Guru Nanak College of Arts, Science & Commerce
GTB Nagar, Sion Koliwada, Mumbai – 37

Department of Information Technology

CERTIFICATE

This is to certify that Mr./Ms. **Jameel Shaikh**, Seat No. **3269635** studying in **Master of Science** in **Information Technology Part II (Semester III)** has satisfactorily completed the Practical of PSIT3P2a **Applied Artificial Intelligence** as prescribed by University of Mumbai, during the academic year **2022-23**.

____

Signature
Subject-In-Charge

____

Signature
Head of the Department

____

Signature
External Examiner

College Seal: ____________________

Date: ____________________

INDEX

Practical 01

Aim: Design an Expert system using AIML.

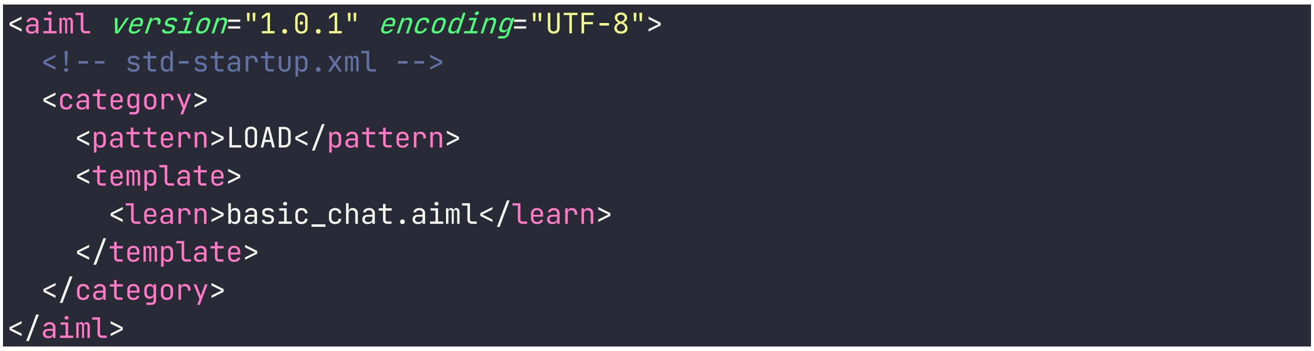
Description:

Code:

basic_chat.aiml:

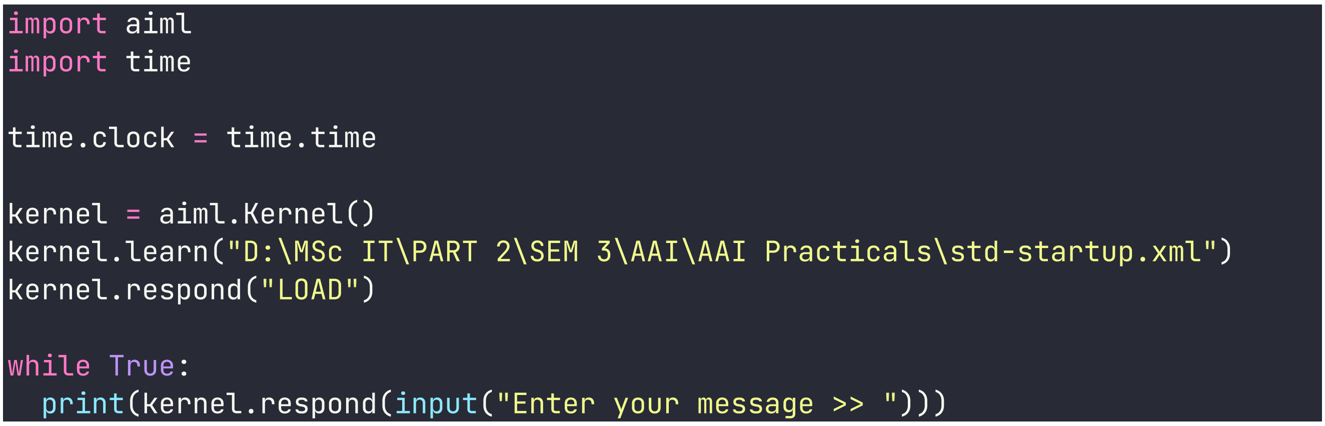
```
<aiml version="1.0.1" encoding="UTF-8">
<!-- basic_chat.aiml -->
 <category>
 <pattern>HELLO I AM *</pattern>
 <template> HELLO <set name="username"> <star/> </set> </template>
 </category>
 <category>
<pattern>I LIKE * COLOR</pattern>
<template><star index="1"/> is a nice color.</template>
 </category>
 <category>
 <pattern>BYE</pattern>
<template> BYE <get name="username"/> THANKS FOR THE CONVERSATION.
</template>
 </category>
</aiml>
```

std-startup.xml:



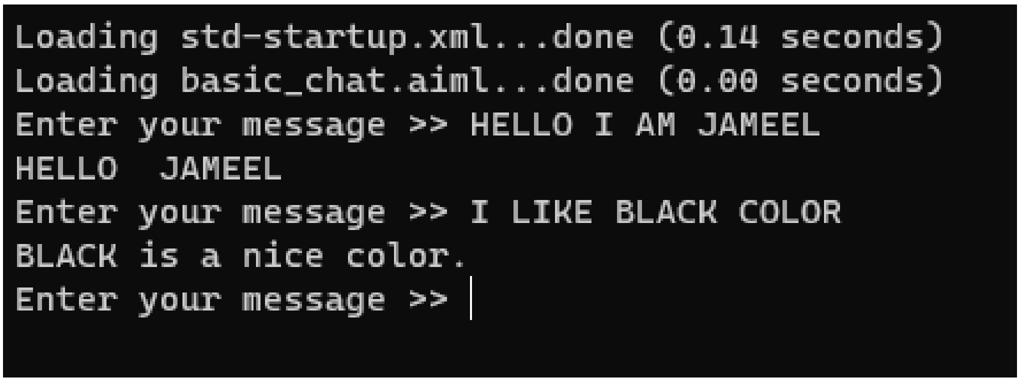
```
<aiml version="1.0.1" encoding="UTF-8">
  <!-- std-startup.xml -->
  <category>
    <pattern>LOAD</pattern>
    <template>
      <learn>basic_chat.aiml</learn>
    </template>
  </category>
</aiml>
```

test_bot.py:



```
import aiml
import time

time.clock = time.time

kernel = aiml.Kernel()
kernel.learn("D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals\std-startup.xml")
kernel.respond("LOAD")

while True:
  print(kernel.respond(input("Enter your message >> ")))
```

Output:



```
Loading std-startup.xml...done (0.14 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
Enter your message >> HELLO I AM JAMEEL
HELLO  JAMEEL
Enter your message >> I LIKE BLACK COLOR
BLACK is a nice color.
Enter your message >>
```

Practical 02

Aim: Design a bot using AIML.

Description:

Code:

basic_chat.aiml:

```
<aiml version="1.0.1" encoding="UTF-8">
<!-- basic_chat.aiml -->
 <category>
 <pattern>HELLO I AM *</pattern>
 <template> HELLO <set name="username"> <star/> </set> </template>
 </category>
 <category>
<pattern>I LIKE * COLOR</pattern>
<template><star index="1"/> is a nice color.</template>
 </category>
 <category>
 <pattern>BYE</pattern>
<template> BYE <get name="username"/> THANKS FOR THE CONVERSATION.
</template>
 </category>
</aiml>
```

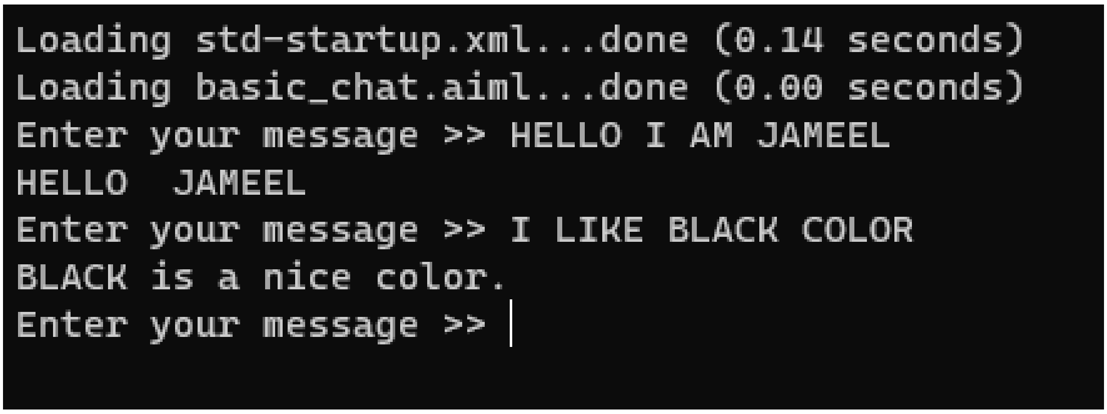
std-startup.xml:

```
<aiml version="1.0.1" encoding="UTF-8">
  <!-- std-startup.xml -->
  <category>
    <pattern>LOAD</pattern>
    <template>
      <learn>basic_chat.aiml</learn>
    </template>
  </category>
</aiml>
```

test_bot.py:

```
import aiml
import time

time.clock = time.time

kernel = aiml.Kernel()
kernel.learn("D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals\std-startup.xml")
kernel.respond("LOAD")

while True:
  print(kernel.respond(input("Enter your message >> ")))
```

