## Exercise - 9

In [134]:
```
import pandas as pd
import numpy as np
from apyori import apriori
```

In [117]:
```
df = pd.read_csv('Groceries.csv',header=None)
df.head()
```

Out[117]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | shrimp | almonds | avocado | vegetables mix | green grapes | whole weat flour | yams | cottage cheese | energy drink | tomato juice | low fat yogurt | green tea | honey | salad | mineral water | salmon | ant |
| **1** | burgers | meatballs | eggs | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **2** | chutney | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **3** | turkey | avocado | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **4** | mineral water | milk | energy bar | whole wheat rice | green tea | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

In [118]:
```
trans = []
for i in range(7501):
    trans.append(
        [df.values[i,j] for j in range(20) if df.values[i,j] is not np.nan])
```

In [132]:
```
rules = apriori(transactions=trans,min_support=0.0045,min_confidence=0.3,min_lift=3,max_length=2)
association_rules = list(rules)
```

In [133]:

```
for item in association_rules:
    items = [x for x in item[0]]
    print("Rule: " + items[0] + " -> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

```
Rule: escalope -> mushroom cream sauce
Support: 0.005732568990801226
Confidence: 0.3006993006993007
Lift: 3.790832696715049
====================================
Rule: escalope -> pasta
Support: 0.005865884548726837
Confidence: 0.3728813559322034
Lift: 4.700811850163794
====================================
Rule: ground beef -> herb & pepper
Support: 0.015997866951073192
Confidence: 0.3234501347708895
Lift: 3.2919938411349285
====================================
Rule: ground beef -> tomato sauce
Support: 0.005332622317024397
Confidence: 0.3773584905660377
Lift: 3.840659481324083
====================================
Rule: shrimp -> pasta
Support: 0.005065991201173177
Confidence: 0.3220338983050847
Lift: 4.506672147735896
====================================
```

## Exercise - 10

In [1]:

```
import pandas as pd
import scipy.stats as stats
```

In [2]:
```
df = pd.read_csv('tips.csv')
df.head()
```

Out[2]:

| | total_bill | tip | sex | smoker | day | time | size | price_per_person | Payer Name | CC Number | Payment ID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 | 8.49 | Christy Cunningham | 3560325168603410 | Sun2959 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 | 3.45 | Douglas Tucker | 4478071379779230 | Sun4608 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 | 7.00 | Travis Walters | 6011812112971322 | Sun4458 |
| **3** | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 | 11.84 | Nathaniel Harris | 4676137647685994 | Sun5260 |
| **4** | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 | 6.15 | Tonya Carter | 4832732618637221 | Sun2251 |

In [7]:
```
tab = pd.crosstab(df['sex'],df['time'])
print(tab)
chi2,p,dof,expected = stats.chi2_contingency(tab)
print('Chi-Square Value :',round(chi2,5))
print('P Value :',round(p,5))
print('Relation') if p <= 0.05 else print('No Relation')
```

```
time    Dinner  Lunch
sex
Female      52     35
Male       124     33
Chi-Square Value : 9.34381
P Value : 0.00224
Relation
```

## Exercise - 11

In [1]:
```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [2]:
```
df = pd.read_csv('iris.csv')
df.head(5)
```

Out[2]:

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

In [4]:
```
X = df.drop('species',axis=1)
y = df['species']
```

In [5]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=101)
model = GaussianNB()
model.fit(X_train,y_train)
```

Out[5]:
```
▾ GaussianNB

GaussianNB()
```

In [7]:
```
preds = model.predict(X_test)
confusion_matrix(preds,y_test)
```

Out[7]:
```
array([[13,  0,  0],
       [ 0, 19,  1],
       [ 0,  1, 11]], dtype=int64)
```

In [8]:
```
accuracy_score(preds,y_test)
```

Out[8]:
```
0.9555555555555556
```

## Exercise - 14

In [9]:
```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

In [10]:
```
df = pd.read_csv('iris.csv')
df.head(5)
```

Out[10]:

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

In [11]:
```
X = df.drop('species',axis=1)
```

In [12]:
```
model = KMeans(n_clusters=3)
model.fit(X)
```

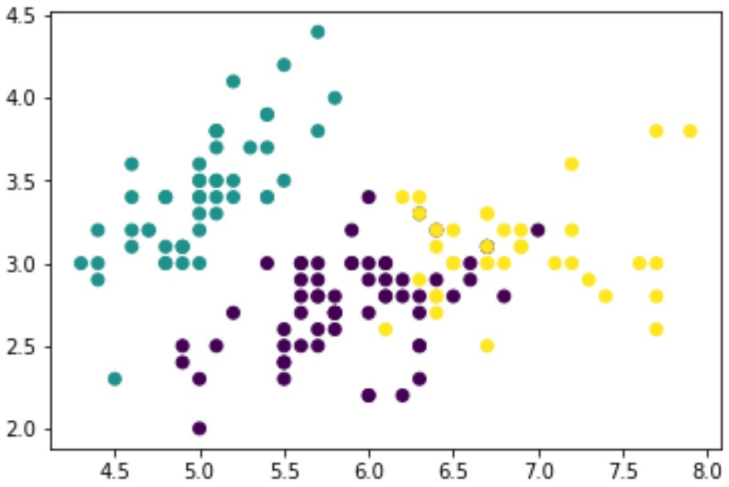
Out[12]:
```
▾        KMeans

KMeans(n_clusters=3)
```

