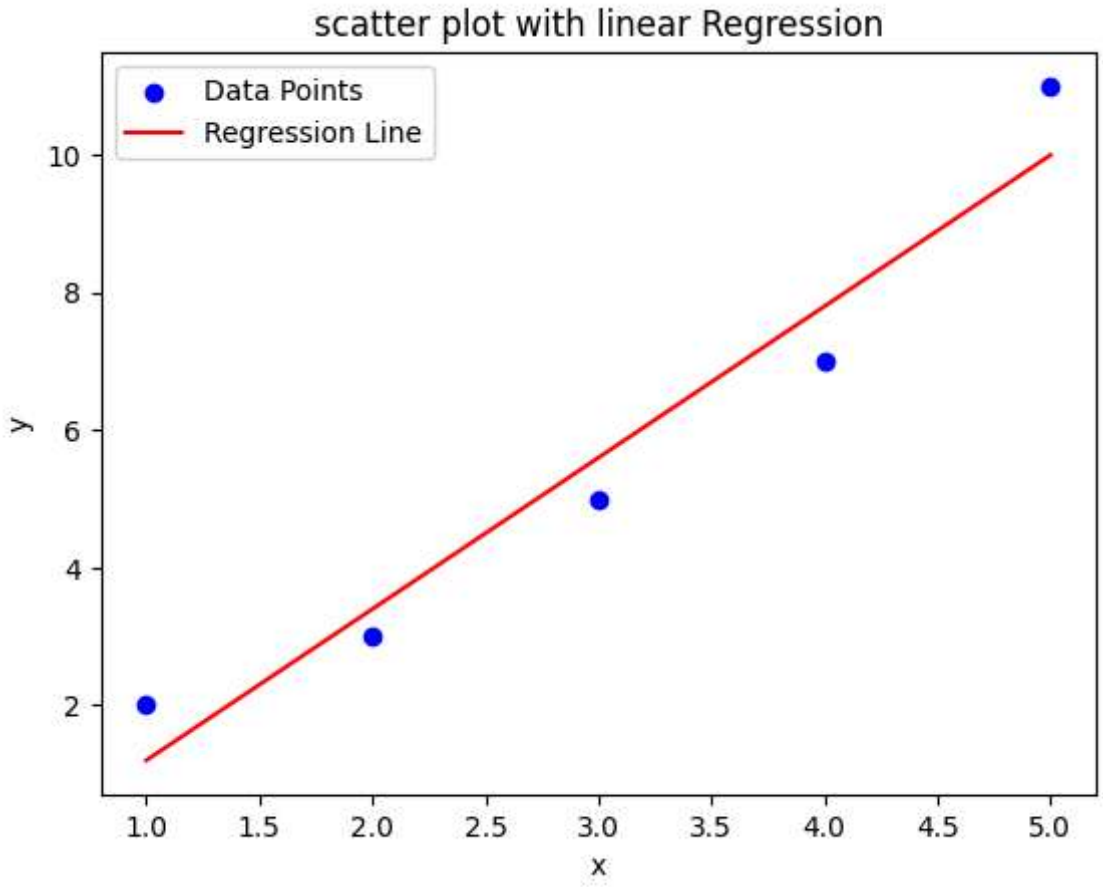
```
C:\Users\harshal\AppData\Local\Temp\ipykernel_10708\1366969501.py:5: UserW
arning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `kdeplot` (an axes-level function for kernel densi
ty plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(random.normal(size=1000),hist=False)
```



In [ ]:

3. Write a python program to draw scatter plot of linear

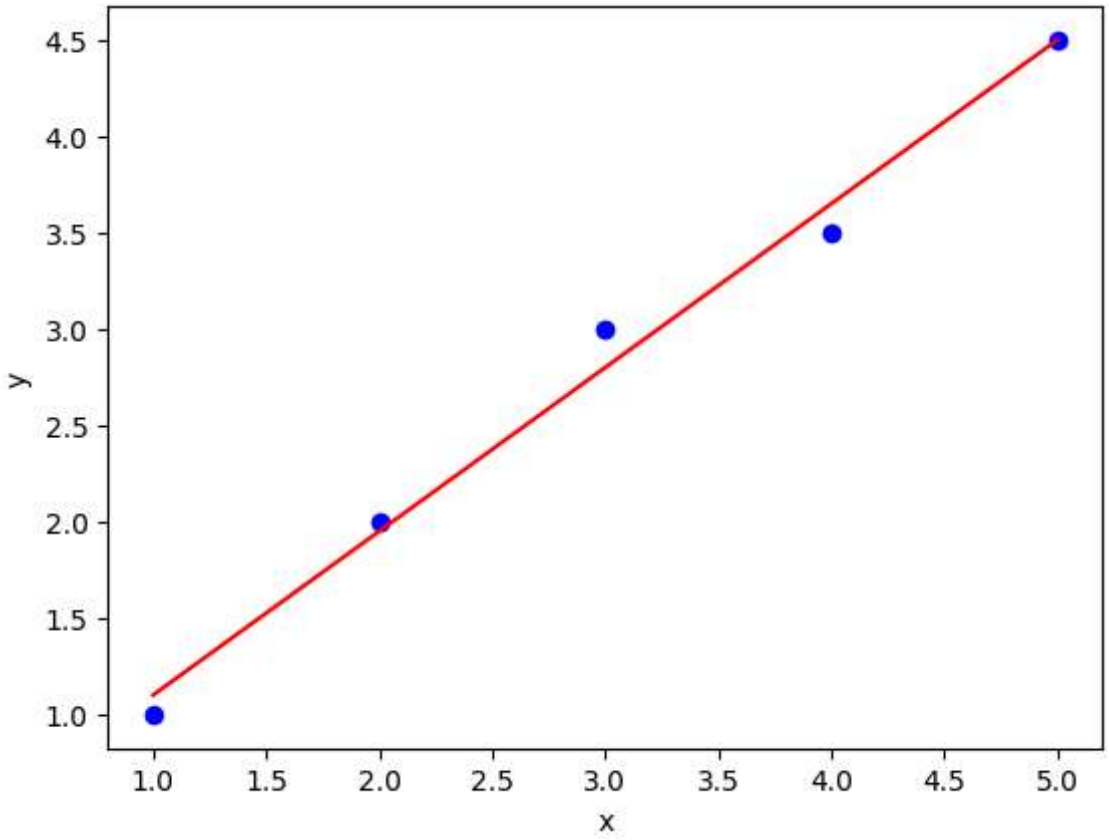
In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
x=np.array([[1],[2],[3],[4],[5]])
y=np.array([[2],[3],[5],[7],[11]])
model=LinearRegression()
model.fit(x,y)
y_pred=model.predict(x)
plt.scatter(x,y,color="b",label="Data Points")
plt.plot(x,y_pred,color='r',label="Regression Line")
plt.xlabel('x')
plt.ylabel('y')
plt.title("scatter plot with linear Regression")
plt.legend()
plt.show()
```



In [ ]:

4. Write a python program to draw the line of Linear Regression

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
x=np.array([[1],[2],[3],[4],[5]])
y=np.array([1,2,3,3.5,4.5])
model=LinearRegression()
model.fit(x,y)
y_pred=model.predict(x)
plt.scatter(x,y,color='b')
plt.plot(x,y_pred,color='r')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



In [ ]:

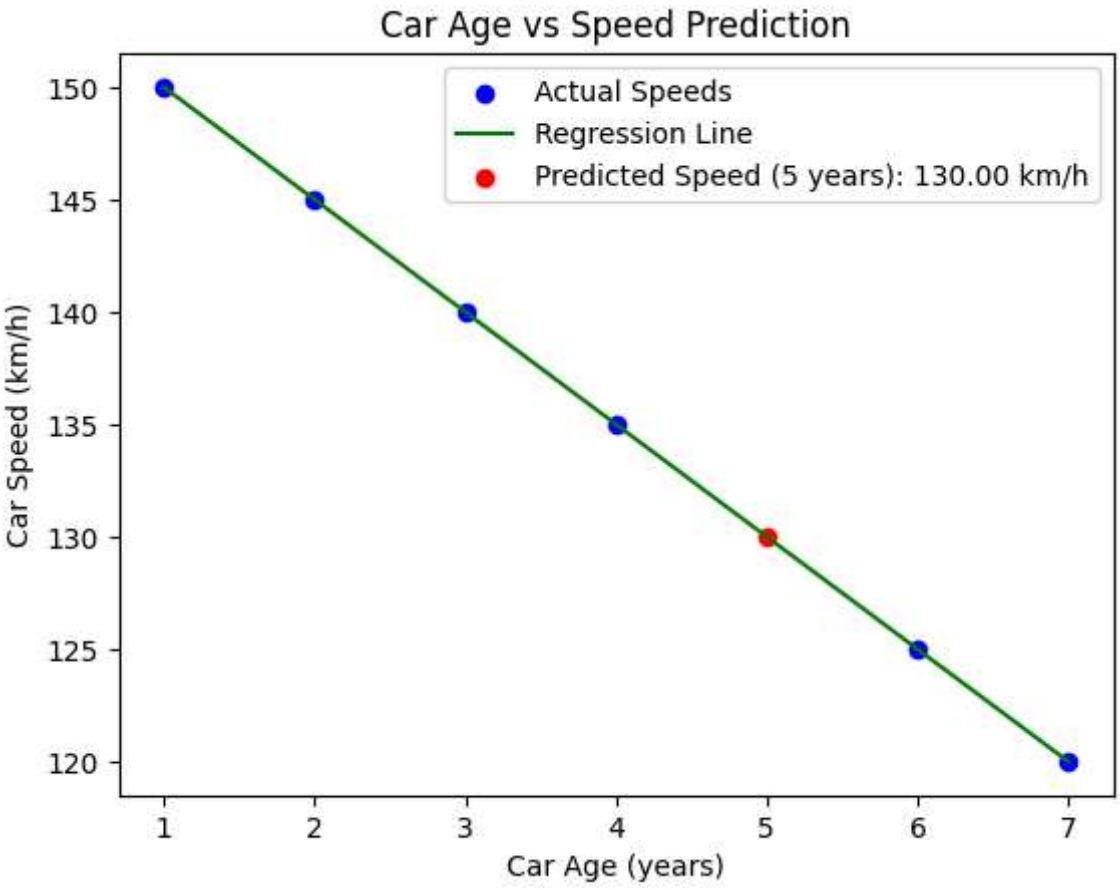
5. Write a python program to predict the speed of a 5 years old car.

In [6]:
```
#1

