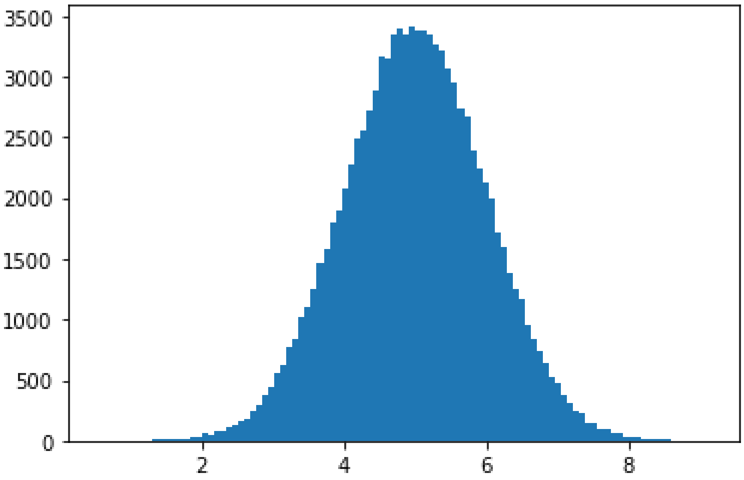
In [1]:

```
#1.Write a python program to find mean, mode, median.
list1 = [12, 16, 20, 20, 12, 30, 25, 23, 24, 20]
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)
frequency = {}
for i in list1:
    frequency.setdefault(i, 0)
    frequency[i]+=1

frequent = max(frequency.values())
for i, j in frequency.items():
    if j == frequent:
        mode = i
print("Mode",mode)
```

```
Mean= 20.2
Median= 20.0
Mode 20
```

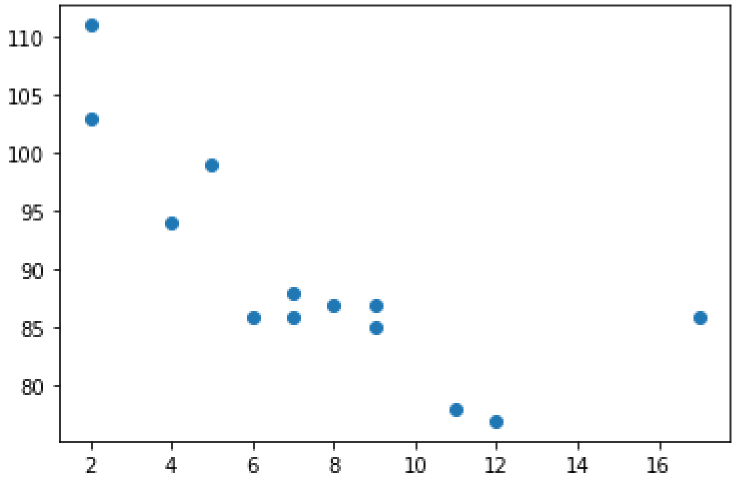
In [2]:

```
#2.Write a python program to typical normal data distribution.
import numpy
import matplotlib.pyplot as plt
x = numpy.random.normal(5.0, 1.0, 100000)
plt.hist(x, 100)
plt.show()
```



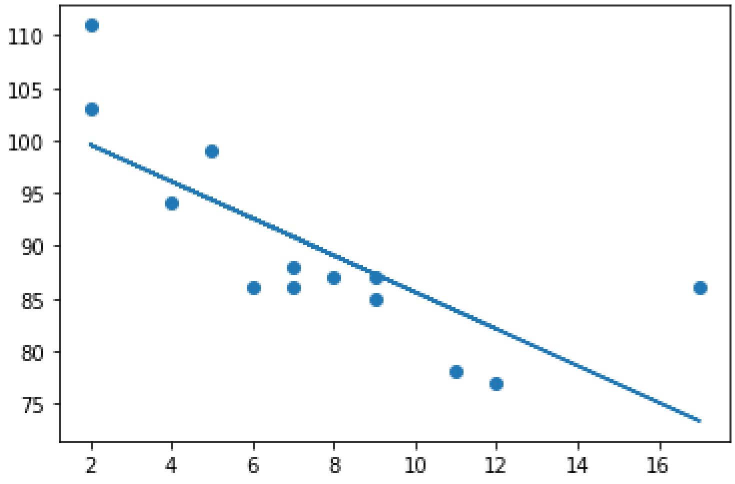
In [3]:

```
#3.Write a python program to draw scatter plot of linear regression
import matplotlib.pyplot as plt
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
plt.scatter(x, y)
plt.show()
```