In [13]:
```
model.labels_
```

Out[13]:
```
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, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

In [14]:
```
1 plt.scatter(df['sepal_length'],df['sepal_width'],c=model.labels_)
```

Out[14]: <matplotlib.collections.PathCollection at 0x1d5c250c340>




## Exercise - 15

In [41]:
```
1  from scipy.spatial import distance_matrix
2  import numpy as np
3
4
5  def GetMatrix(text, metric):
6      rows = text.split('\n')
7      rows = [row.strip() for row in rows if row.strip() != '']
8      mat = [list(map(int, row.split(' '))) for row in rows]
9      dist_mat = distance_matrix(mat, mat, p=metric)
10     dist_mat = np.round(np.matrix(dist_mat), 2)
11     return dist_mat
```

In [42]:
```
print('-------Metrics-------')
print('1. Manhattan Distance')
print('2. Euclidean Distance')
print('3. Mahalanobis Distance')


text = '''
        1 2 3
        4 5 6
        7 8 9
        1 4 5
       '''
metirc = int(input('Enter Metric : '))
print(GetMatrix(text, metirc))
```

```
-------Metrics-------
1. Manhattan Distance
2. Euclidean Distance
3. Mahalanobis Distance
Enter Metric : 2
[[ 0.    5.2  10.39  2.83]
 [ 5.2   0.    5.2   3.32]
 [10.39  5.2   0.    8.25]
 [ 2.83  3.32  8.25  0.  ]]
```

## Exercise - 16

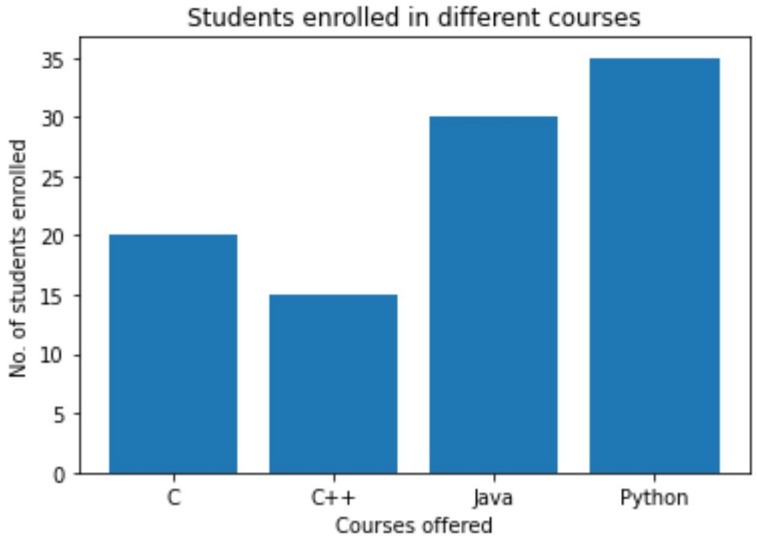
In [51]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [44]:

```
data = {'C':20, 'C++':15, 'Java':30,'Python':35}
courses = list(data.keys())
values = list(data.values())