Output:



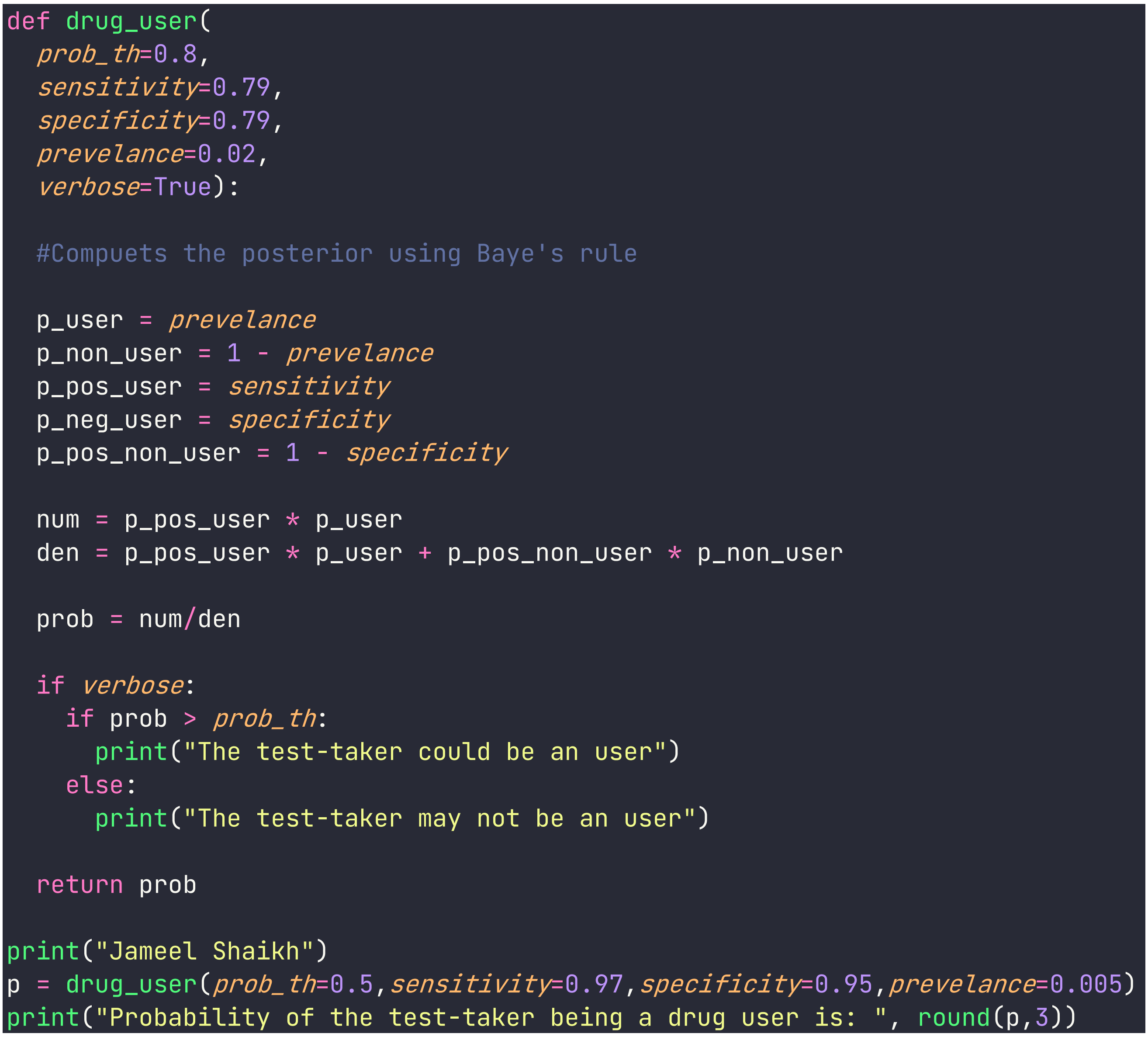
```
Loading std-startup.xml...done (0.14 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
Enter your message >> HELLO I AM JAMEEL
HELLO  JAMEEL
Enter your message >> I LIKE BLACK COLOR
BLACK is a nice color.
Enter your message >>
```

Practical 03

Aim: **Implement Bayes Theorem using Python.**

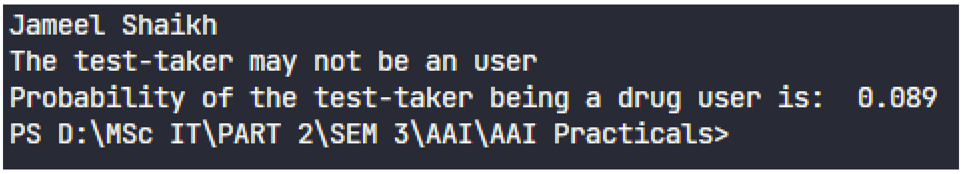
Description:

Code:



```
def drug_user(
  prob_th=0.8,
  sensitivity=0.79,
  specificity=0.79,
  prevelance=0.02,
  verbose=True):

  #Compuets the posterior using Baye's rule

  p_user = prevelance
  p_non_user = 1 - prevelance
  p_pos_user = sensitivity
  p_neg_user = specificity
  p_pos_non_user = 1 - specificity

  num = p_pos_user * p_user
  den = p_pos_user * p_user + p_pos_non_user * p_non_user

  prob = num/den

  if verbose:
    if prob > prob_th:
      print("The test-taker could be an user")
    else:
      print("The test-taker may not be an user")

  return prob

print("Jameel Shaikh")
p = drug_user(prob_th=0.5,sensitivity=0.97,specificity=0.95,prevelance=0.005)
print("Probability of the test-taker being a drug user is: ", round(p,3))
```

Output:



```
Jameel Shaikh
The test-taker may not be an user
Probability of the test-taker being a drug user is:  0.089
PS D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals>
```

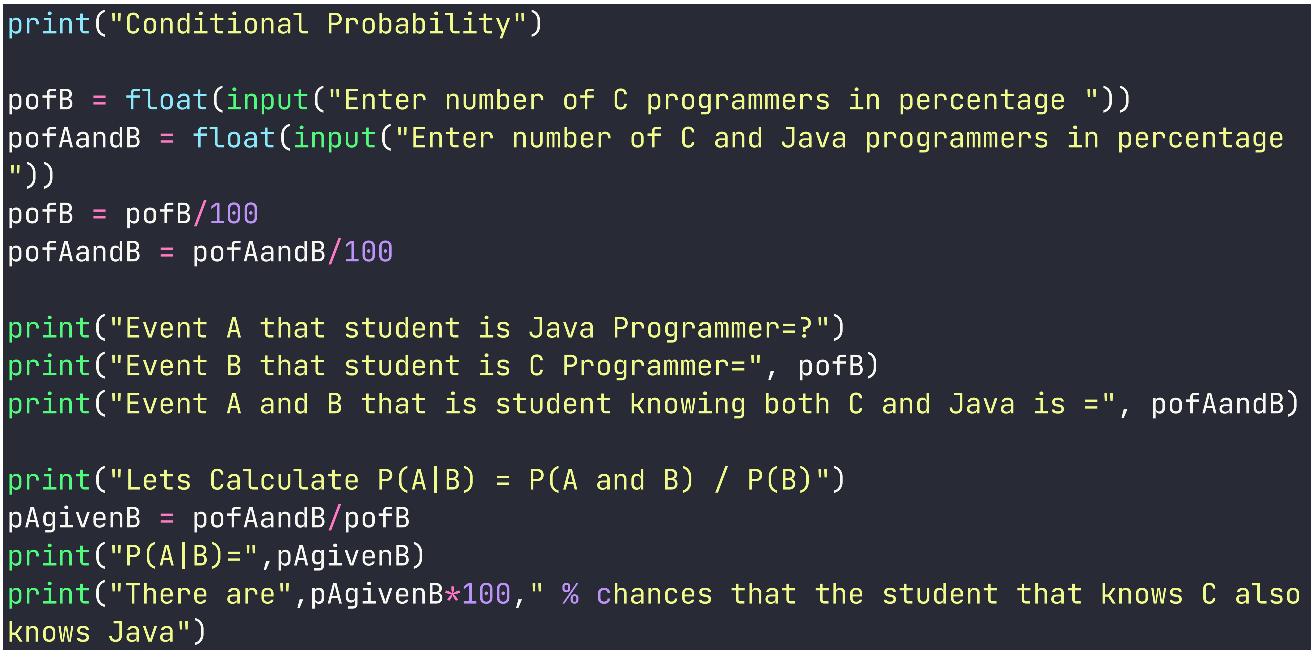
Practical 04

Aim:

A) Implement Conditional Probability using Python.