import numpy as np
from sklearn.linear_model import LinearRegression
car_age=np.array([[1],[2],[3],[4],[6],[7]])
car_speed=np.array([150,145,140,135,125,120])
model.fit(car_age,car_speed)
predicted_speed=model.predict([[5]])
print(f"The predict speed of a 5-year-old car is: {predicted_speed[0]:.2f} k
```

```
The predict speed of a 5-year-old car is: 130.00 km/h
```

In [1]:
```
#2

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Data: car age and corresponding car speed
car_age = np.array([[1], [2], [3], [4], [6], [7]])
car_speed = np.array([150, 145, 140, 135, 125, 120])

# Create the LinearRegression model
model = LinearRegression()

# Train the model
model.fit(car_age, car_speed)

# Make a prediction for a 5-year-old car
predicted_speed = model.predict([[5]])
print(f"The predicted speed of a 5-year-old car is: {predicted_speed[0]:.2f}

# Plotting the data and the prediction
plt.scatter(car_age, car_speed, color='blue', label='Actual Speeds')
plt.plot(car_age, model.predict(car_age), color='green', label='Regression l

# Marking the predicted point
plt.scatter(5, predicted_speed, color='red', label=f'Predicted Speed (5 year

# Labels and Title
plt.xlabel('Car Age (years)')
plt.ylabel('Car Speed (km/h)')
plt.title('Car Age vs Speed Prediction')
plt.legend()