plt.bar(courses,values)
plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
```

Out[44]:

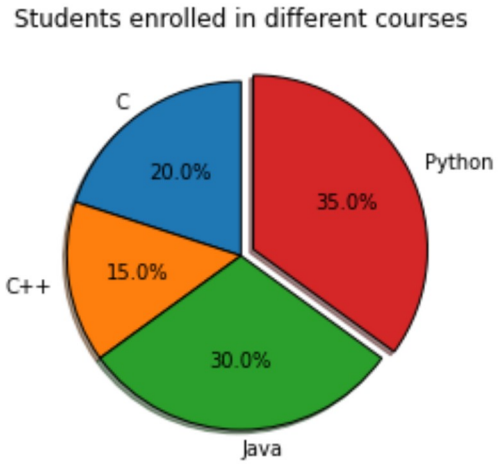
```
Text(0.5, 1.0, 'Students enrolled in different courses')
```



In [45]:

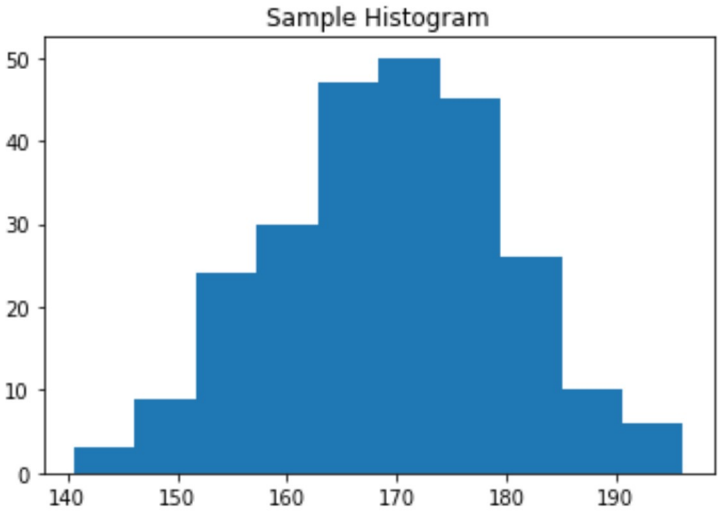
```
data = {'C':20, 'C++':15, 'Java':30,'Python':35}
labels = list(data.keys())
slices = list(data.values())
explode = [0, 0, 0, 0.08]
plt.pie(slices, labels=labels, explode=explode, shadow=True,
        startangle=90, autopct='%1.1f%%', wedgeprops={'edgecolor': 'black'})
plt.title("Students enrolled in different courses")
```

Out[45]: Text(0.5, 1.0, 'Students enrolled in different courses')



In [46]:
```
1 data =  np.random.normal(170, 10, 250)
2 plt.hist(data)
3 plt.title("Sample Histogram")
```

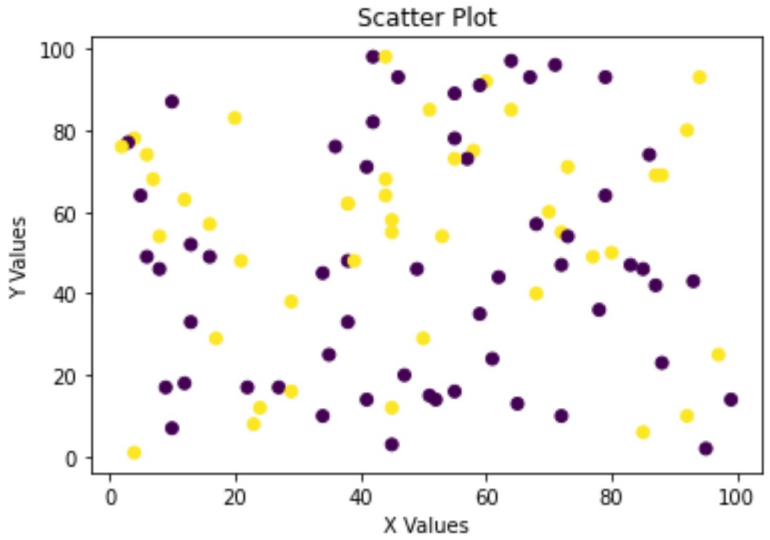
Out[46]:
```
Text(0.5, 1.0, 'Sample Histogram')
```



In [47]:

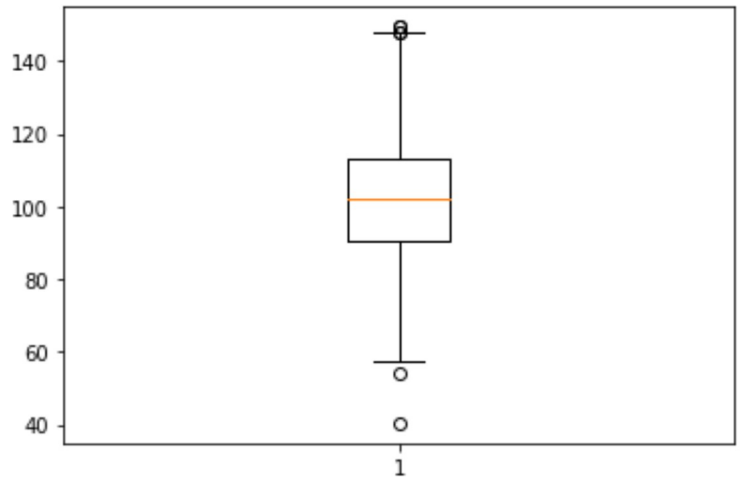
```
x = np.random.randint(1,100,size=(100,))
y = np.random.randint(1,100,size=(100,))
labels = np.random.randint(0,2,size=(100,))

plt.scatter(x,y,c=labels)
plt.xlabel("X Values")
plt.ylabel("Y Values")
plt.title("Scatter Plot")
```

Out[47]: Text(0.5, 1.0, 'Scatter Plot')



In [50]:

```
1 np.random.seed(10)
2 data = np.random.normal(100, 20, 200)
3 plt.boxplot(data);
```




In [ ]:

```
1
```