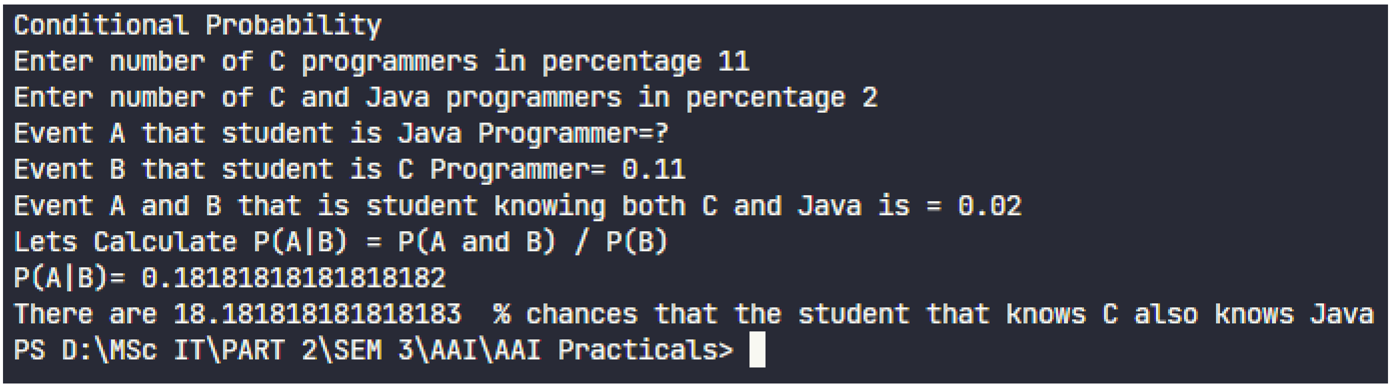
Description:

Code:



```
print("Conditional Probability")

pofB = float(input("Enter number of C programmers in percentage "))
pofAandB = float(input("Enter number of C and Java programmers in percentage
"))
pofB = pofB/100
pofAandB = pofAandB/100

print("Event A that student is Java Programmer=?")
print("Event B that student is C Programmer=", pofB)
print("Event A and B that is student knowing both C and Java is =", pofAandB)

print("Lets Calculate P(A|B) = P(A and B) / P(B)")
pAgivenB = pofAandB/pofB
print("P(A|B)=",pAgivenB)
print("There are",pAgivenB*100," % chances that the student that knows C also
knows Java")
```

Output:

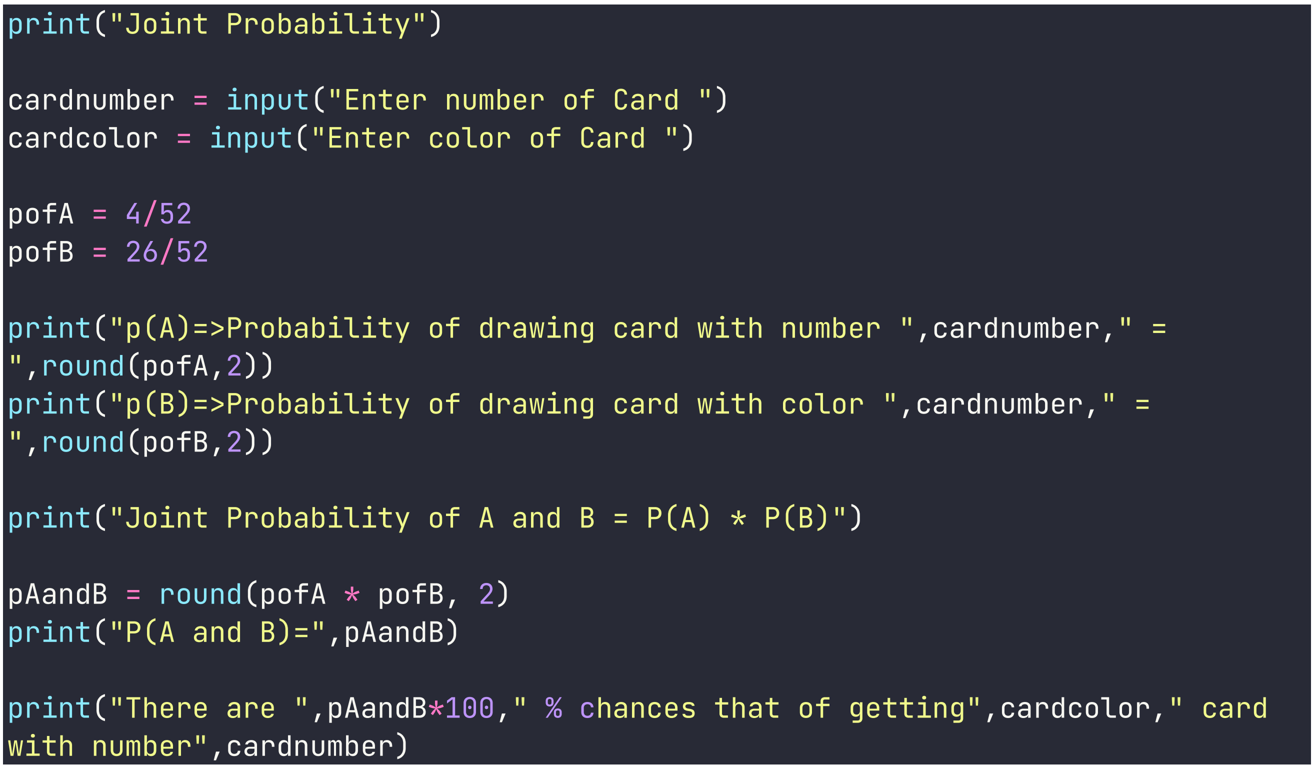


```
Conditional Probability
Enter number of C programmers in percentage 11
Enter number of C and Java programmers in percentage 2
Event A that student is Java Programmer=?
Event B that student is C Programmer= 0.11
Event A and B that is student knowing both C and Java is = 0.02
Lets Calculate P(A|B) = P(A and B) / P(B)
P(A|B)= 0.18181818181818182
There are 18.181818181818183  % chances that the student that knows C also knows Java
PS D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals> 
```

B) Implement Joint Probability using Python.

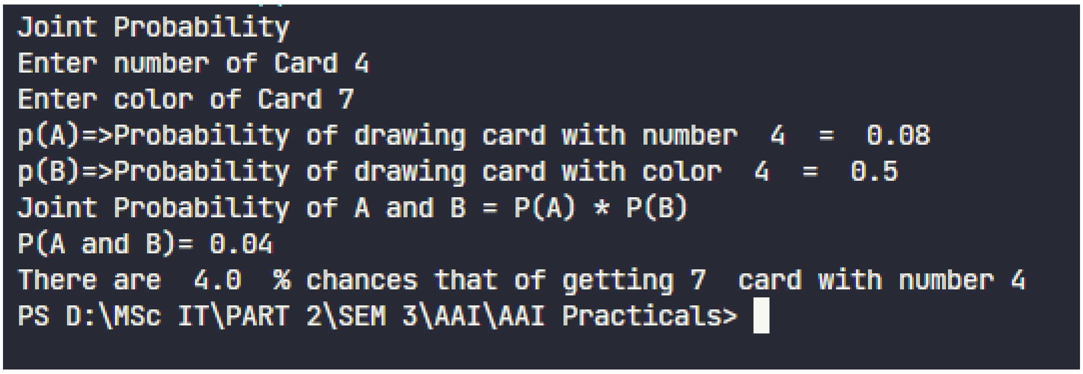
Description:

Code:



```
print("Joint Probability")

cardnumber = input("Enter number of Card ")
cardcolor = input("Enter color of Card ")

pofA = 4/52
pofB = 26/52

print("p(A)=>Probability of drawing card with number ",cardnumber," =
",round(pofA,2))
print("p(B)=>Probability of drawing card with color ",cardnumber," =
",round(pofB,2))

print("Joint Probability of A and B = P(A) * P(B)")

pAandB = round(pofA * pofB, 2)
print("P(A and B)=",pAandB)

print("There are ",pAandB*100," % chances that of getting",cardcolor," card
with number",cardnumber)
```

Output:



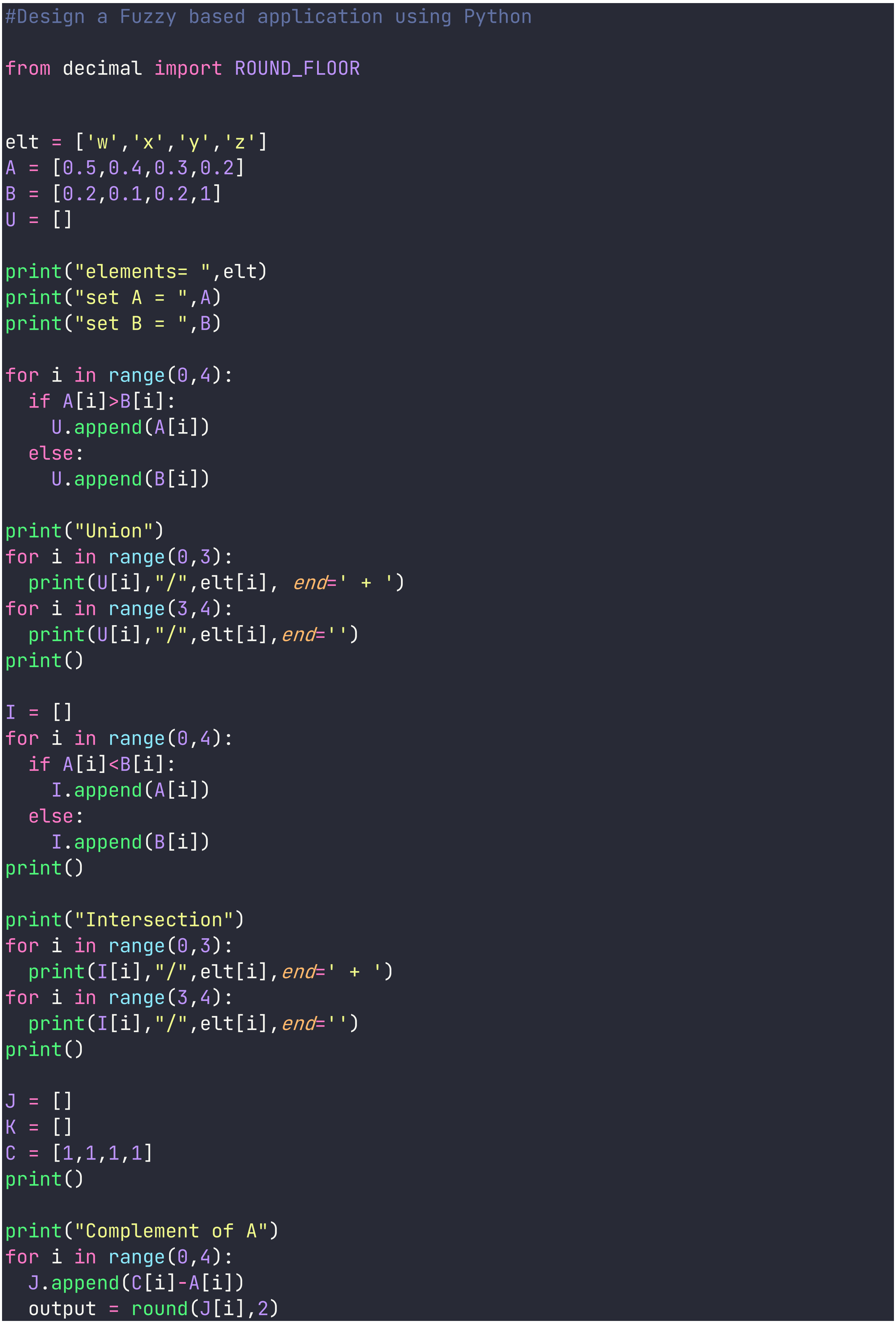
```
Joint Probability
Enter number of Card 4
Enter color of Card 7
p(A)=>Probability of drawing card with number  4  =  0.08
p(B)=>Probability of drawing card with color  4  =  0.5
Joint Probability of A and B = P(A) * P(B)
P(A and B)= 0.04
There are  4.0  % chances that of getting 7  card with number 4
PS D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals>
```

Practical 05

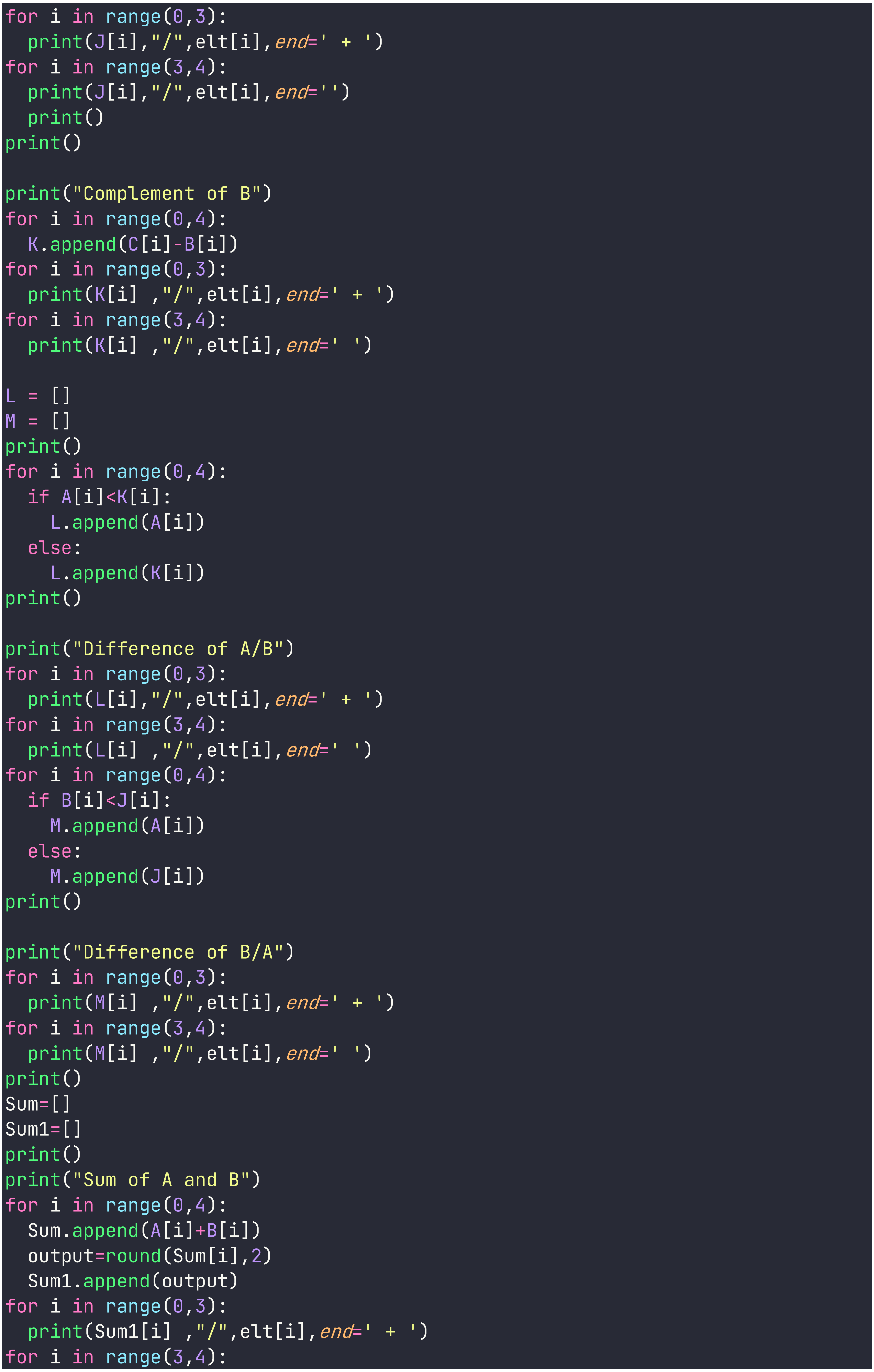
Aim: **Design a Fuzzy based application using Python.**