# Show the plot
plt.show()
```

```
The predicted speed of a 5-year-old car is: 130.00 km/h
```



In [ ]:

6. Write a python prgoram to print the coefficient values of the regression object

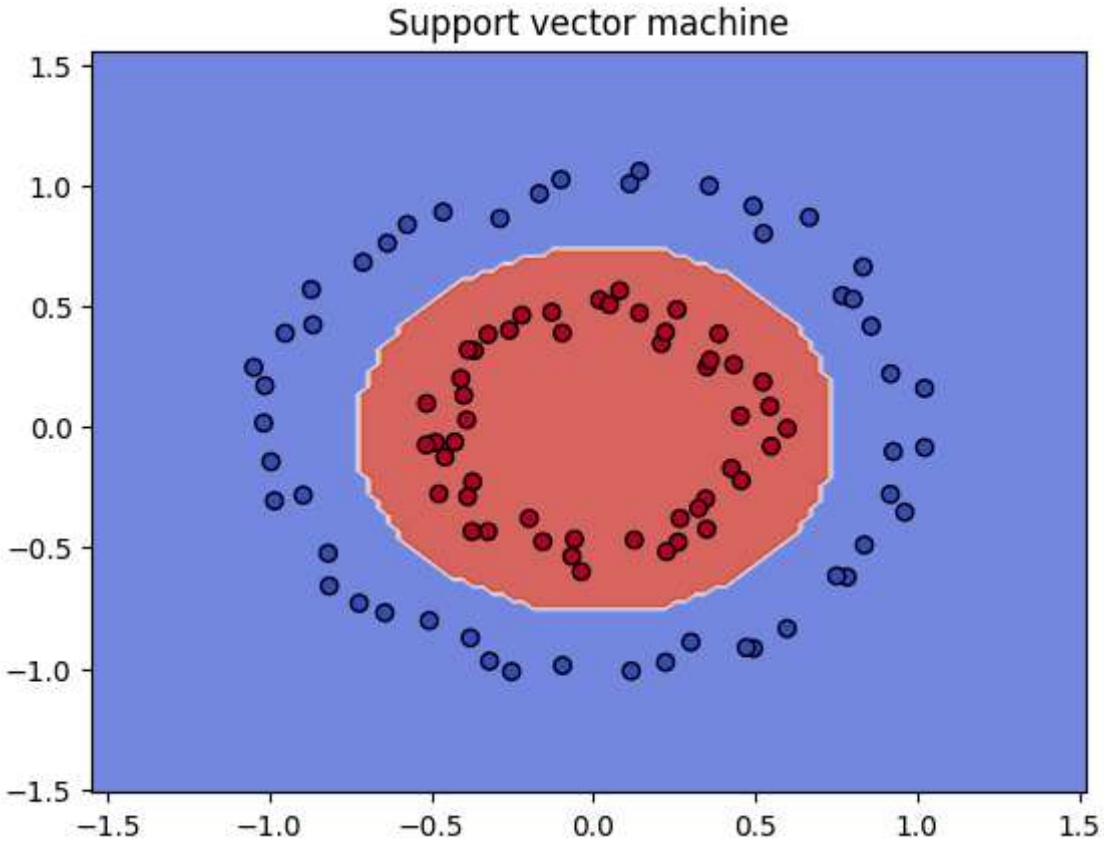
In [7]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
car_age = np.array([[1], [2], [3], [4], [6], [7]])
car_speed = np.array([150, 145, 140, 135, 125, 120])
model = LinearRegression()
model.fit(car_age, car_speed)
print(f"Coefficient (Slope): {model.coef_[0]:.2f}")
print(f"Intercept: {model.intercept_:.2f}")
```

```
Coefficient (Slope): -5.00
Intercept: 155.00
```

In [ ]:

7. Write a python program to 2d binary classification data generated by make_circles() have a spherical decision boundary