In [4]:

```
#4.Write a python program to draw the line of Linear Regression.
import matplotlib.pyplot as plt
from scipy import stats
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
slope, intercept, r, p, std_err = stats.linregress(x, y)
def myfunc(x):
  return slope * x + intercept
mymodel = list(map(myfunc, x))
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```



In [23]:

```
#5.Write a python program to predict the speed of a 5 years old car.
def predict_car_speed(age):
    initial_speed = 100  # mph
    speed_decrease_per_year = 2  # mph/year
    predicted_speed = initial_speed - speed_decrease_per_year * age
    if predicted_speed < 0:
        predicted_speed = 0  # Speed cannot be negative
    return predicted_speed
age_of_car = 5
predicted_speed = predict_car_speed(age_of_car)
print(f"The predicted speed of a {age_of_car} years old car is {predicted_sp
```

```
The predicted speed of a 5 years old car is 90 mph.
```

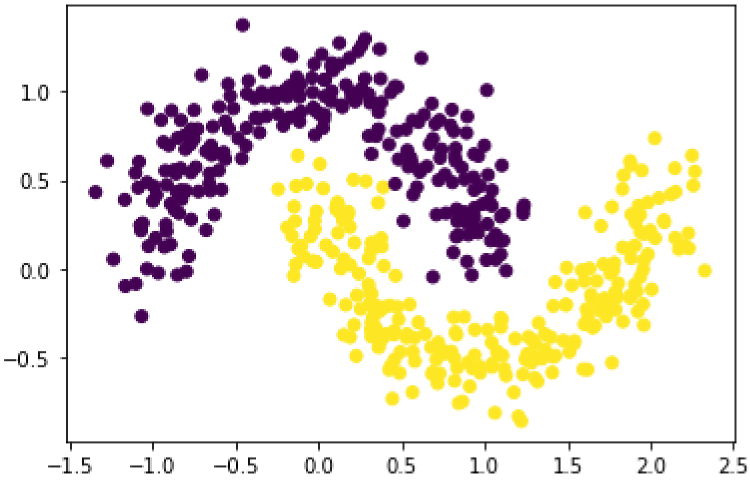
In [24]:

```
#6.Write a python program to print the coefficient values of the regression
import numpy
from sklearn import linear_model
X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52,
y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])logr = linear_model.Log
logr.fit(X,y)
predicted = logr.predict(numpy.array([3.46]).reshape(-1,1))
print(predicted)
```

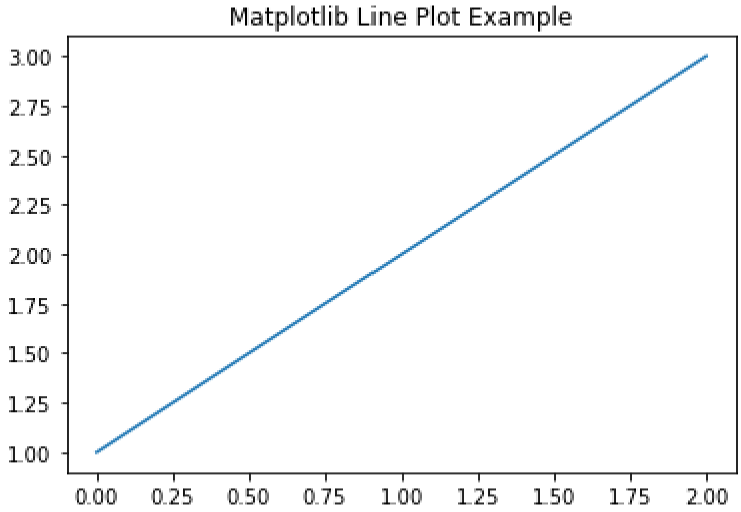
```
  File "<ipython-input-24-d413a0f3ff5f>", line 5
    y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])logr = linear_mod
el.LogisticRegression()
                                                         ^

SyntaxError: invalid syntax
```

In [25]:

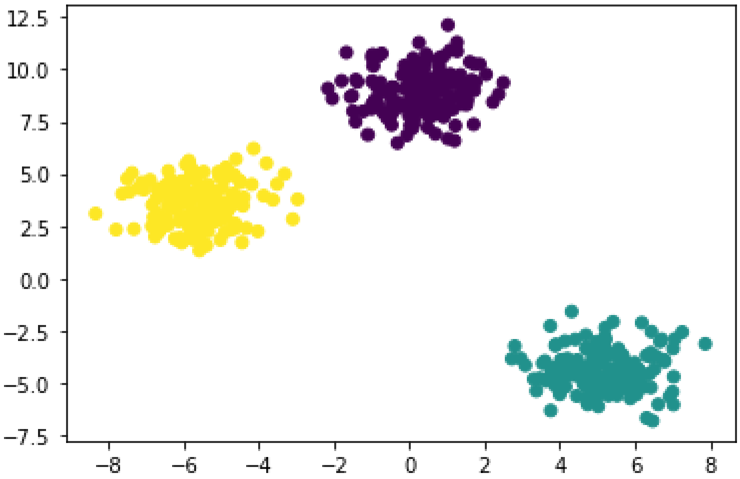
```
#7.Write a python program to 2d binary classification data generated by make
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
X, y = make_moons(n_samples=500, shuffle=True,
                noise=0.15, random_state=42)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```