Description:

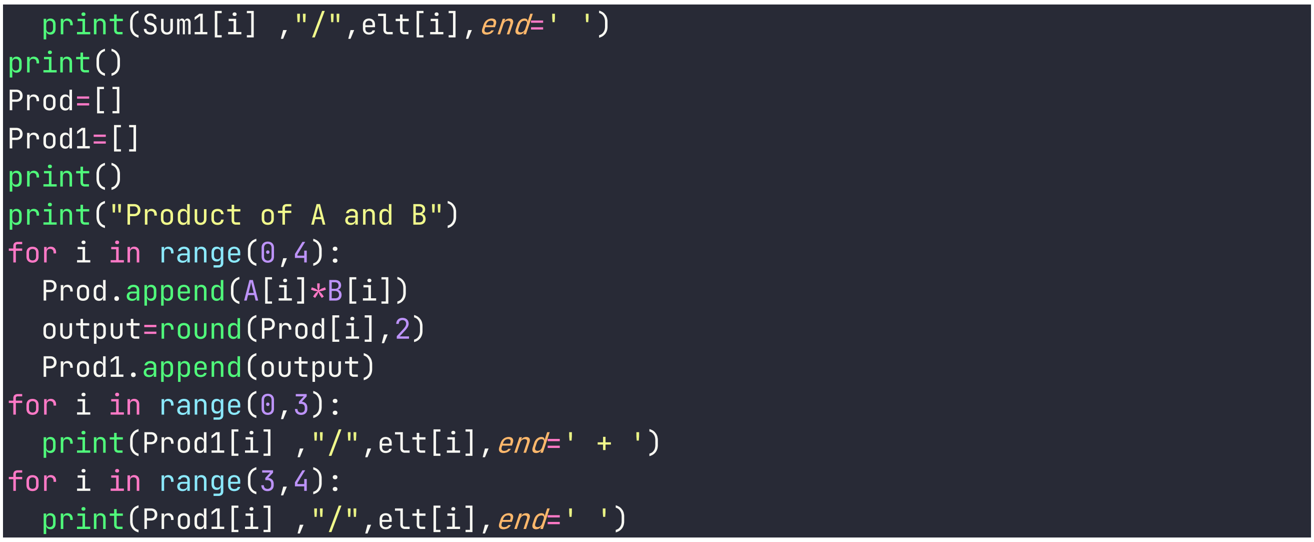
Code:



```
#Design a Fuzzy based application using Python

from decimal import ROUND_FLOOR


elt = ['w','x','y','z']
A = [0.5,0.4,0.3,0.2]
B = [0.2,0.1,0.2,1]
U = []

print("elements= ",elt)
print("set A = ",A)
print("set B = ",B)

for i in range(0,4):
  if A[i]>B[i]:
    U.append(A[i])
  else:
    U.append(B[i])

print("Union")
for i in range(0,3):
  print(U[i],"/",elt[i], end=' + ')
for i in range(3,4):
  print(U[i],"/",elt[i],end='')
print()

I = []
for i in range(0,4):
  if A[i]<B[i]:
    I.append(A[i])
  else:
    I.append(B[i])
print()

print("Intersection")
for i in range(0,3):
  print(I[i],"/",elt[i],end=' + ')
for i in range(3,4):
  print(I[i],"/",elt[i],end='')
print()

J = []
K = []
C = [1,1,1,1]
print()

print("Complement of A")
for i in range(0,4):
  J.append(C[i]-A[i])
  output = round(J[i],2)
```

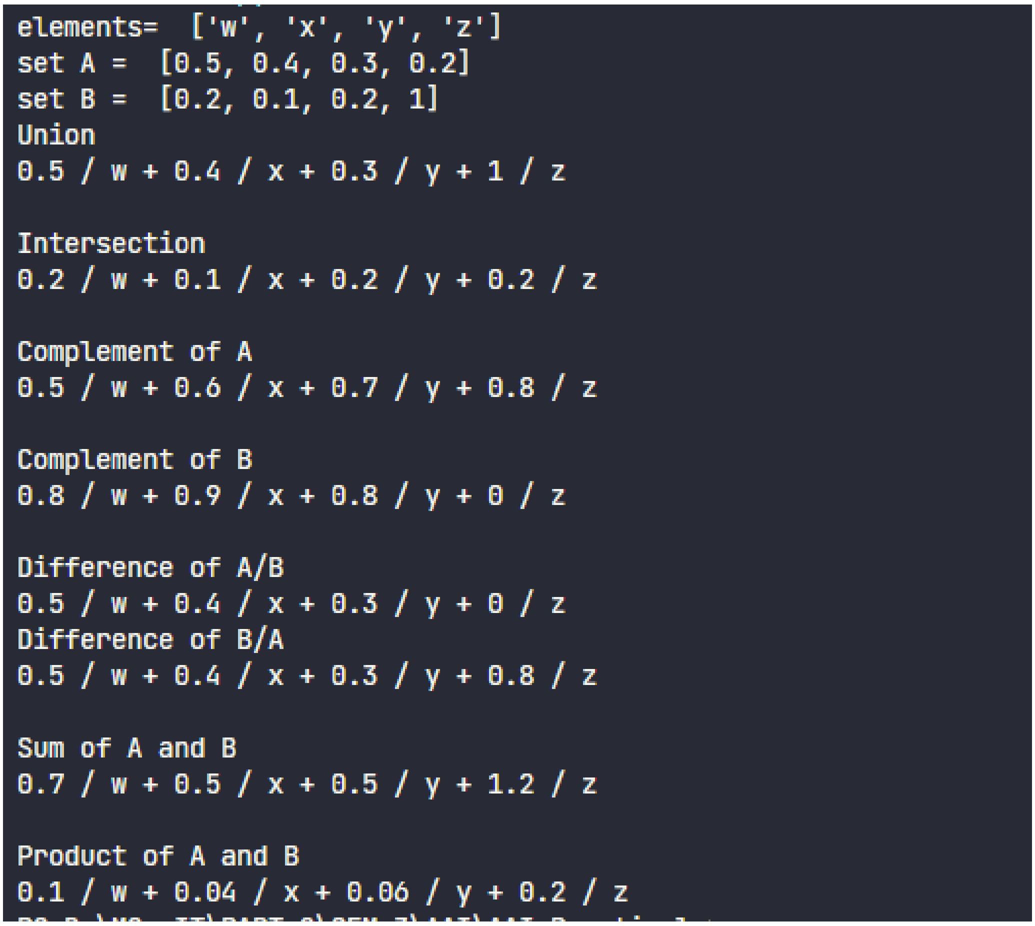


```
for i in range(0,3):
  print(J[i],"/",elt[i],end=' + ')
for i in range(3,4):
  print(J[i],"/",elt[i],end='')
  print()
print()

print("Complement of B")
for i in range(0,4):
  K.append(C[i]-B[i])
for i in range(0,3):
  print(K[i] ,"/",elt[i],end=' + ')
for i in range(3,4):
  print(K[i] ,"/",elt[i],end=' ')

L = []
M = []
print()
for i in range(0,4):
  if A[i]<K[i]:
    L.append(A[i])
  else:
    L.append(K[i])
print()

print("Difference of A/B")
for i in range(0,3):
  print(L[i],"/",elt[i],end=' + ')
for i in range(3,4):
  print(L[i] ,"/",elt[i],end=' ')
for i in range(0,4):
  if B[i]<J[i]:
    M.append(A[i])
  else:
    M.append(J[i])
print()

print("Difference of B/A")
for i in range(0,3):
  print(M[i] ,"/",elt[i],end=' + ')
for i in range(3,4):
  print(M[i] ,"/",elt[i],end=' ')
print()
Sum=[]
Sum1=[]
print()
print("Sum of A and B")
for i in range(0,4):
  Sum.append(A[i]+B[i])
  output=round(Sum[i],2)
  Sum1.append(output)
for i in range(0,3):
  print(Sum1[i] ,"/",elt[i],end=' + ')
for i in range(3,4):
```



```
  print(Sum1[i] ,"/",elt[i],end=' ')
print()
Prod=[]
Prod1=[]
print()
print("Product of A and B")
for i in range(0,4):
  Prod.append(A[i]*B[i])
  output=round(Prod[i],2)
  Prod1.append(output)
for i in range(0,3):
  print(Prod1[i] ,"/",elt[i],end=' + ')
for i in range(3,4):
  print(Prod1[i] ,"/",elt[i],end=' ')
```

Output:



```
elements=  ['w', 'x', 'y', 'z']
set A =  [0.5, 0.4, 0.3, 0.2]
set B =  [0.2, 0.1, 0.2, 1]
Union
0.5 / w + 0.4 / x + 0.3 / y + 1 / z

Intersection
0.2 / w + 0.1 / x + 0.2 / y + 0.2 / z

Complement of A
0.5 / w + 0.6 / x + 0.7 / y + 0.8 / z

Complement of B
0.8 / w + 0.9 / x + 0.8 / y + 0 / z

Difference of A/B
0.5 / w + 0.4 / x + 0.3 / y + 0 / z
Difference of B/A
0.5 / w + 0.4 / x + 0.3 / y + 0.8 / z

Sum of A and B
0.7 / w + 0.5 / x + 0.5 / y + 1.2 / z

Product of A and B
0.1 / w + 0.04 / x + 0.06 / y + 0.2 / z
```

Practical 06

Aim: Write an application to simulate supervised and un-supervised learning model.