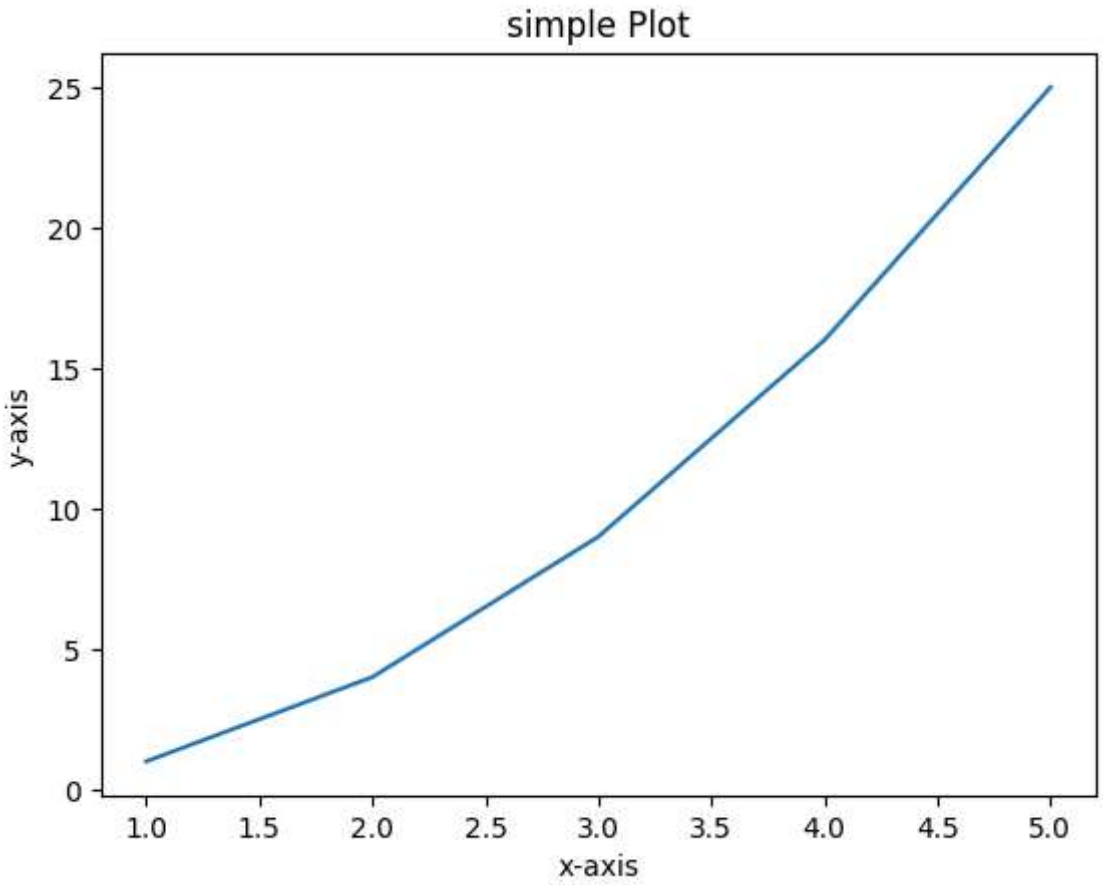
In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.svm import SVC
X, y = make_circles(n_samples=100, noise=0.05, factor=0.5)
model = SVC(kernel='rbf', gamma=1.0)
model.fit(X, y)
xx, yy = np.meshgrid(np.linspace(X[:, 0].min()-0.5, X[:, 0].max()+0.5, 100),
                     np.linspace(X[:, 1].min()-0.5, X[:, 1].max()+0.5, 100))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.title("Support vector machine")