In [26]:
```
#8. Write a python program to display the plot we can use the functions plot
import matplotlib.pyplot as plt
import numpy as np
plt.plot([1, 2, 3])
plt.title('Matplotlib Line Plot Example')
plt.draw()
plt.show()
```

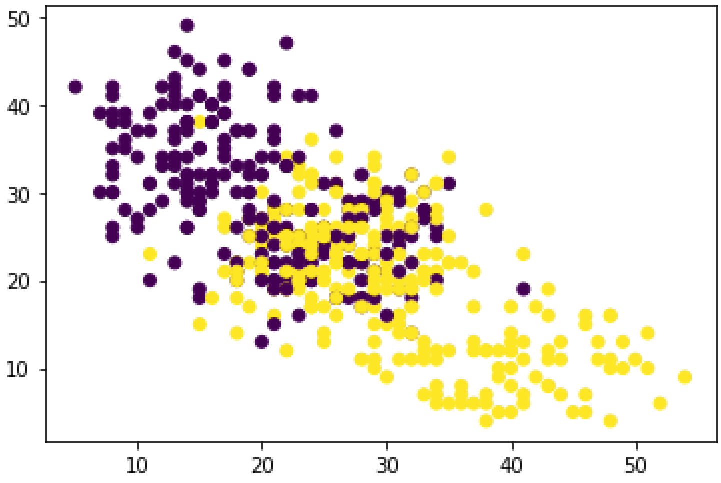


In [27]:
```
#9.Write a python program to data generated by the function make_blobs() are
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
X, y = make_blobs(n_samples=500, centers=3, n_features=2, random_state=23)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```



```
In [17]: #10.Write a python program to random multi-label classification data is cre
         from sklearn.datasets import make_multilabel_classification
         import pandas as pd
         import matplotlib.pyplot as plt
         X, y = make_multilabel_classification(n_samples=500, n_features=2,
                                               n_classes=2, n_labels=2,
                                               allow_unlabeled=True,
                                               random_state=23)
         df = pd.concat([pd.DataFrame(X, columns=['X1', 'X2']),
                         pd.DataFrame(y, columns=['Label1', 'Label2'])],
                     axis=1)
         display(df.head())
         plt.scatter(df['X1'], df['X2'], c=df['Label1'])
         plt.show()
```

	X1	X2	Label1	Label2
0	14.0	34.0	0	1
1	30.0	22.0	1	1
2	29.0	19.0	1	1
3	21.0	19.0	1	1
4	16.0	32.0	0	1



```
In [28]: #11.Write a python program to implement the KNN algorithm.
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.datasets import load_iris
         irisData = load_iris()
         X = irisData.data
         y = irisData.target
         X_train, X_test, y_train, y_test = train_test_split(
                     X, y, test_size = 0.2, random_state=42)
         knn = KNeighborsClassifier(n_neighbors=7)
         knn.fit(X_train, y_train)
         print(knn.predict(X_test))
```

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

In [21]:

```
#Write a python program to creating a dataframe to implement one hot encodin
import numpy as np
import pandas as pd
data = pd.read_csv('dataset.csv')
print(data.head())
```

```
  GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  \
0      M   65        1               1        1              2   
1      F   55        1               2        2              1   
2      F   78        2               2        1              1   
3      M   60        2               1        1              1   
4      F   80        1               1        2              1   

   CHRONIC_DISEASE  FATIGUE  ALLERGY  WHEEZING  ALCOHOL_CONSUMING  COUGHIN
G  \
0                2        1        2         2                  2        
2   
1                1        2        2         2                  1        
1   
2                1        2        1         2                  1        
1   
3                2        1        2         1                  1        
2   
4                1        2        1         2                  1        
1   

   SHORTNESS_OF_BREATH  SWALLOWING_DIFFICULTY  CHEST_PAIN LUNG_CANCER  
0                    2                      2           1          NO  
1                    1                      2           2          NO  
2                    2                      1           1         YES  
3                    1                      2           2         YES  
4                    1                      1           2          NO  
```

In [ ]:

```
```