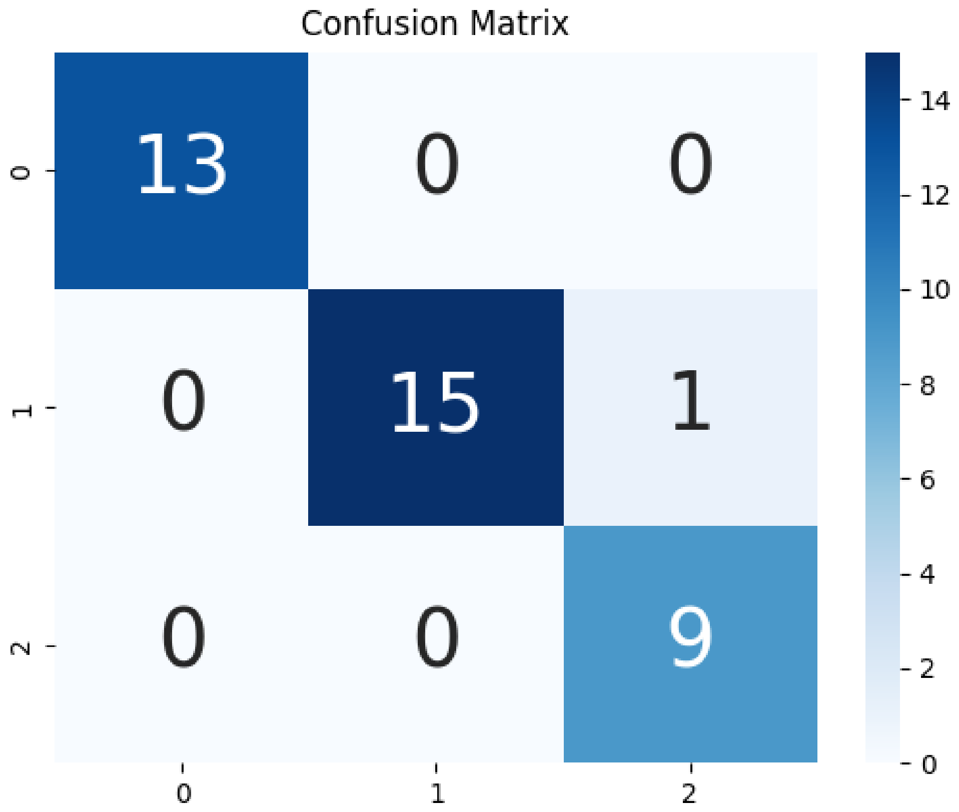
A) Supervised Learning Model

Description:

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
import seaborn as sns

print("Supervised Learning Model")
#Importing the datasets
dataset = pd.read_csv("D:\\MSc IT\\PART 2\\SEM 3\\AAI\\AAI
Practicals\\prac7\\iris.csv")
dataset.describe()

#Splitting the dataset into the Training set and test set
x = dataset.iloc[:, [0,1,2,3]].values
y = dataset.iloc[:, 4].values


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=0)


sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#Fitting Logistic Regression to the Training set
classifier = LogisticRegression(random_state=0, solver='lbfgs',
multi_class='auto')
classifier.fit(x_train, y_train)

#Predicting the Test set results
y_pred = classifier.predict(x_test)

#Predict probabilities
probs_y = classifier.predict_proba(x_test)

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

#Plot confusion matrix
#confusion matrix sns heatmap
ax = plt.axes()
df_cm = cm
sns.heatmap(df_cm, annot=True, annot_kws={"size":30}, fmt='d', cmap="Blues",
ax = ax)
ax.set_title("Confusion Matrix")
plt.show()
```

Output:



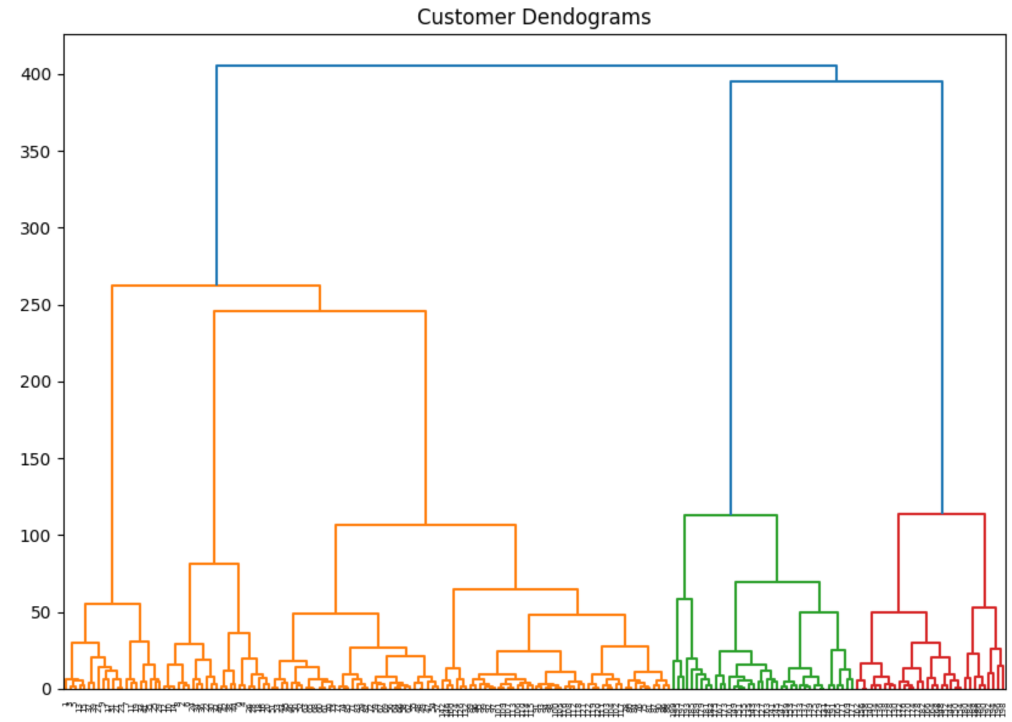
B) Un-supervised Learning Model

Description:

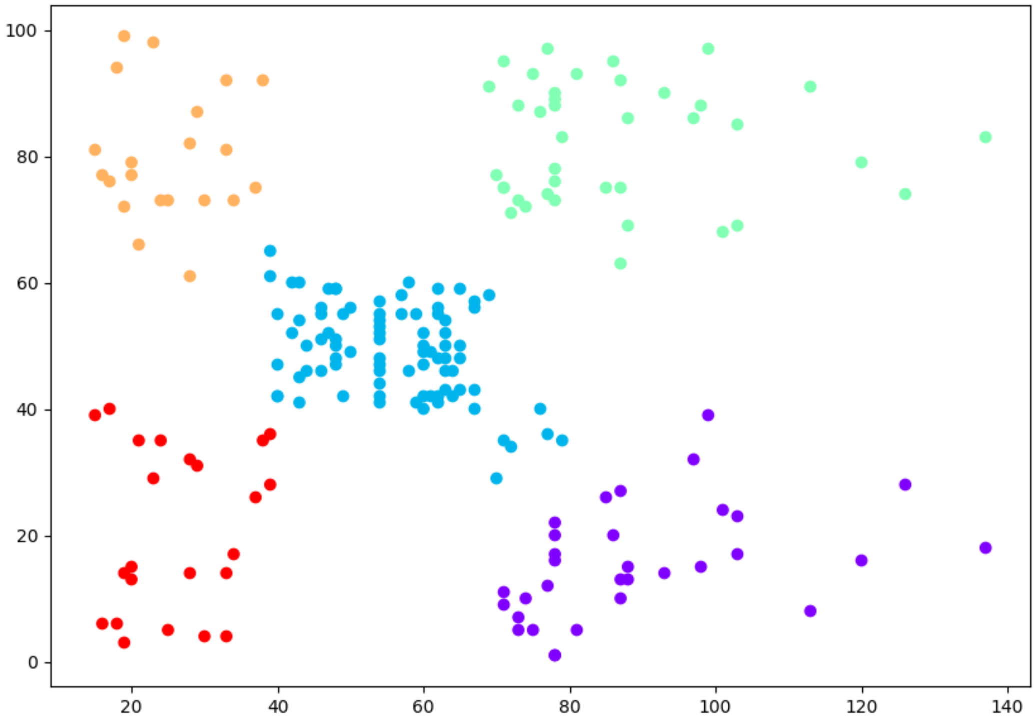
Code:

```
from operator import methodcaller
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering

print("Un-supervised Learning Model")

customer_data = pd.read_csv("D:\\MSc IT\\PART 2\\SEM 3\\AAI\\AAI
Practicals\\prac7\\Mall_Customers.csv")
customer_data.shape
customer_data.head()
data = customer_data.iloc[:, 3:5].values

plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")

dend = shc.dendrogram(shc.linkage(data, method='ward'))

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='ward')
cluster.fit_predict(data)

plt.figure(figsize=(10, 7))
plt.scatter(data[:, 0], data[:,1], c = cluster.labels_, cmap = 'rainbow')
plt.show()
```

Output:



Customer Dendograms
400
350
300
250
200
150
100
50
0



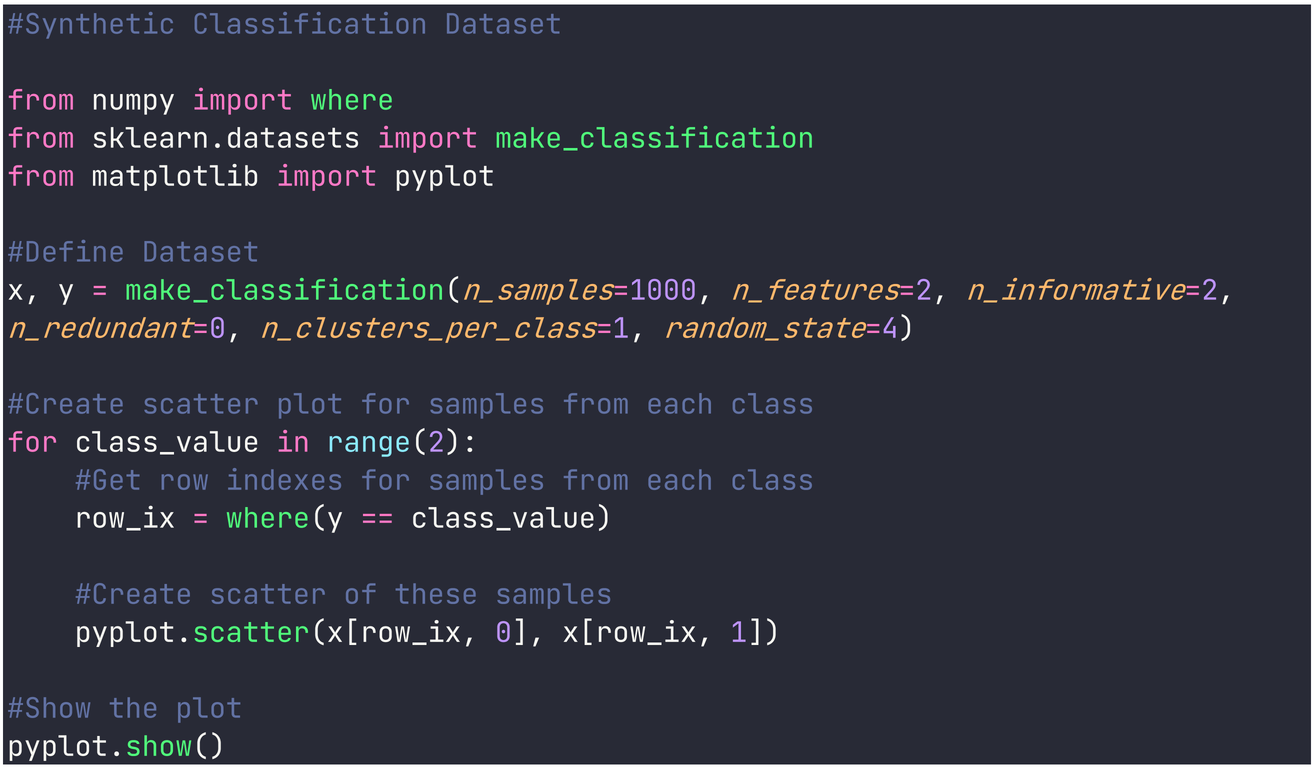
100
80
60
40
20
0
20
40
60
80
100
120
140

Practical 07

Aim: Write an application to implement Clustering algorithm.

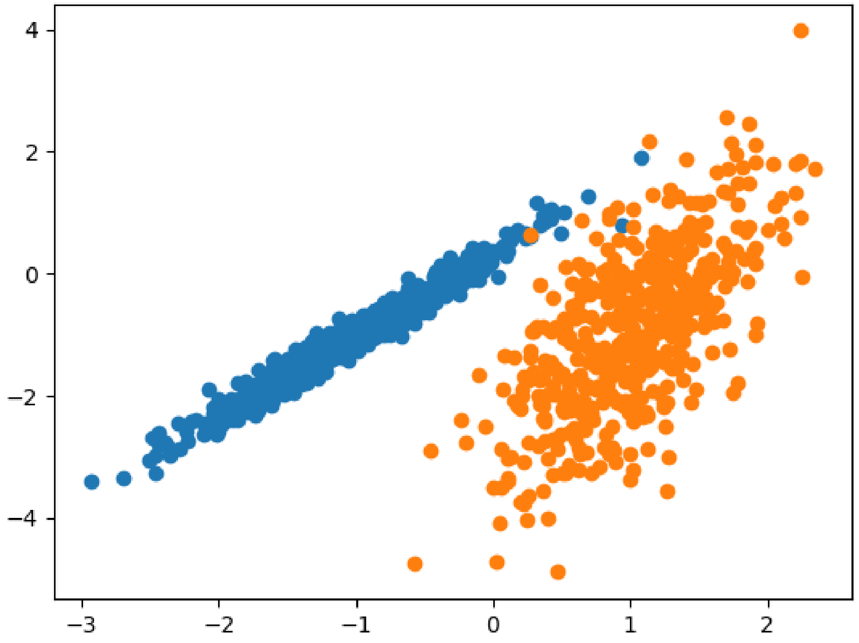
Description:

Code:



```
#Synthetic Classification Dataset

from numpy import where
from sklearn.datasets import make_classification
from matplotlib import pyplot

#Define Dataset
x, y = make_classification(n_samples=1000, n_features=2, n_informative=2,
n_redundant=0, n_clusters_per_class=1, random_state=4)

#Create scatter plot for samples from each class
for class_value in range(2):
    #Get row indexes for samples from each class
    row_ix = where(y == class_value)

    #Create scatter of these samples
    pyplot.scatter(x[row_ix, 0], x[row_ix, 1])

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

Output:



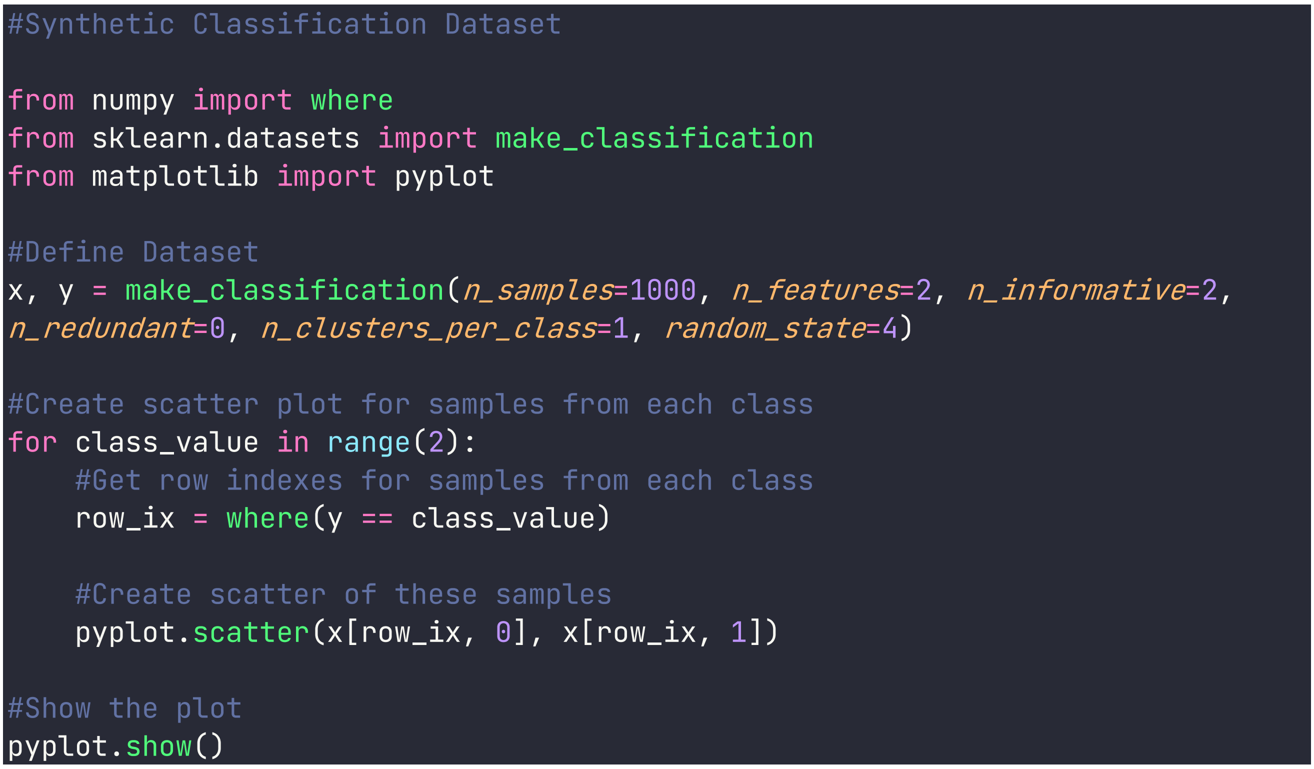
Practical 08

Aim: Write an application to implement BFS and DFS algorithm.

A) BFS Algorithm

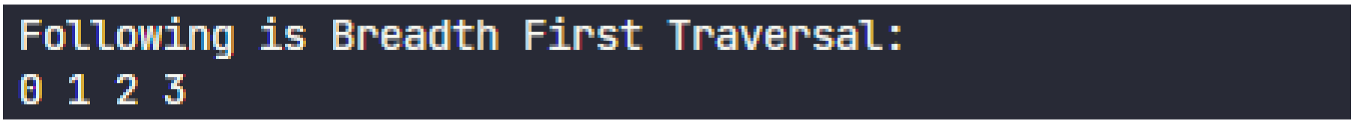
Description:

Code:



```python
#Synthetic Classification Dataset

from numpy import where
from sklearn.datasets import make_classification
from matplotlib import pyplot

#Define Dataset
x, y = make_classification(n_samples=1000, n_features=2, n_informative=2,
n_redundant=0, n_clusters_per_class=1, random_state=4)

#Create scatter plot for samples from each class
for class_value in range(2):
    #Get row indexes for samples from each class
    row_ix = where(y == class_value)

    #Create scatter of these samples
    pyplot.scatter(x[row_ix, 0], x[row_ix, 1])

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

Output:

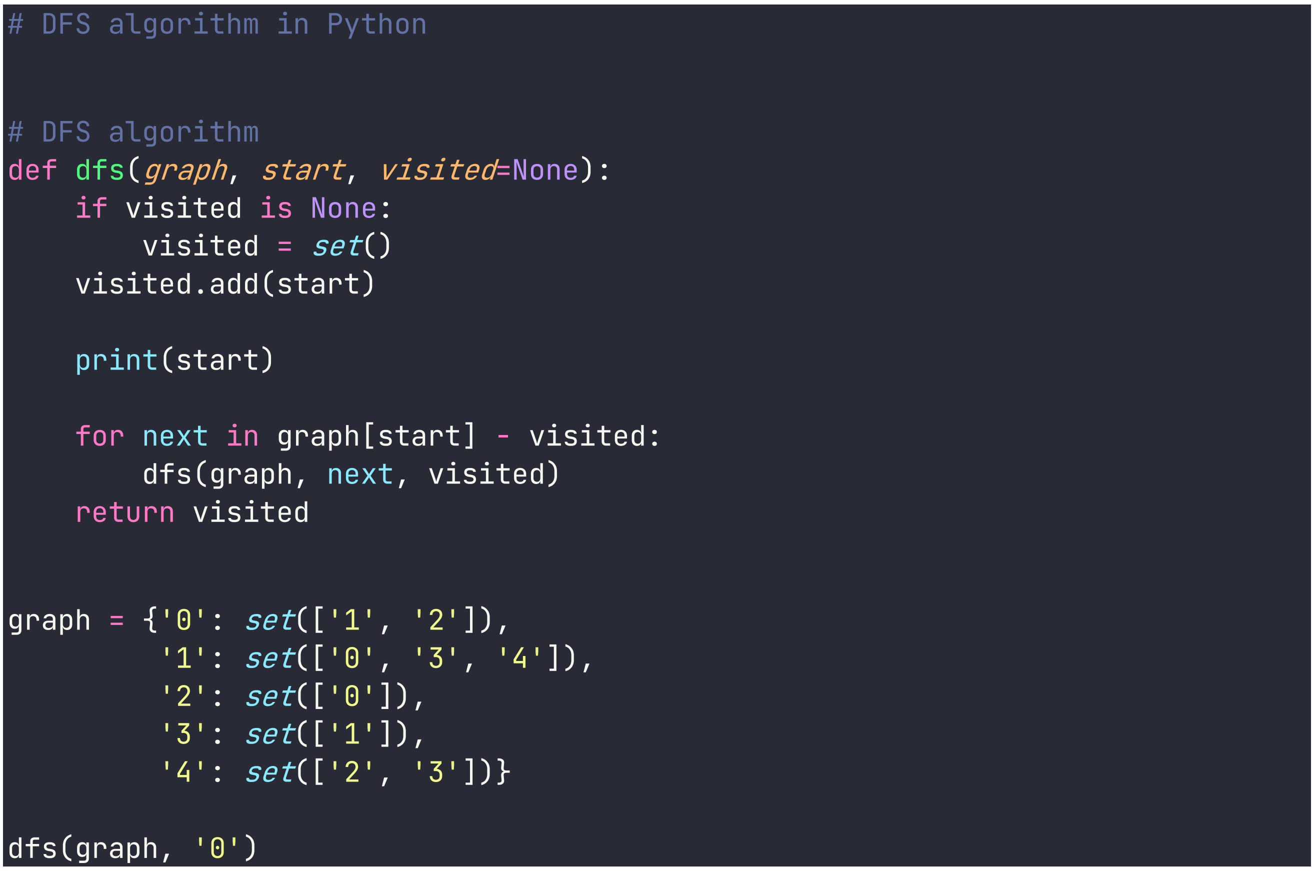


```
Following is Breadth First Traversal:
0 1 2 3
```

B) DFS Algorithm

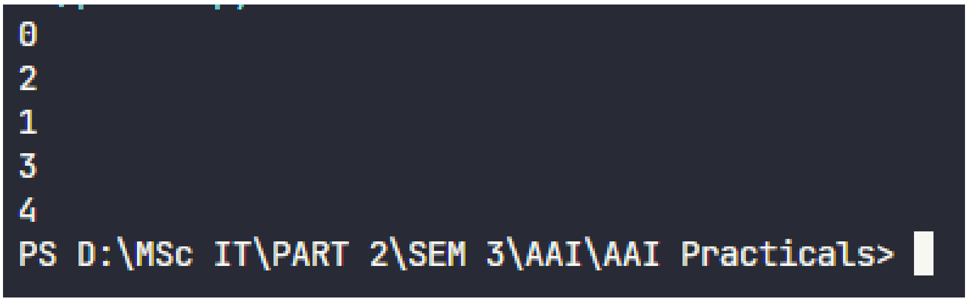
Description:

Code:



```python
# DFS algorithm in Python


# DFS algorithm
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': set(['1', '2']),
         '1': set(['0', '3', '4']),
         '2': set(['0']),
         '3': set(['1']),
         '4': set(['2', '3'])}

dfs(graph, '0')
```

Output:



```
0
2
1
3
4
PS D:\MSc IT\PART 2\SEM 3\AAI\AAI Practicals>
```