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap=plt.cm.coolwarm)
plt.show()
```



In [ ]:

8. Write a python program to display the plot we can use the functions plot() and show() from pyplot

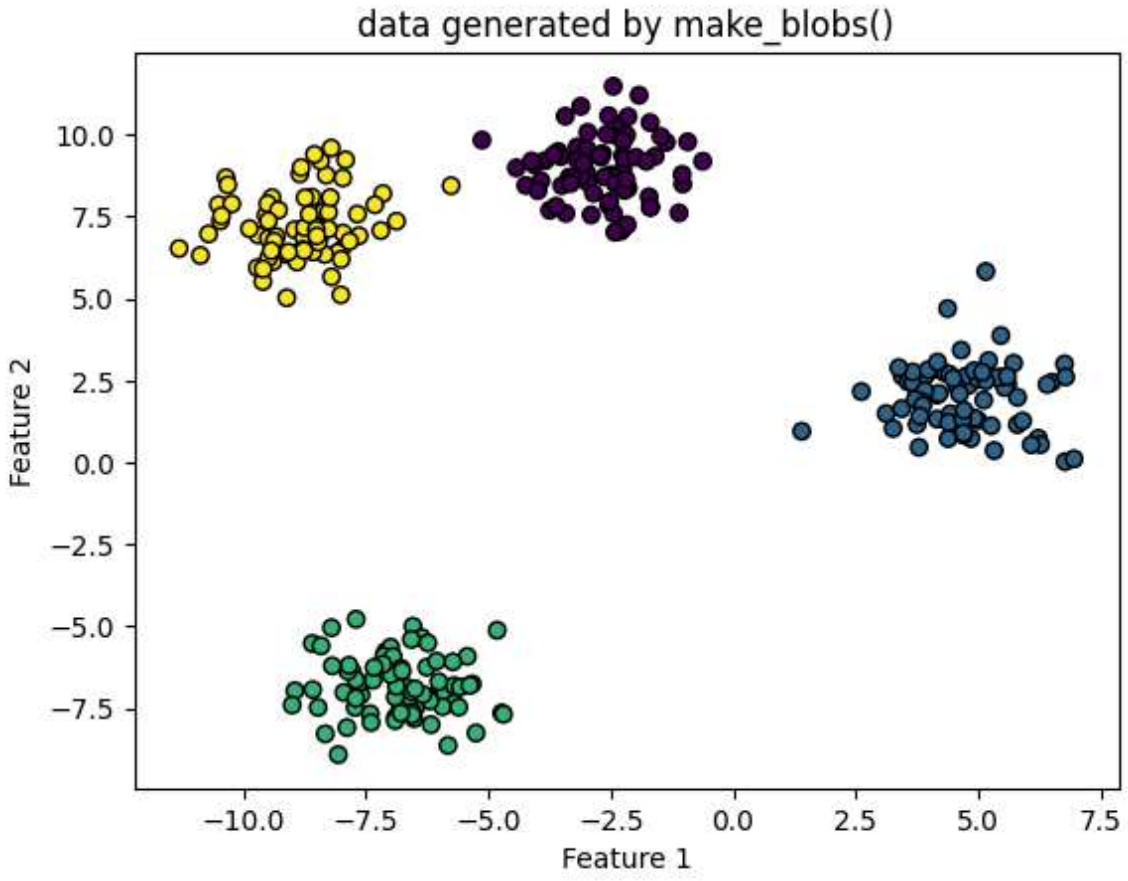
In [9]:
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5]
y=[1,4,9,16,25]
plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('simple Plot')
plt.show()
```



In [ ]:

9. write a python program to data generated by the function make_blobs() are blobs that can be utilized for clustering

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
x,y=make_blobs(n_samples=300,centers=4,random_state=42,cluster_std=1.0)
plt.scatter(x[:,0],x[:,1],c=y,cmap='viridis',edgecolor='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('data generated by make_blobs()')
plt.show()
```

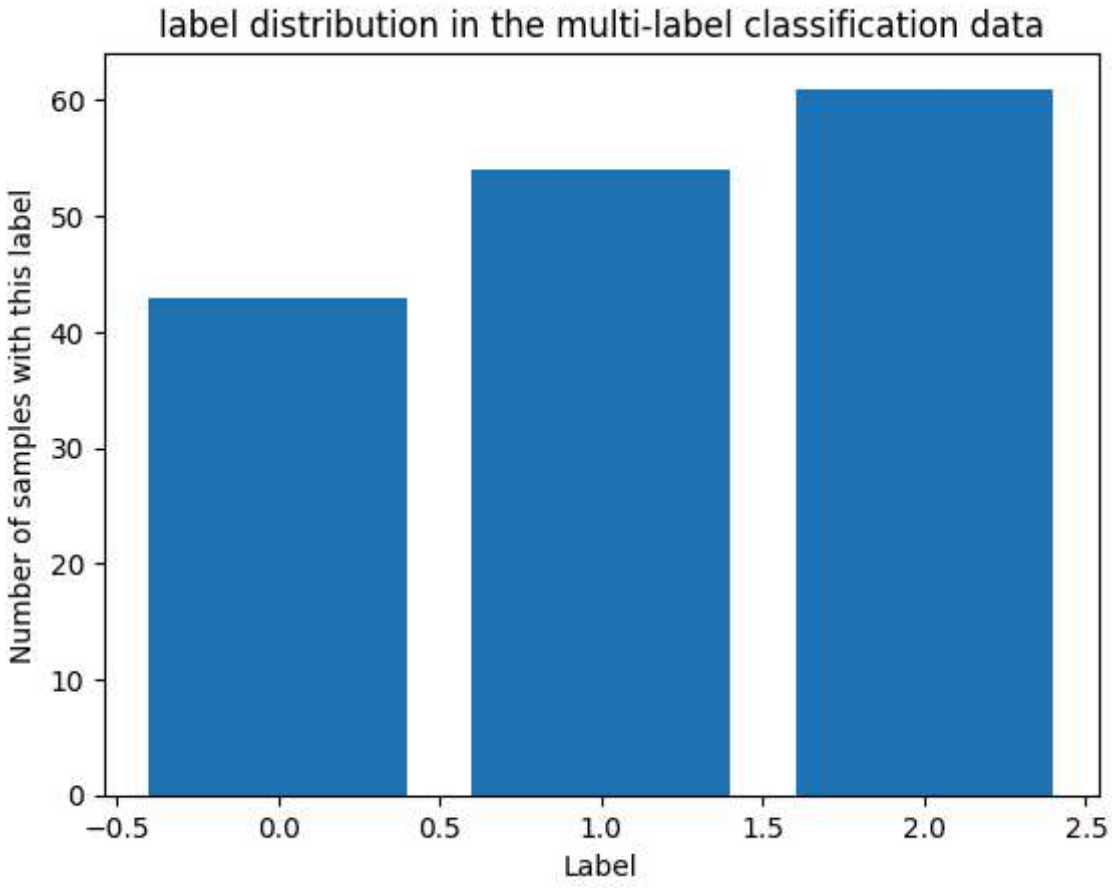


In [ ]:

10.Write a python program to random multi-label classification data is created by the function make make_multilabel_classification()

In [11]:
```
import numpy as np
from sklearn.datasets import make_multilabel_classification
import matplotlib.pyplot as plt

x,y=make_multilabel_classification(n_samples=100,n_features=5,n_classes=3,n_
print("feature matrix (x) : ")
print(x[:5])

print("\n multi-label target matrix (y)")
print(y[:5])

label_counts=np.sum(y,axis=0)
plt.bar(range(len(label_counts)),label_counts)
plt.title("label distribution in the multi-label classification data")
plt.xlabel("Label")
plt.ylabel("Number of samples with this label")
plt.show()
```

```
feature matrix (x) :
[[ 6. 22.  1.  3.  9.]
 [ 8.  4. 16. 10.  5.]
 [ 5. 27. 14.  4. 10.]
 [ 7. 19. 11. 10. 14.]
 [ 6. 18. 13.  6. 12.]]

 multi-label target matrix (y)
[[0 1 0]
 [1 0 1]
 [0 1 1]
 [1 1 1]
 [0 1 1]]
```

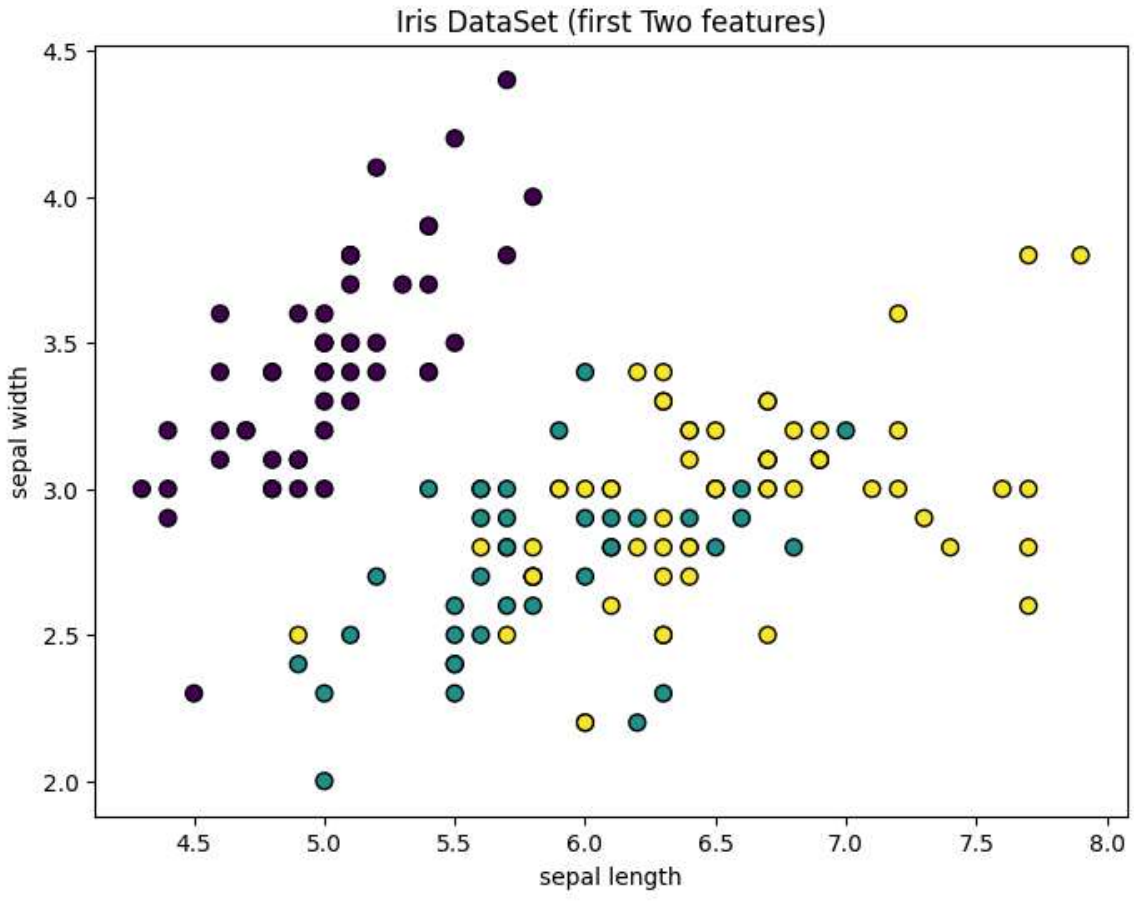

label distribution in the multi-label classification data
Number of samples with this label
Label
0
10
20
30
40
50
60
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

In [ ]:

11. write a python program to implement the knn algorithm

In [12]:
```
#1

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_sta
k=3
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
accuracy=accuracy_score(y_test,y_pred)
print(f"Accuracy of KNN classifier with k= {k}: {accuracy:.2f}")
plt.figure(figsize=(8,6))
plt.scatter(x[:,0],x[:,1],c=y,cmap='viridis',edgecolor='k',s=50)
plt.title("Iris DataSet (first Two features)")
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.show()
```

```
Accuracy of KNN classifier with k= 3: 1.00
```



In [13]:
```
#2

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

iris=load_iris()
x,y=iris.data,iris.target

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_stat
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
accuracy=accuracy_score(y_test,y_pred)
print(f"Accuracy of KNN classifier: {accuracy:.2f}")
```

```
Accuracy of KNN classifier: 1.00
```

12. write a python program to creating a dataframe to implement one hot encoding from CSV file.

In [14]:
```
import pandas as pd

data={
    "name":["harshal","Sanket","Rohit","Ritesh"],
    "marks":[77,99,69,89],
    "city":["Shirpur","surat","bhopal","nageshwar"],
    "packege":["12lk","9lk","13lk","9lk"]
}

df=pd.DataFrame(data)
df.to_csv('sample_data.csv',index=False)
df=pd.read_csv('sample_data.csv')
print("Original Dataframe")
print(df)
df_encoded=pd.get_dummies(df,drop_first=True)
print("\n One -Hot Encoded DataFrame: ")
print(df_encoded)
```

```
Original Dataframe
      name  marks       city packege
0  harshal     77    Shirpur    12lk
1   Sanket     99      surat     9lk
2    Rohit     69     bhopal    13lk
3   Ritesh     89  nageshwar     9lk

 One -Hot Encoded DataFrame:
   marks  name_Rohit  name_Sanket  name_harshal  city_bhopal  city_nageshw
ar  \
0     77       False        False          True        False           Fal
se
1     99       False         True         False        False           Fal
se
2     69        True        False         False         True           Fal
se
3     89       False        False         False        False            Tr
ue

   city_surat  packege_13lk  packege_9lk
0       False         False        False
1        True         False         True
2       False          True        False
3       False         False         True
```

In [ ]: