VALIA C.I. COLLEGE OF COMMERCE & VALIA LC COLLEGE OF ARTS
CES ROAD D.N NAGAR

(Affiliated to University Of Mumbai)

Mumbai-Maharashtra-400053

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Journal entitled **APPLIED ARTIFICIAL INTELLIGENCE** is bonafied work of **GOVIND SAINI** bearing Roll No: **07** submitted in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY** from University of Mumbai.

Date: **Internal Guide**

INDEX

Practical : 1

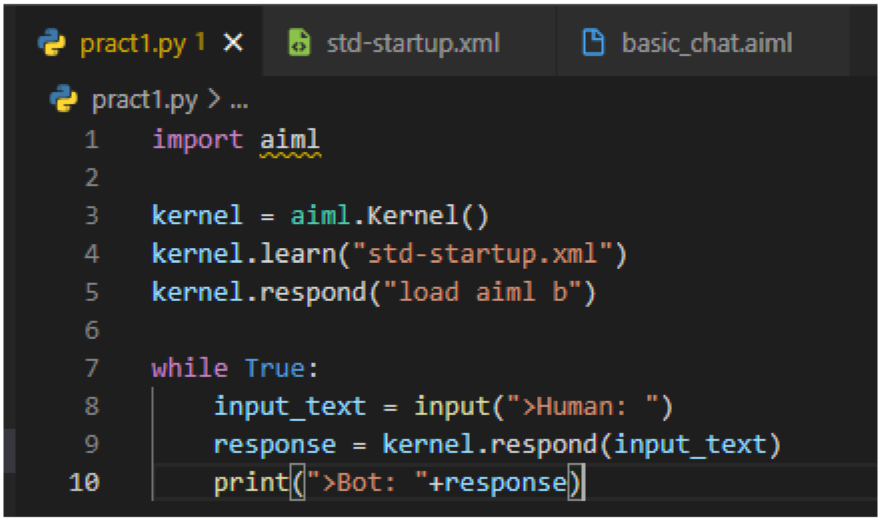
Aim : Design an Expert system using AIML.

Description :

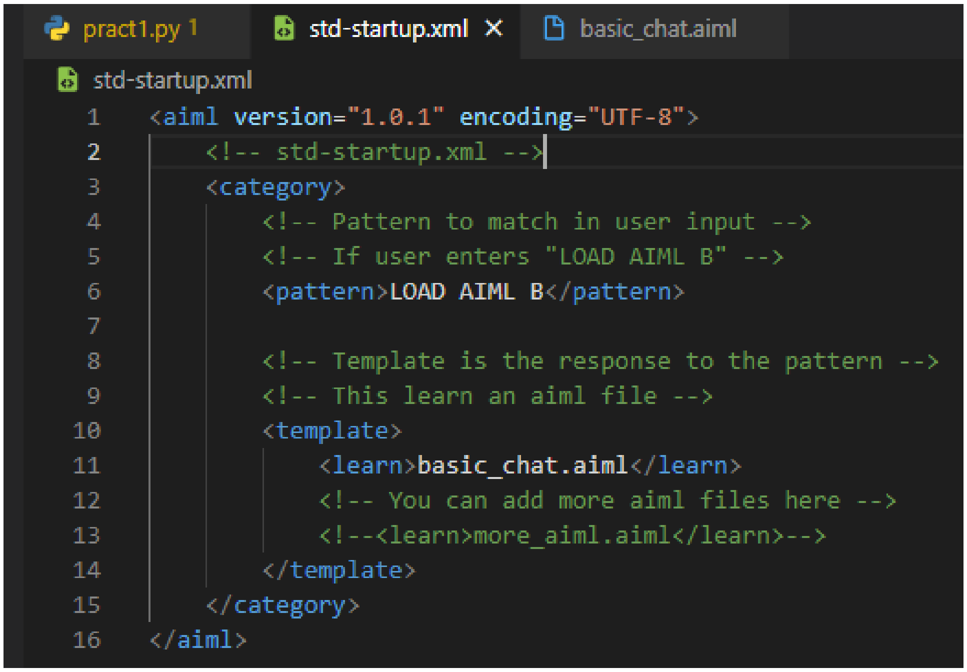
What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

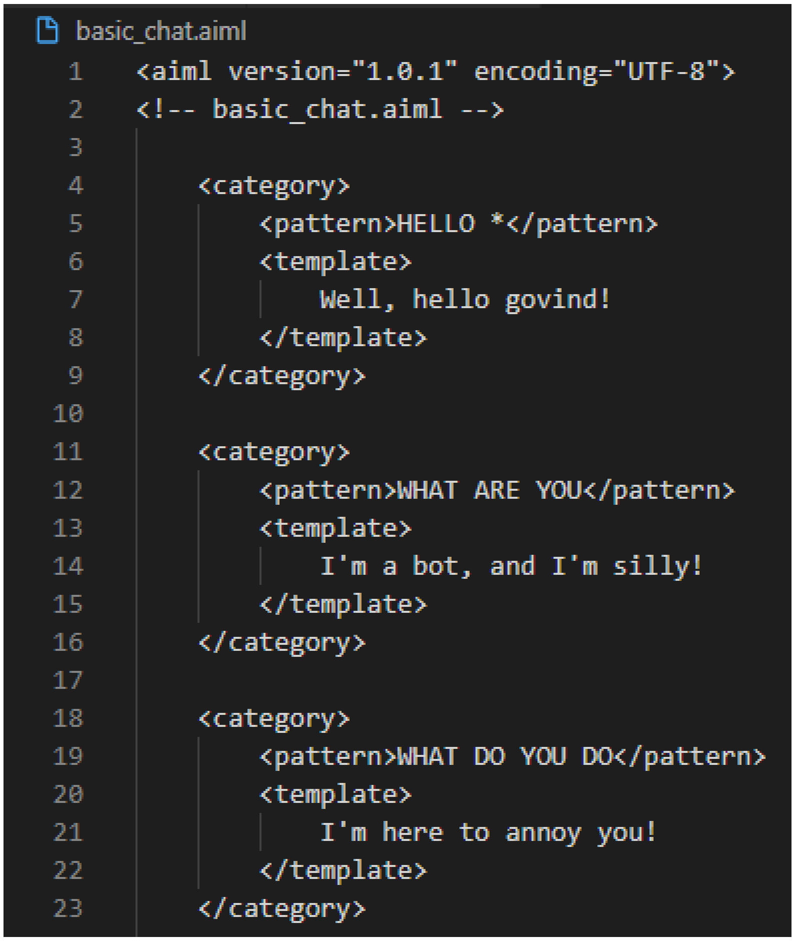
Code :



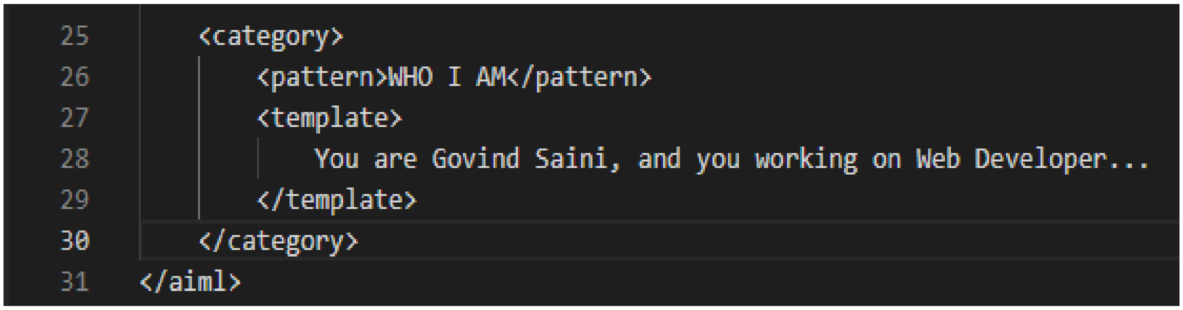
```python
import aiml

kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")

while True:
    input_text = input(">Human: ")
    response = kernel.respond(input_text)
    print(">Bot: "+response)
```



```
<aiml version="1.0.1" encoding="UTF-8">
    <!-- std-startup.xml -->
    <category>
        <!-- Pattern to match in user input -->
        <!-- If user enters "LOAD AIML B" -->
        <pattern>LOAD AIML B</pattern>

        <!-- Template is the response to the pattern -->
        <!-- This learn an aiml file -->
        <template>
            <learn>basic_chat.aiml</learn>
            <!-- You can add more aiml files here -->
            <!--<learn>more_aiml.aiml</learn>-->
        </template>
    </category>
</aiml>
```

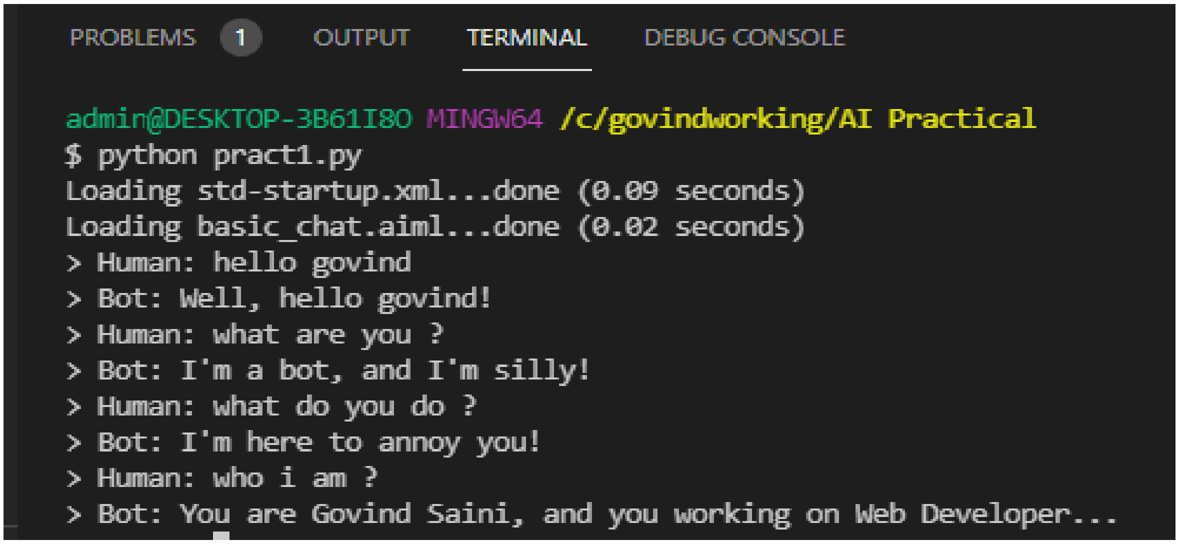


```
<aiml version="1.0.1" encoding="UTF-8">
<!-- basic_chat.aiml -->

    <category>
        <pattern>HELLO *</pattern>
        <template>
            Well, hello govind!
        </template>
    </category>

    <category>
        <pattern>WHAT ARE YOU</pattern>
        <template>
            I'm a bot, and I'm silly!
        </template>
    </category>

    <category>
        <pattern>WHAT DO YOU DO</pattern>
        <template>
            I'm here to annoy you!
        </template>
    </category>
```



```
    <category>
        <pattern>WHO I AM</pattern>
        <template>
            You are Govind Saini, and you working on Web Developer...
        </template>
    </category>
</aiml>
```

Ouput :



```
admin@DESKTOP-3B61I8O MINGW64 /c/govindworking/AI Practical
$ python pract1.py
Loading std-startup.xml...done (0.09 seconds)
Loading basic_chat.aiml...done (0.02 seconds)
> Human: hello govind
> Bot: Well, hello govind!
> Human: what are you ?
> Bot: I'm a bot, and I'm silly!
> Human: what do you do ?
> Bot: I'm here to annoy you!
> Human: who i am ?
> Bot: You are Govind Saini, and you working on Web Developer...
```

Practical : 2

Aim : Design a bot using AIML.

Description :

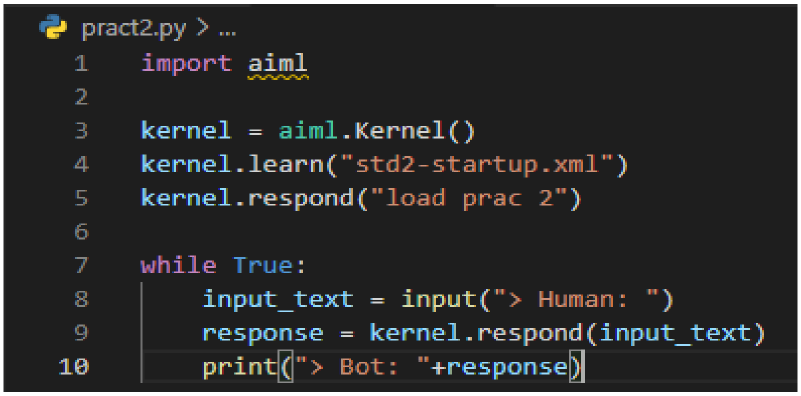
What is AIML?

AIML stands for Artificial Intelligence Modelling Language. AIML is an XML based markup language meant to create artificial intelligent applications. AIML makes it possible to create human interfaces while keeping the implementation simple to program, easy to understand and highly maintainable. This tutorial will teach you the basics of AIML. All the basic components of AIML with suitable examples have been discussed in this tutorial.

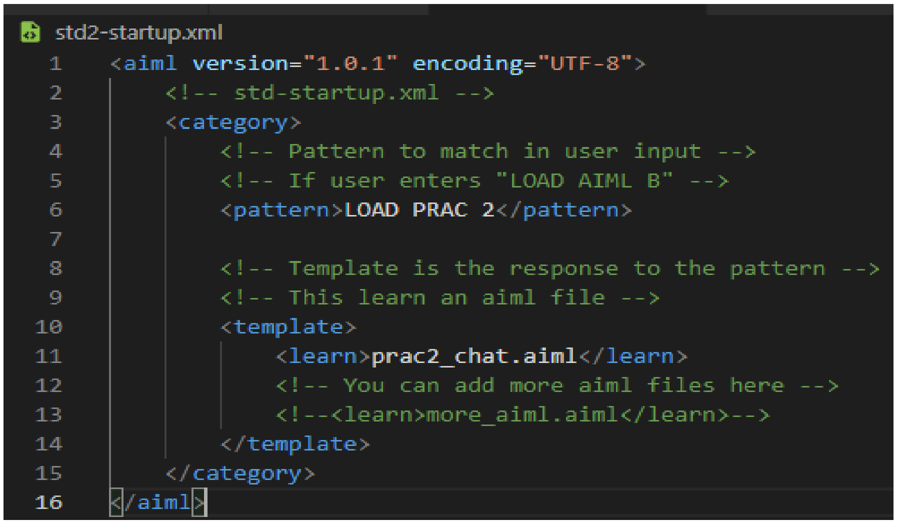
AIML Tags/Description

- <aiml> − defines the beginning and end of a AIML document.
- <category> − defines the unit of knowledge in bot’s knowledge base.
- <pattern> − defines the pattern to match what a user may input to an bot.
- <template> − defines the response of a bot to user's input.

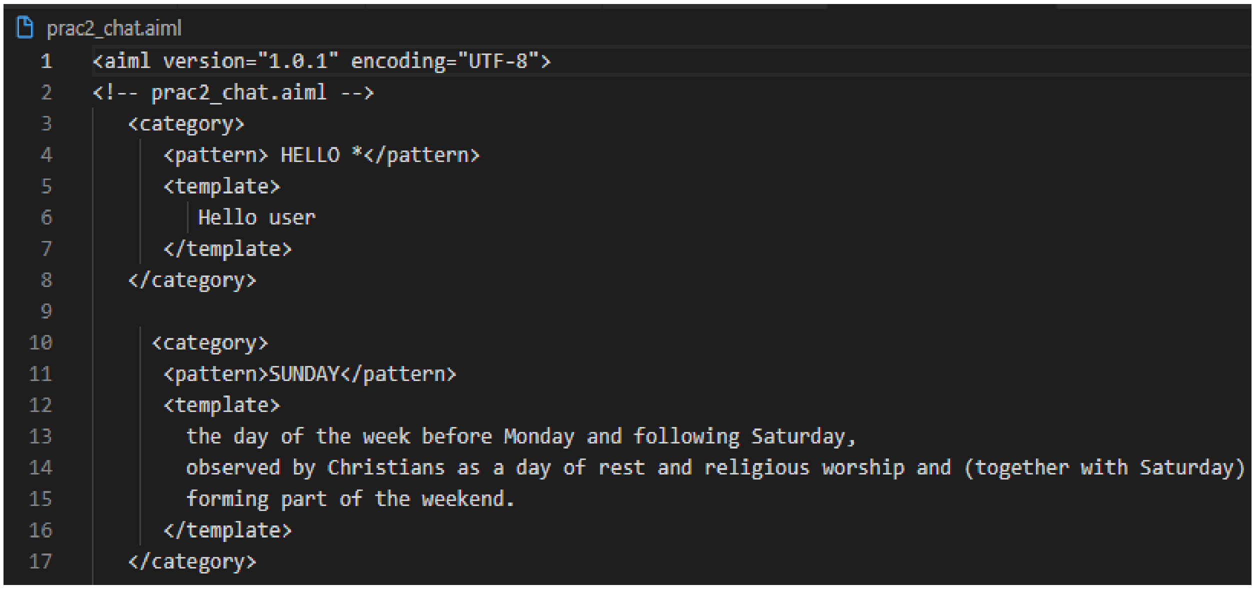
Code :



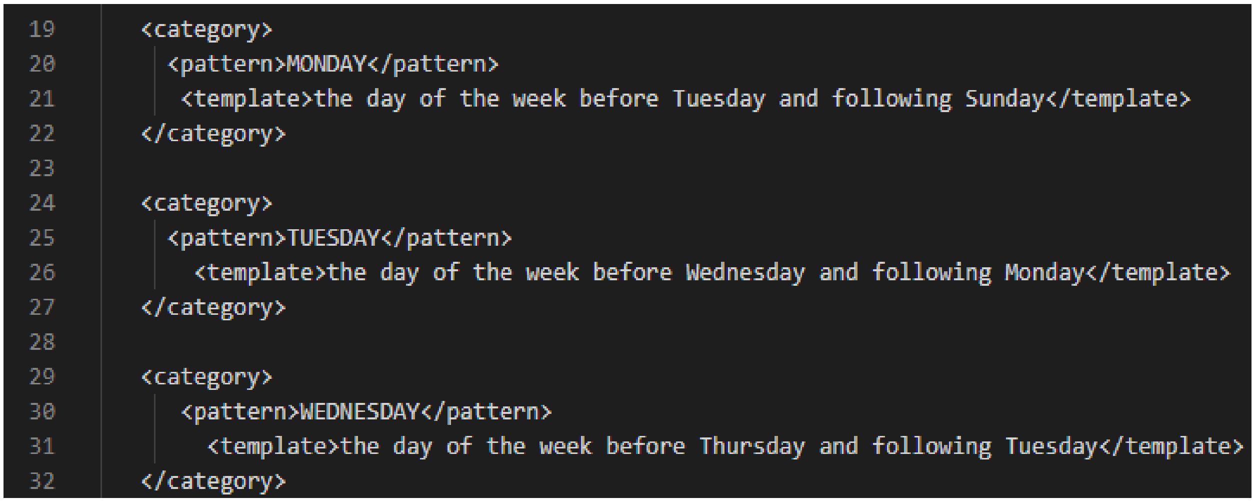
```python
import aiml

kernel = aiml.Kernel()
kernel.learn("std2-startup.xml")
kernel.respond("load prac 2")

while True:
    input_text = input("> Human: ")
    response = kernel.respond(input_text)
    print("> Bot: "+response)
```



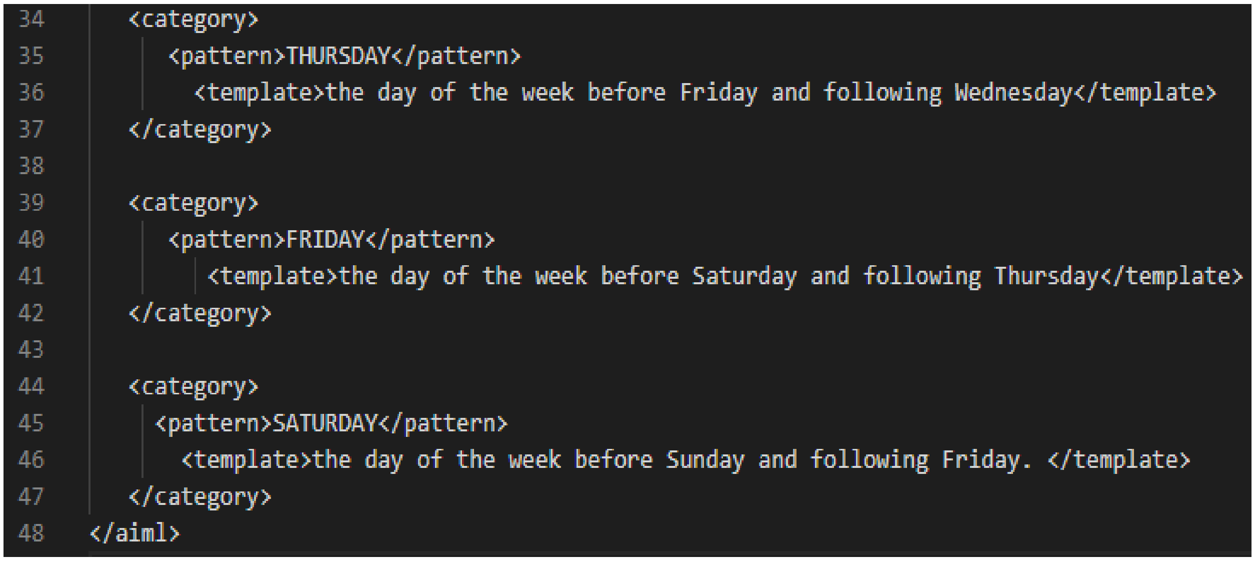
```
<aiml version="1.0.1" encoding="UTF-8">
    <!-- std-startup.xml -->
    <category>
        <!-- Pattern to match in user input -->
        <!-- If user enters "LOAD AIML B" -->
        <pattern>LOAD PRAC 2</pattern>

        <!-- Template is the response to the pattern -->
        <!-- This learn an aiml file -->
        <template>
            <learn>prac2_chat.aiml</learn>
            <!-- You can add more aiml files here -->
            <!--<learn>more_aiml.aiml</learn>-->
        </template>
    </category>
</aiml>
```



```
<aiml version="1.0.1" encoding="UTF-8">
<!-- prac2_chat.aiml -->
   <category>
      <pattern> HELLO *</pattern>
      <template>
         Hello user
      </template>
   </category>

     <category>
      <pattern>SUNDAY</pattern>
      <template>
        the day of the week before Monday and following Saturday,
        observed by Christians as a day of rest and religious worship and (together with Saturday)
        forming part of the weekend.
      </template>
   </category>
```

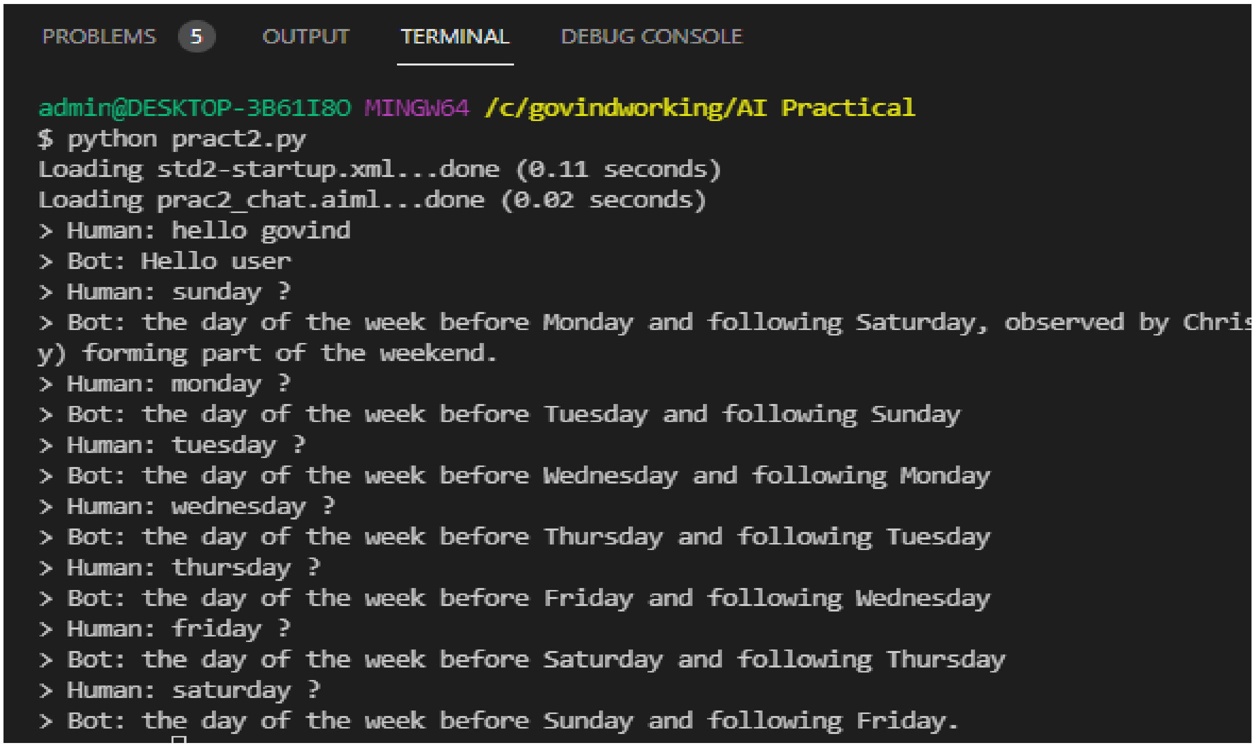


```
    <category>
      <pattern>MONDAY</pattern>
       <template>the day of the week before Tuesday and following Sunday</template>
    </category>

    <category>
      <pattern>TUESDAY</pattern>
        <template>the day of the week before Wednesday and following Monday</template>
    </category>

    <category>
       <pattern>WEDNESDAY</pattern>
         <template>the day of the week before Thursday and following Tuesday</template>
    </category>
```



```
    <category>
        <pattern>THURSDAY</pattern>
          <template>the day of the week before Friday and following Wednesday</template>
    </category>

    <category>
        <pattern>FRIDAY</pattern>
           <template>the day of the week before Saturday and following Thursday</template>
    </category>

    <category>
       <pattern>SATURDAY</pattern>
         <template>the day of the week before Sunday and following Friday. </template>
    </category>
</aiml>
```

Ouput :



```
admin@DESKTOP-3B61I8O MINGW64 /c/govindworking/AI Practical
$ python pract2.py
Loading std2-startup.xml...done (0.11 seconds)
Loading prac2_chat.aiml...done (0.02 seconds)
> Human: hello govind
> Bot: Hello user
> Human: sunday ?
> Bot: the day of the week before Monday and following Saturday, observed by Chris
y) forming part of the weekend.
> Human: monday ?
> Bot: the day of the week before Tuesday and following Sunday
> Human: tuesday ?
> Bot: the day of the week before Wednesday and following Monday
> Human: wednesday ?
> Bot: the day of the week before Thursday and following Tuesday
> Human: thursday ?
> Bot: the day of the week before Friday and following Wednesday
> Human: friday ?
> Bot: the day of the week before Saturday and following Thursday
> Human: saturday ?
> Bot: the day of the week before Sunday and following Friday.
```

Practical : 3

Aim : Implement Bayes Theorem using Python.

Description :

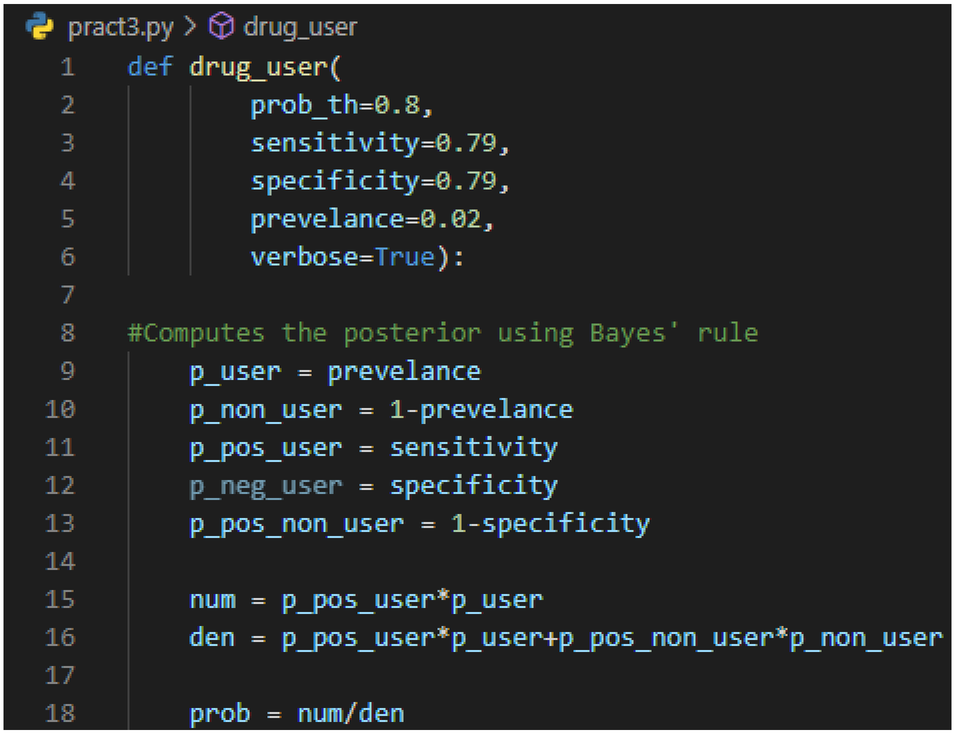
Bayes’ Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes’ Theorem is stated as:

P(class|data) = (P(data|class) * P(class)) / P(data)

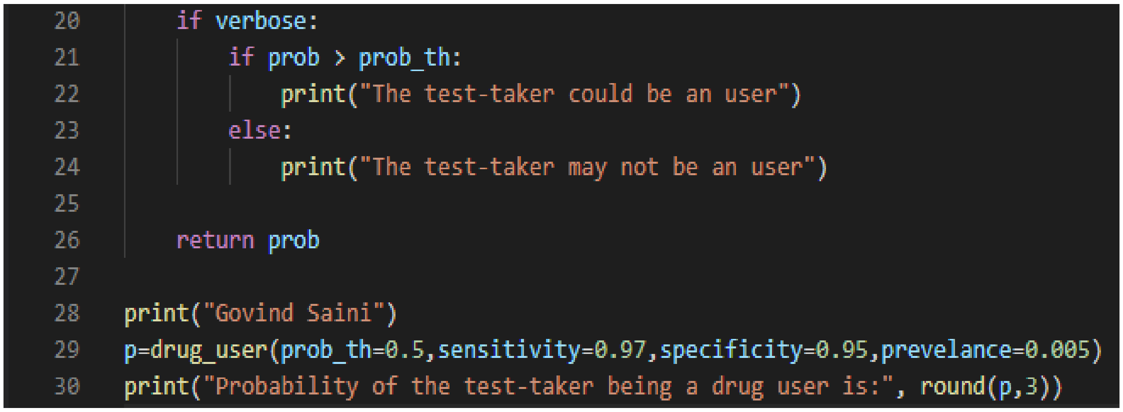
Where P(class|data) is the probability of class given the provided data.

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. It is called Naive Bayes or idiot Bayes because the calculations of the probabilities for each class are simplified to make their calculations tractable.

Code :

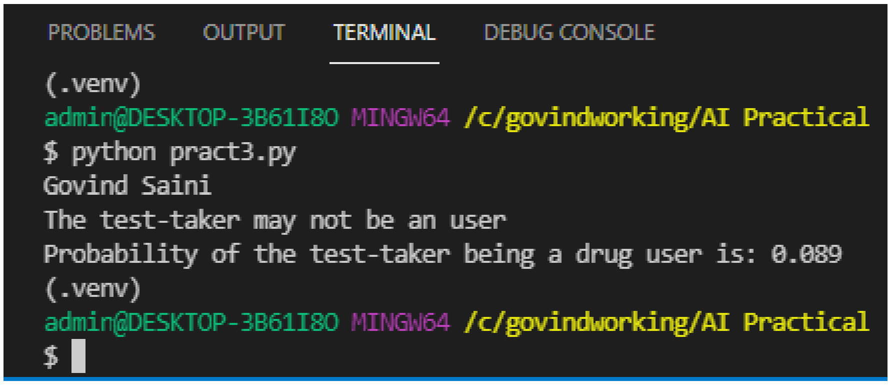


```python
def drug_user(
        prob_th=0.8,
        sensitivity=0.79,
        specificity=0.79,
        prevelance=0.02,
        verbose=True):

#Computes the posterior using Bayes' 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
```



```python
    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("Govind Saini")
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))
```

Ouput :



```
(.venv)
admin@DESKTOP-3B61I8O MINGW64 /c/govindworking/AI Practical
$ python pract3.py
Govind Saini
The test-taker may not be an user
Probability of the test-taker being a drug user is: 0.089
(.venv)
admin@DESKTOP-3B61I8O MINGW64 /c/govindworking/AI Practical
$
```

Practical : 4

Aim : Implement Conditional Probability and joint probability using Python.

Description :

What is Conditional Probability?

The probability of one event given the occurrence of another event is called the conditional probability. The conditional probability of one to one or more random variables is referred to as the conditional probability distribution.

For example, the conditional probability of event A given event B is written formally as:

- **P(A given B)**

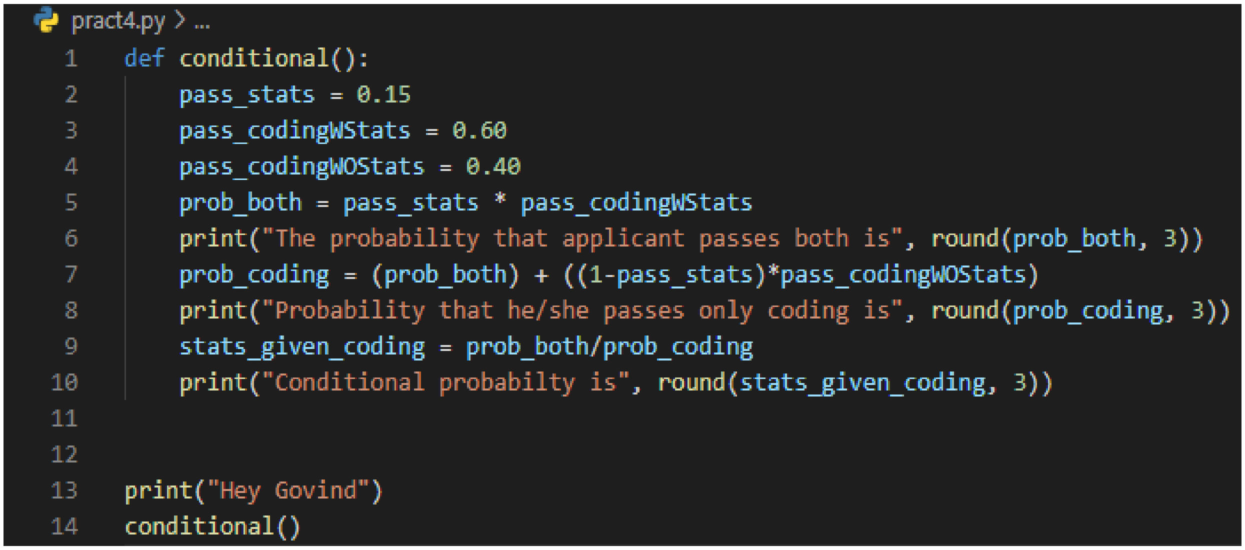
The “given” is denoted using the pipe “|” operator; for example:

- **P(A | B)**

The conditional probability for events A given event B is calculated as follows:

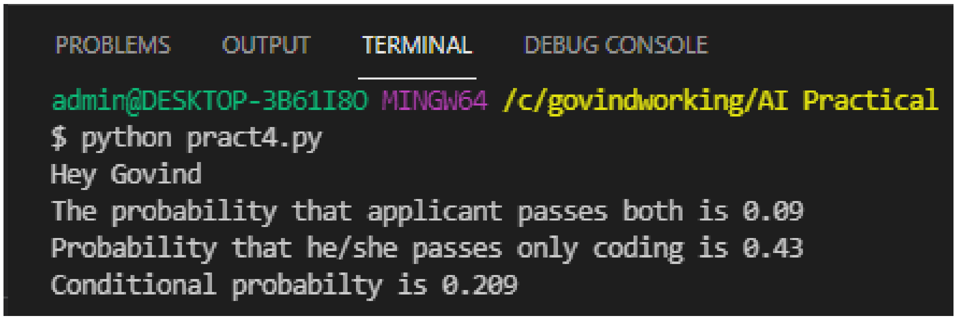
- **P(A given B) = P(A and B) / P(B)**

Code :



```python
def conditional():
    pass_stats = 0.15
    pass_codingWStats = 0.60
    pass_codingWOStats = 0.40
    prob_both = pass_stats * pass_codingWStats
    print("The probability that applicant passes both is", round(prob_both, 3))
    prob_coding = (prob_both) + ((1-pass_stats)*pass_codingWOStats)
    print("Probability that he/she passes only coding is", round(prob_coding, 3))
    stats_given_coding = prob_both/prob_coding
    print("Conditional probabilty is", round(stats_given_coding, 3))


print("Hey Govind")
conditional()
```

Ouput :



```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ python pract4.py
Hey Govind
The probability that applicant passes both is 0.09
Probability that he/she passes only coding is 0.43
Conditional probabilty is 0.209
```

Description :

What is Joint Probability?

The probability of two (or more) events is called the joint probability. The joint probability of two or more random variables is referred to as the joint probability distribution.The joint probability for events A and B is calculated as the probability of event A given event B multiplied by the probability of event B.

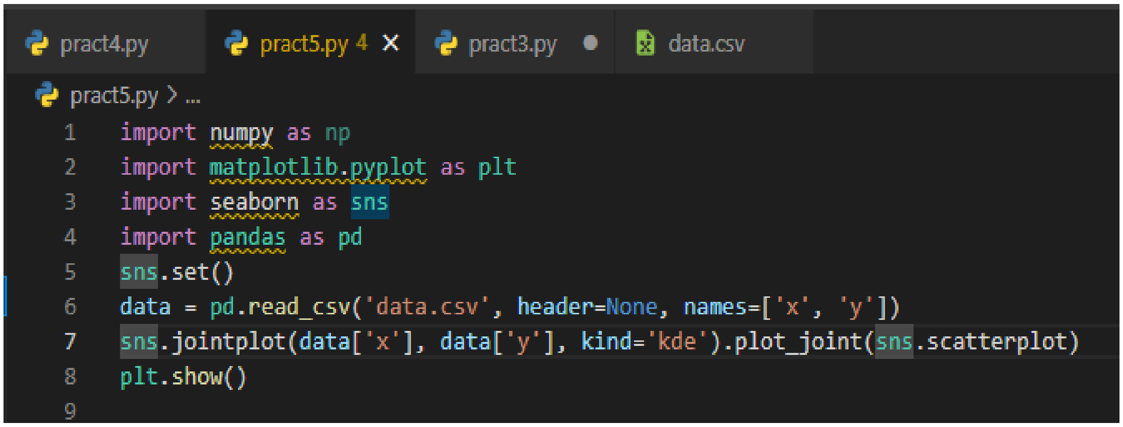
This can be stated formally as follows:

P(A and B) = P(A given B) * P(B)

The calculation of the joint probability is sometimes called the fundamental rule of probability or the “product rule” of probability or the “chain rule” of probability

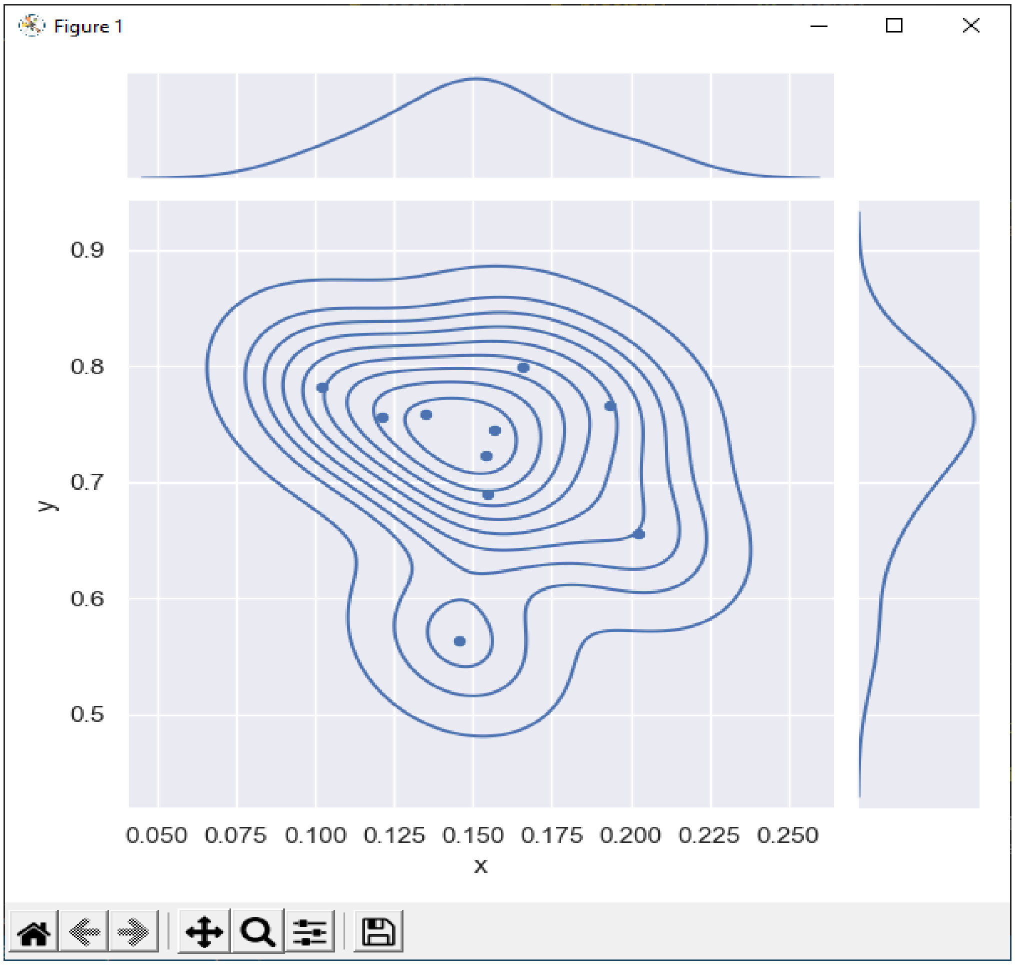
P(A and B) = P(A given B) * P(B) = P(B given A) * P(A)

Code :



```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
sns.set()
data = pd.read_csv('data.csv', header=None, names=['x', 'y'])
sns.jointplot(data['x'], data['y'], kind='kde').plot_joint(sns.scatterplot)
plt.show()
```

Output :



Practical : 5

Aim : A program to implement Rule Based System.

Description :

What is Rule Based System?

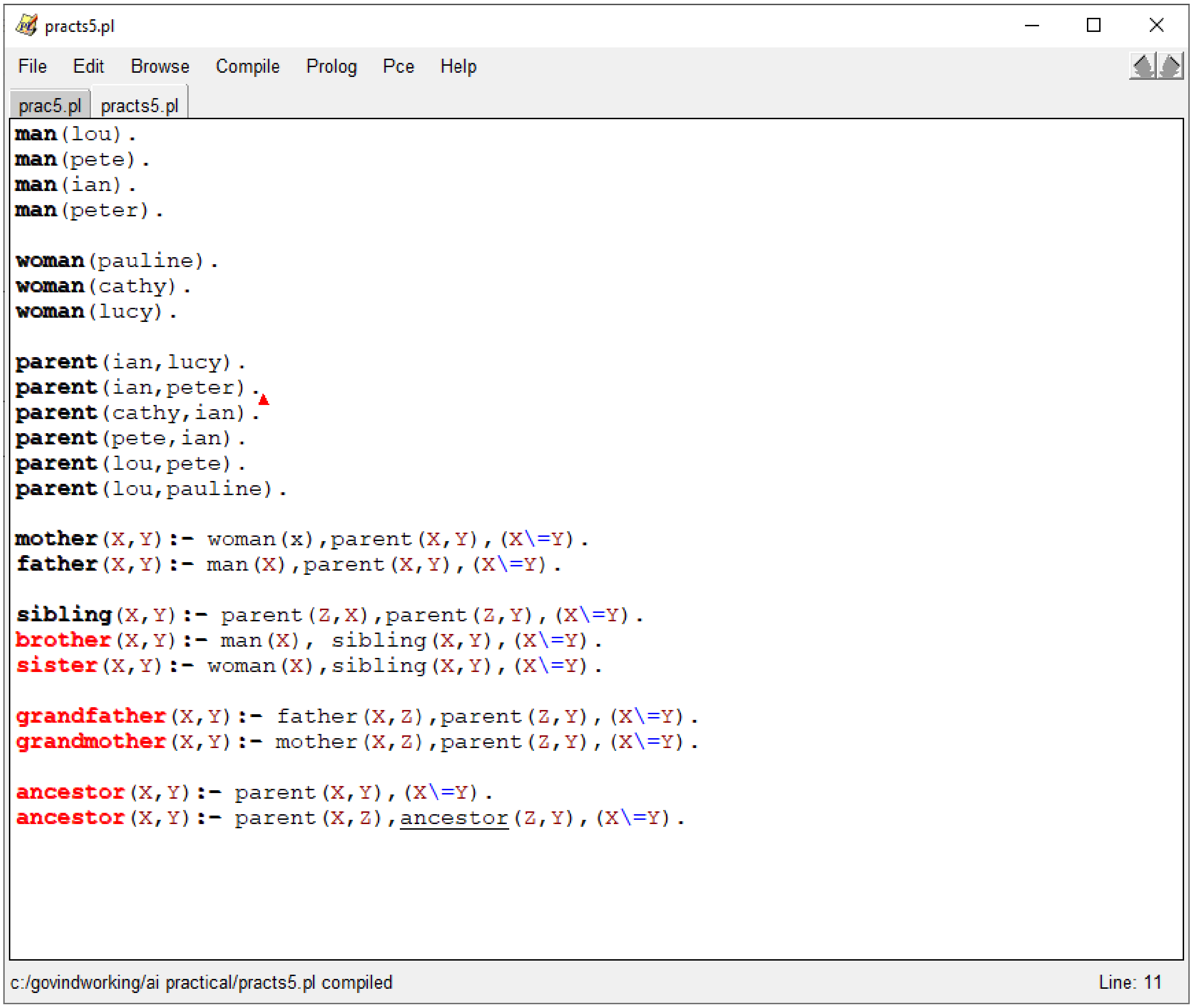
A rule-based system is a system that applies human-made rules to store, sort and manipulate data. In doing so, it mimics human intelligence.

To work, rule-based systems require a set of facts or source of data, and a set of rules for manipulating that data. These rules are sometimes referred to as ‘If statements’ as they tend to follow the line of ‘IF X happens THEN do Y’.

Automation software like Think Automation is a good example. It automates processes by breaking them down into steps.

- First comes the data or new business event
- Then comes the analysis: the part where the system conditonally processes the data against its rules
- Then comes any subsequent automated follow-up actions

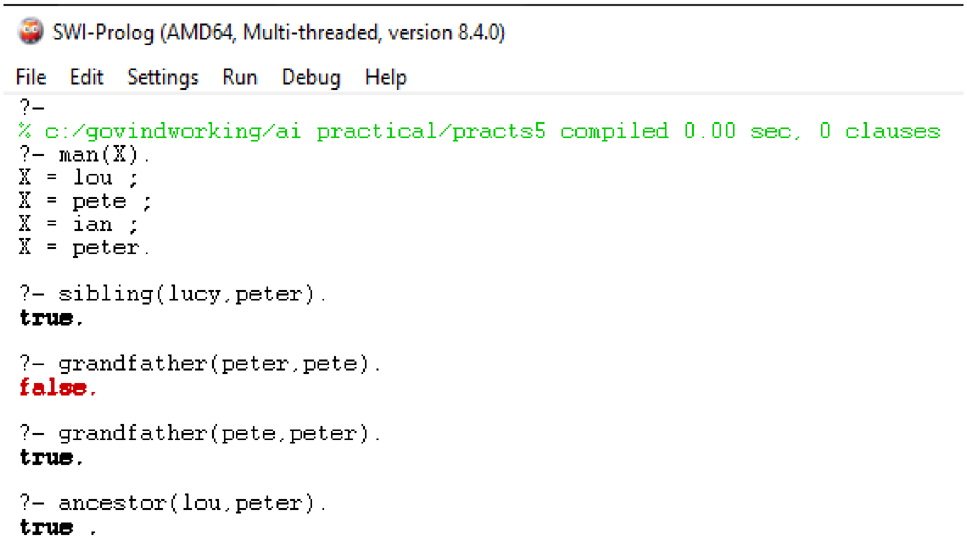
Code :



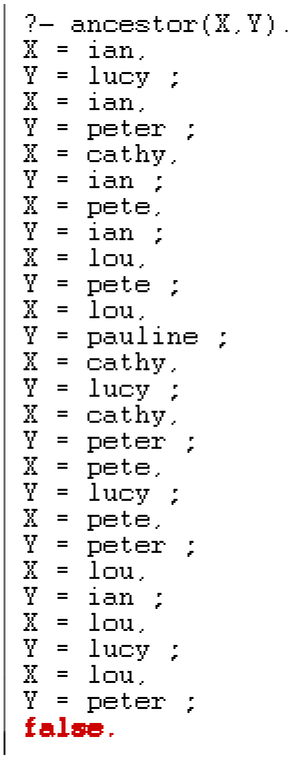
```
man(lou).
man(pete).
man(ian).
man(peter).

woman(pauline).
woman(cathy).
woman(lucy).

parent(ian,lucy).
parent(ian,peter).
parent(cathy,ian).
parent(pete,ian).
parent(lou,pete).
parent(lou,pauline).

mother(X,Y):- woman(x),parent(X,Y),(X\=Y).
father(X,Y):- man(X),parent(X,Y),(X\=Y).

sibling(X,Y):- parent(Z,X),parent(Z,Y),(X\=Y).
brother(X,Y):- man(X), sibling(X,Y),(X\=Y).
sister(X,Y):- woman(X),sibling(X,Y),(X\=Y).

grandfather(X,Y):- father(X,Z),parent(Z,Y),(X\=Y).
grandmother(X,Y):- mother(X,Z),parent(Z,Y),(X\=Y).

ancestor(X,Y):- parent(X,Y),(X\=Y).
ancestor(X,Y):- parent(X,Z),ancestor(Z,Y),(X\=Y).
```

c:/govindworking/ai practical/practs5.pl compiled Line: 11

Ouput :


SWI-Prolog (AMD64, Multi-threaded, version 8.4.0)
File Edit Settings Run Debug Help

```
?-
% c:/govindworking/ai practical/practs5 compiled 0.00 sec, 0 clauses
?- man(X).
X = lou ;
X = pete ;
X = ian ;
X = peter.

?- sibling(lucy,peter).
true.

?- grandfather(peter,pete).
false.

?- grandfather(pete,peter).
true.

?- ancestor(lou,peter).
true .
```



```
?- ancestor(X,Y).
X = ian,
Y = lucy ;
X = ian,
Y = peter ;
X = cathy,
Y = ian ;
X = pete,
Y = ian ;
X = lou,
Y = pete ;
X = lou,
Y = pauline ;
X = cathy,
Y = lucy ;
X = cathy,
Y = peter ;
X = pete,
Y = lucy ;
X = pete,
Y = peter ;
X = lou,
Y = ian ;
X = lou,
Y = lucy ;
X = lou,
Y = peter ;
false.
```

Practical : 6

Aim : Design a Fuzzy based application using Python / R

Description :

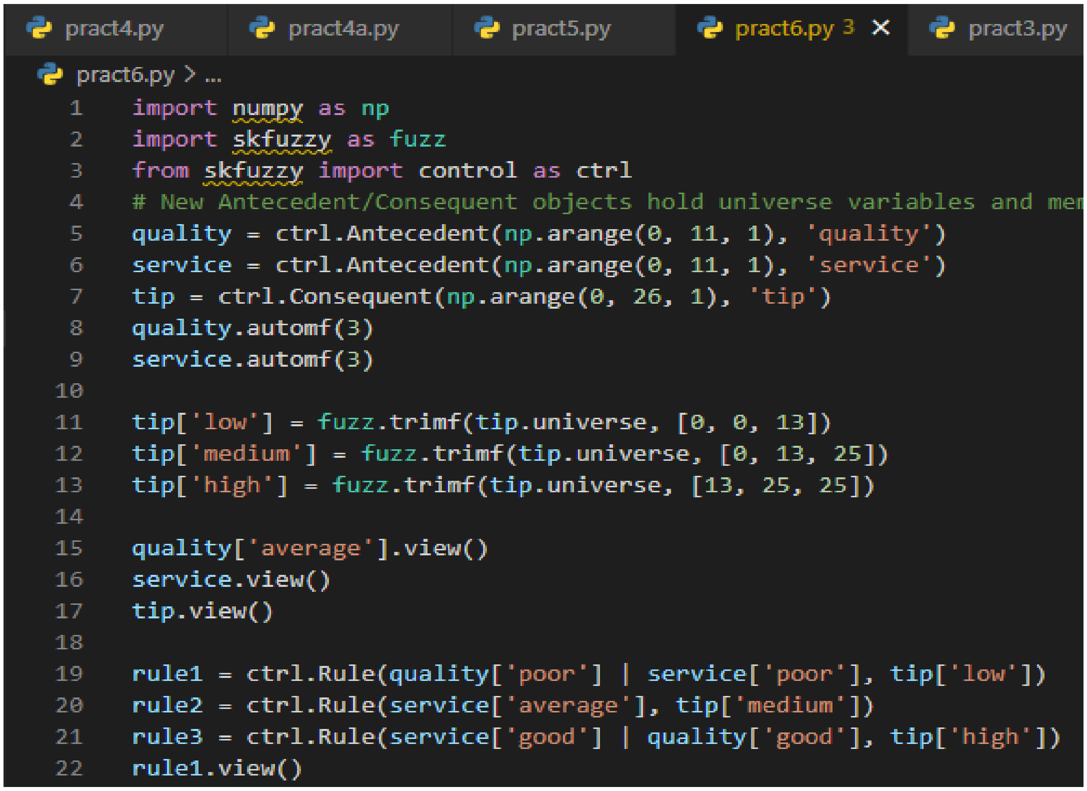
What is Fuzzy based application?

Fuzzy sets were introduced by Lotfi Zadeh (1921–2017) in 1965.

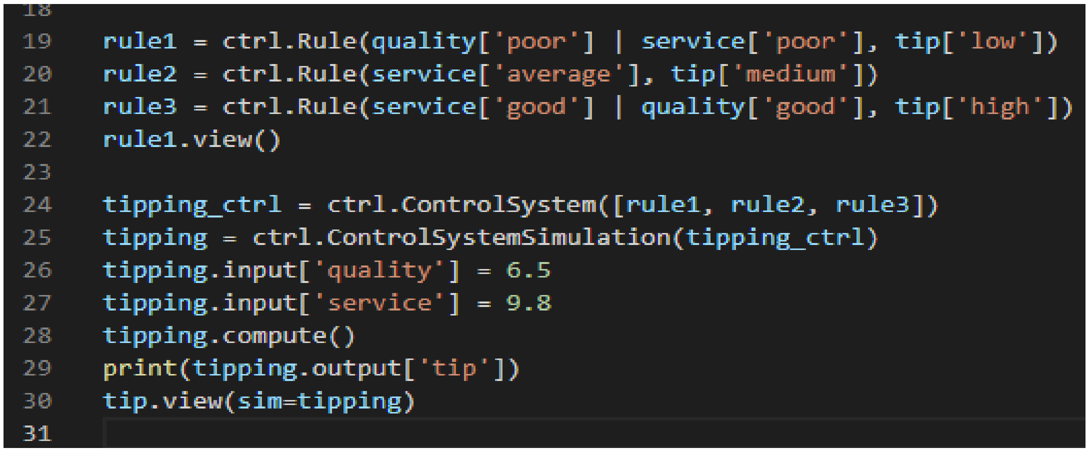
Unlike crisp sets, a fuzzy set allows partial belonging to a set, that is defined by a degree of membership, denoted by µ, that can take any value from 0 (element does not belong at all in the set) to 1 (element belongs fully to the set).

It is evident that if we remove all the values of belonging except from 0 and 1, the fuzzy set will collapse to a crisp set that was described in the previous section.

Code :

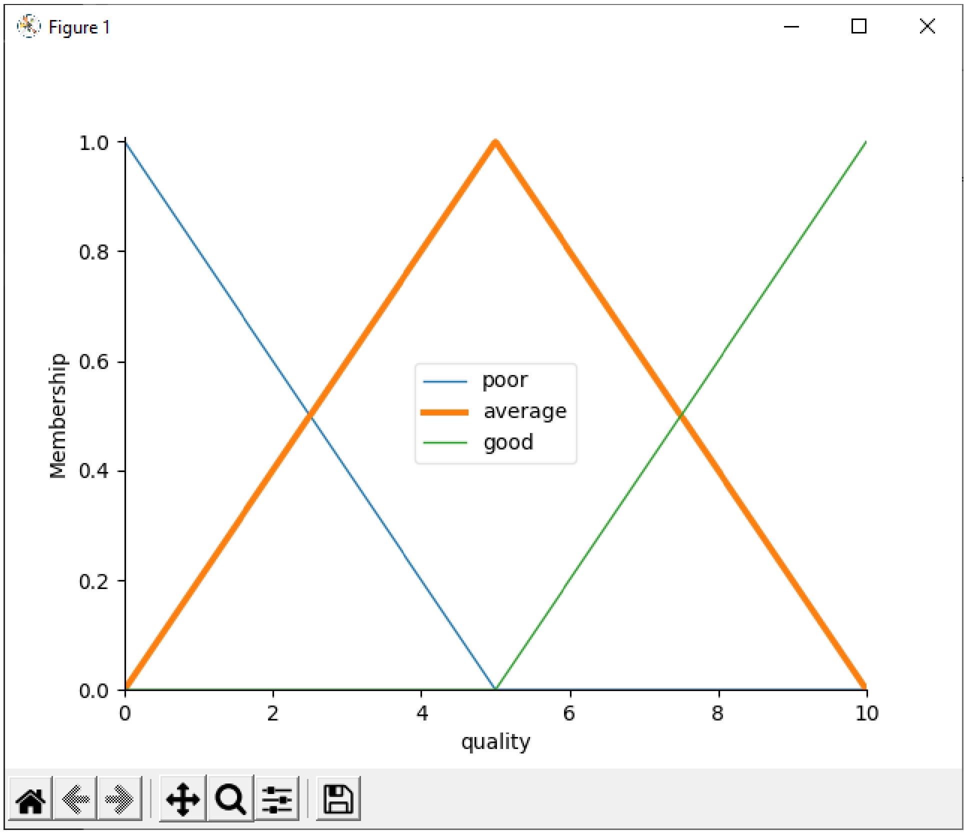


```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
# New Antecedent/Consequent objects hold universe variables and mem
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
quality.automf(3)
service.automf(3)

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

quality['average'].view()
service.view()
tip.view()

rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])
rule1.view()
```



```
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])
rule1.view()

tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8
tipping.compute()
print(tipping.output['tip'])
tip.view(sim=tipping)
```

Ouput :



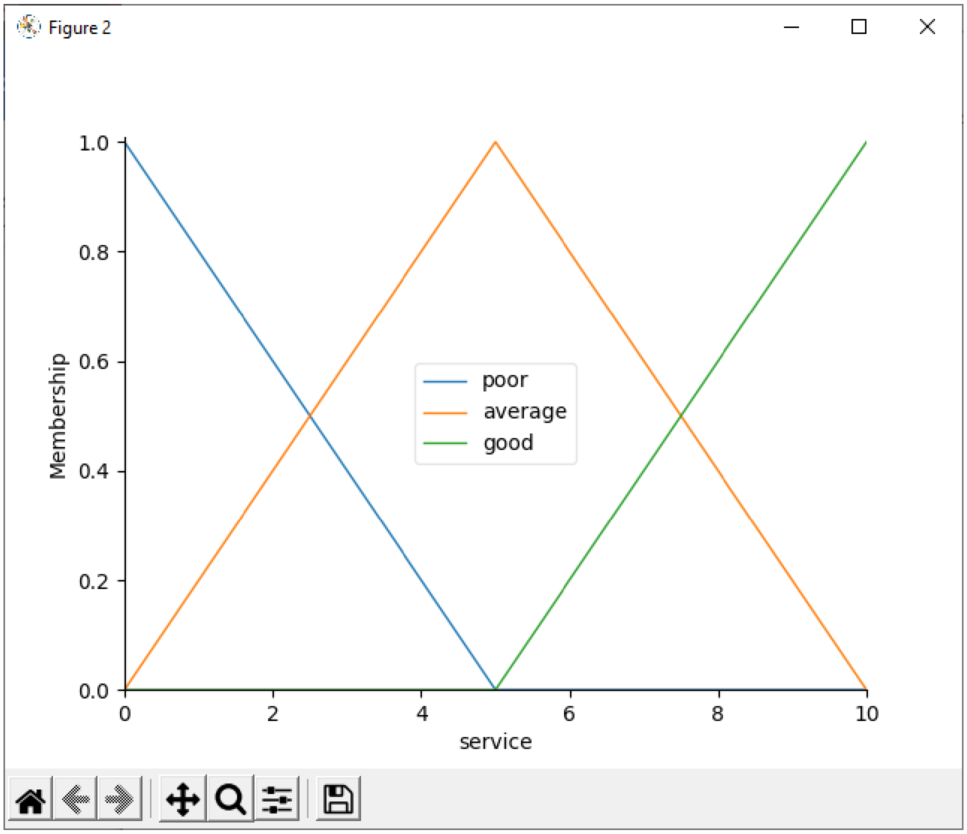


Figure 2
Membership
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
service
poor
average
good

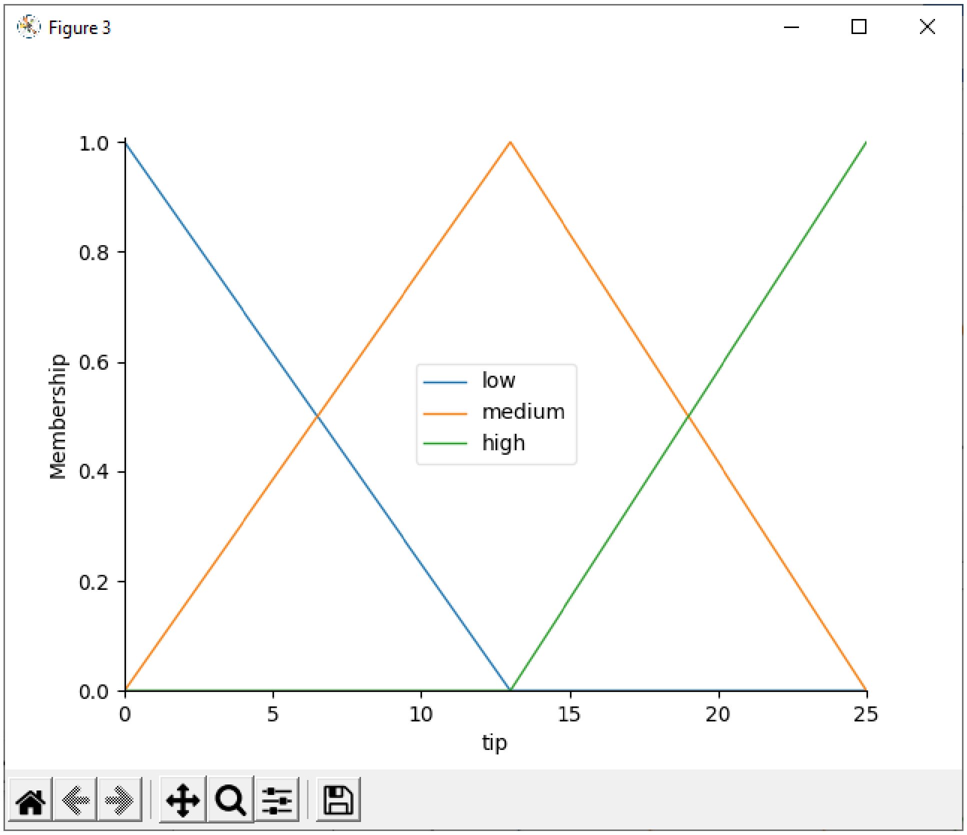
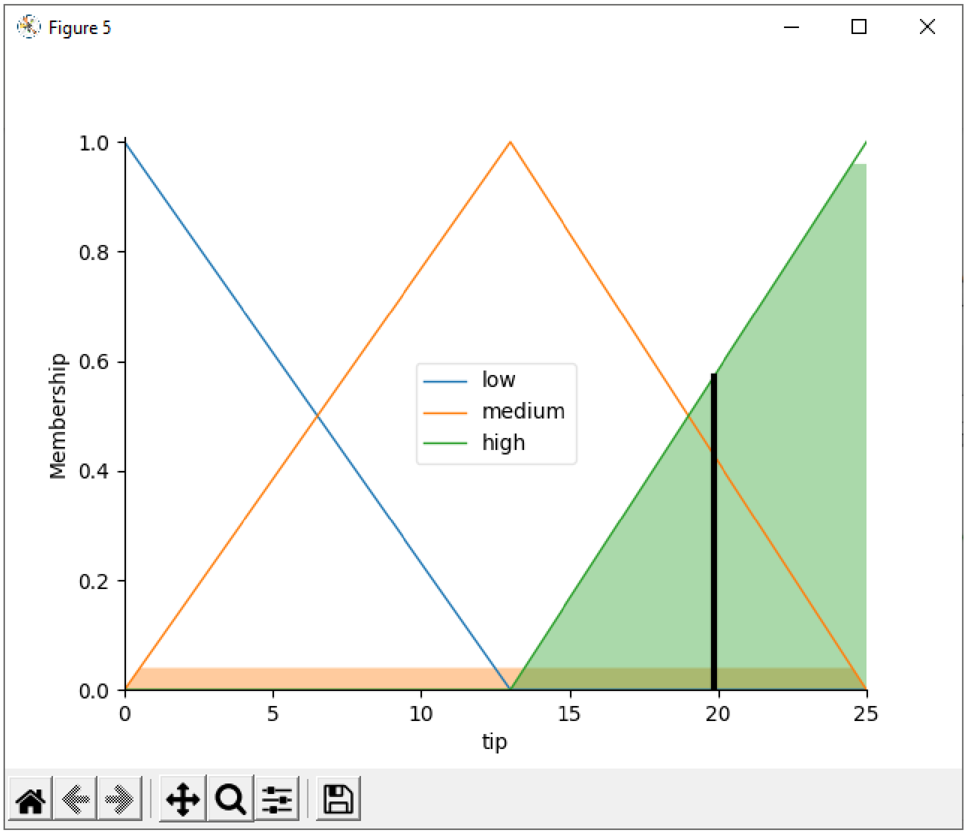
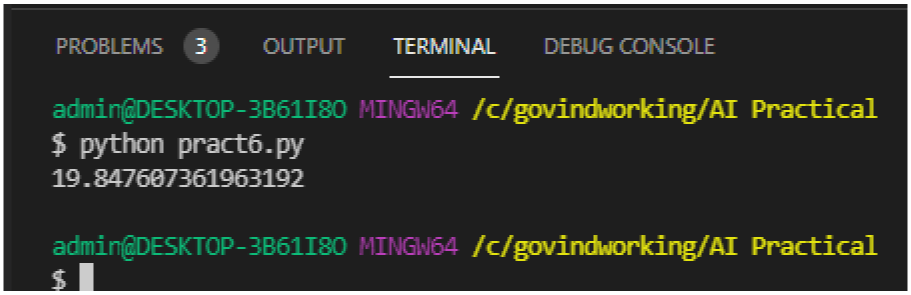


Figure 3
Membership
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
tip
low
medium
high





```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ python pract6.py
19.847607361963192

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ 
```

Practical : 7

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

Description :

What is supervised learning?

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically, supervised learning is a learning in which we teach or train the machine using data which is well labelled that means some data is already tagged with the correct answer.

Supervised learning classified into two categories of algorithms:

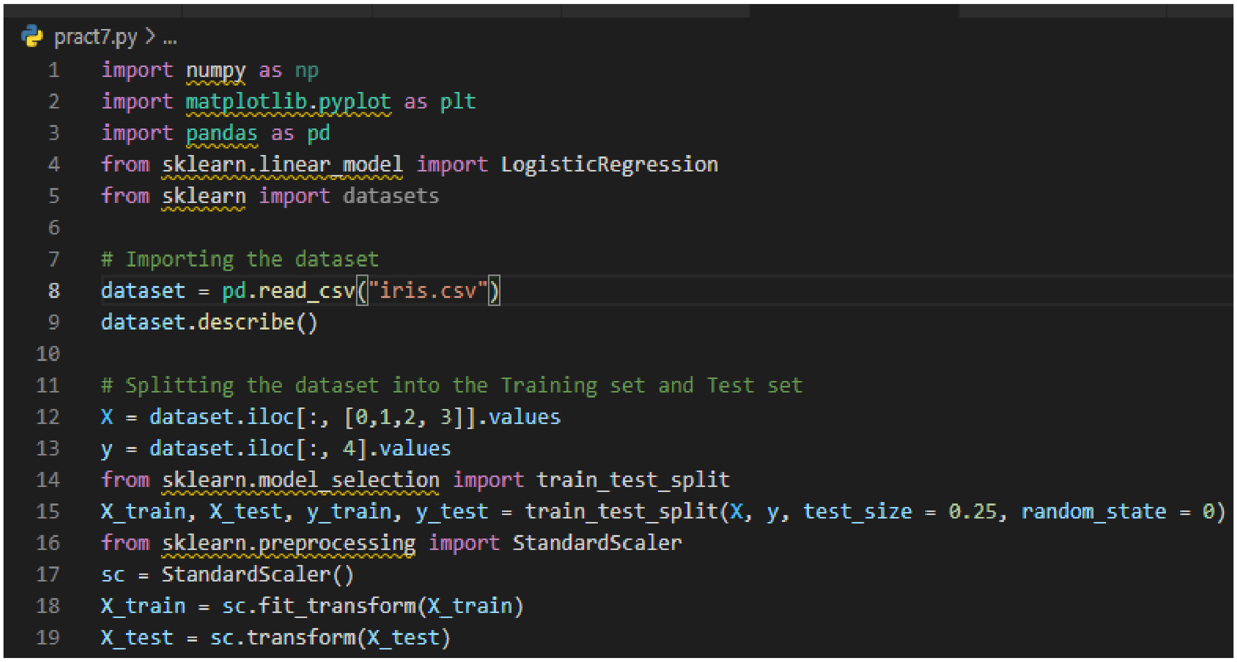
- Classification: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Supervised learning deals with or learns with “labelled” data. Which implies that some data is already tagged with the correct answer.

Types: -

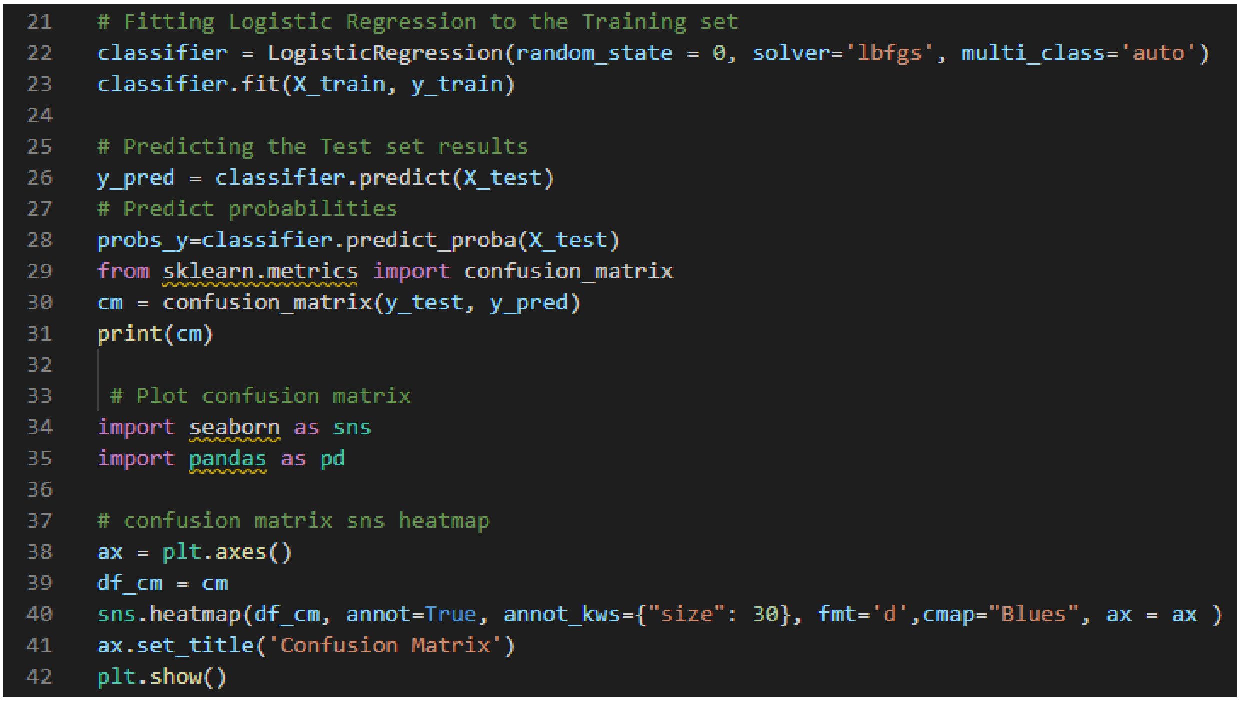
- **Regression**
- **Logistic Regression**
- **Classification**
- **Naive Bayes Classifiers**
- **K-NN (k nearest neighbours)**
- **Decision Trees**
- **Support Vector Machine**

Code :



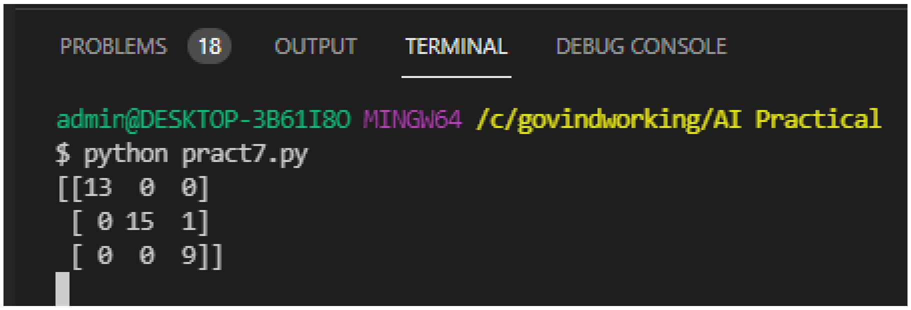
pract7.py > ...

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn import datasets

# Importing the dataset
dataset = pd.read_csv("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
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

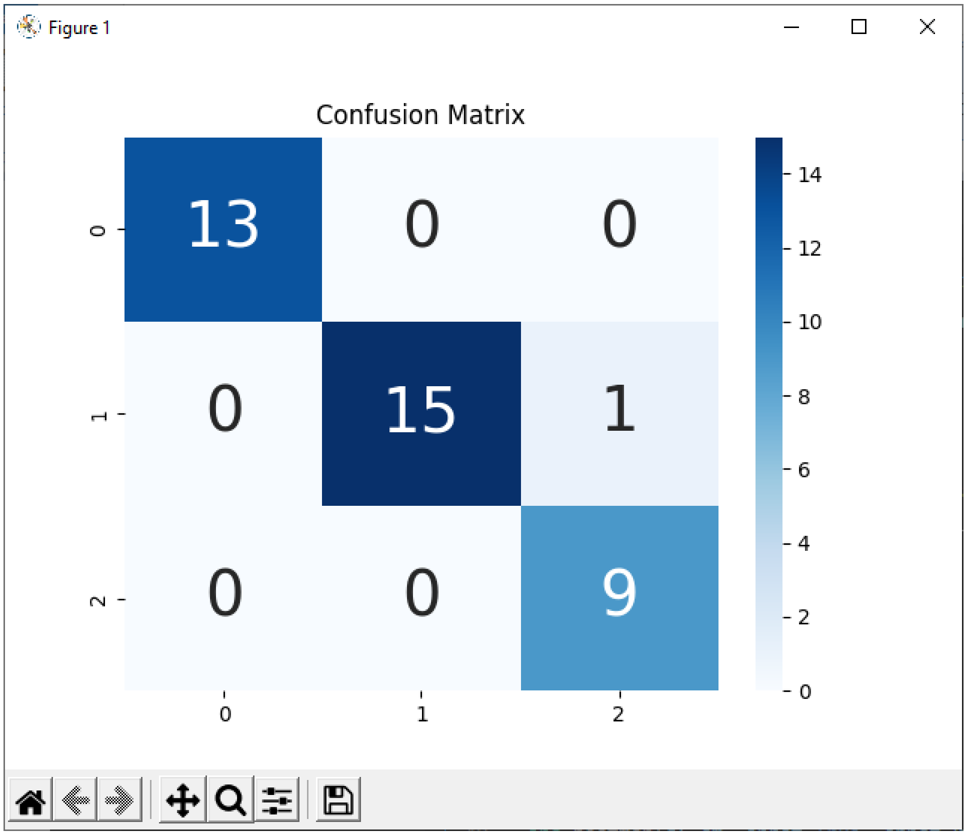


```
# Fitting Logistic Regression to the Training set
classifier = LogisticRegression(random_state = 0, solver='lbfgs', multi_class='auto')
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Predict probabilities
probs_y=classifier.predict_proba(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

 # Plot confusion matrix
import seaborn as sns
import pandas as pd

# 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()
```

Ouput :



```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ python pract7.py
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```



Description :

What is Unsupervised Learning?

Unsupervised learning is the training of machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Unsupervised learning classified into two categories of algorithms:

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.

- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Types of Unsupervised Learning: -

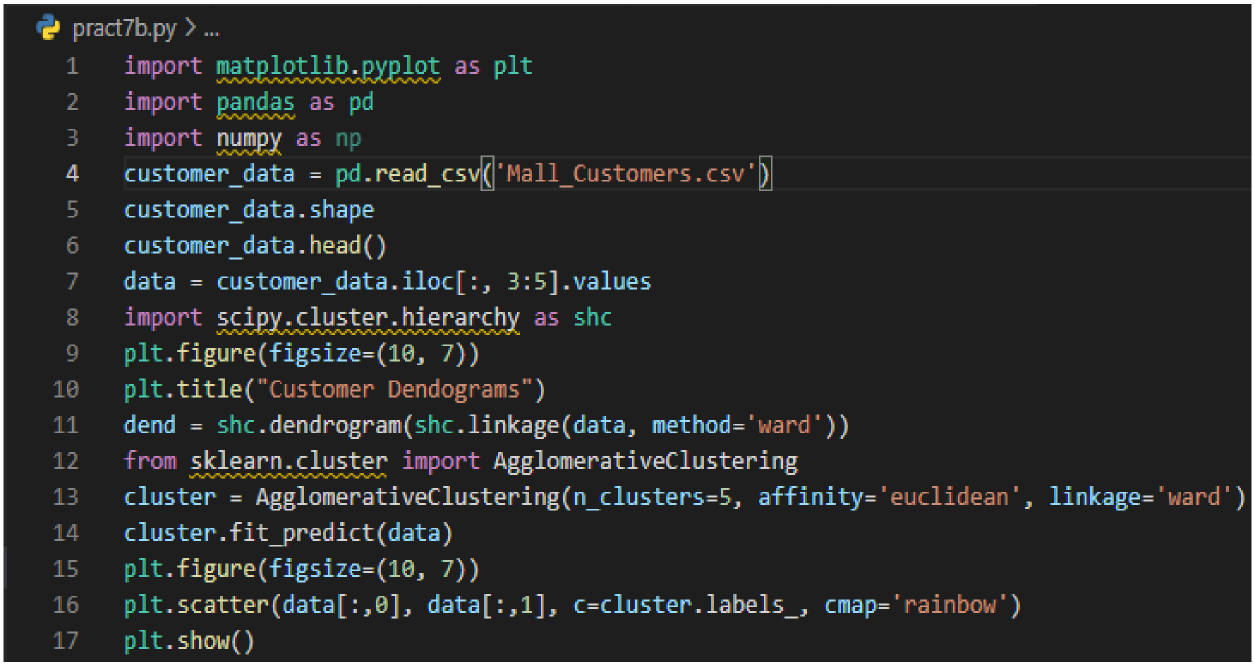
Clustering

- Exclusive (partitioning)
- Agglomerative
- Overlapping
- Probabilistic

Clustering Types: -

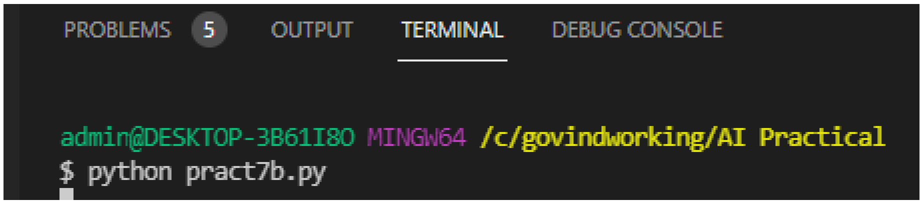
- Hierarchical clustering
- K-means clustering
- Principal Component Analysis
- Singular Value Decomposition
- Independent Component Analysis

Code :

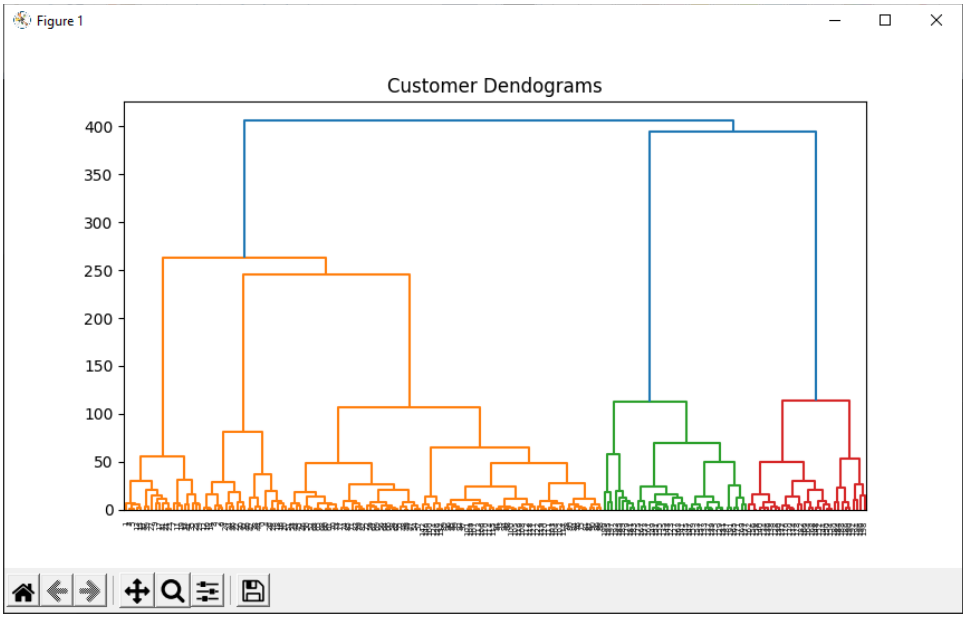


```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
customer_data = pd.read_csv('Mall_Customers.csv')
customer_data.shape
customer_data.head()
data = customer_data.iloc[:, 3:5].values
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))
from sklearn.cluster import AgglomerativeClustering
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()
```

Ouput :



```
admin@DESKTOP-3B61I8O MINGW64 /c/govindworking/AI Practical
$ python pract7b.py
```



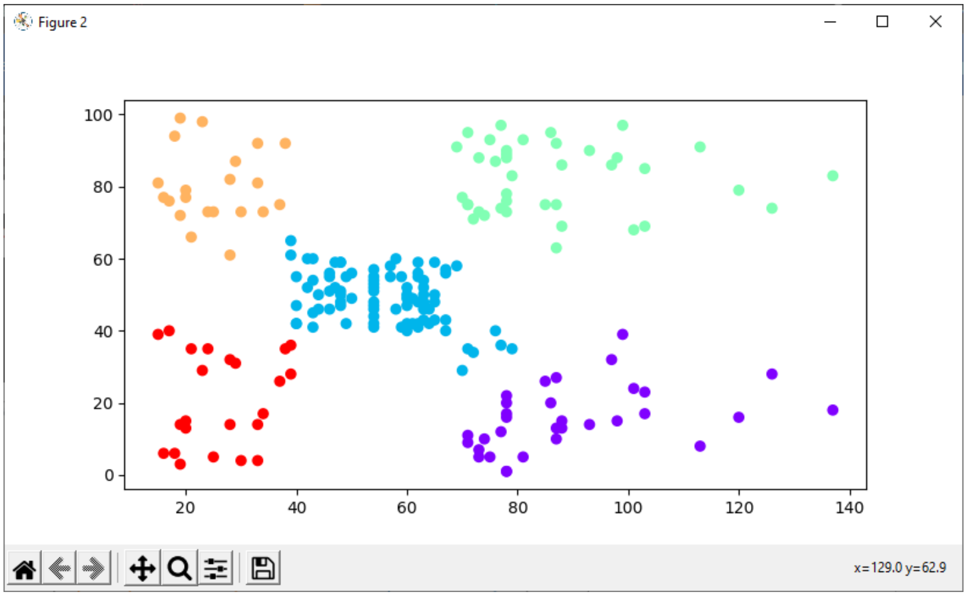


Figure 2
100
80
60
40
20
0
20
40
60
80
100
120
140
x=129.0 y=62.9

Practical : 8

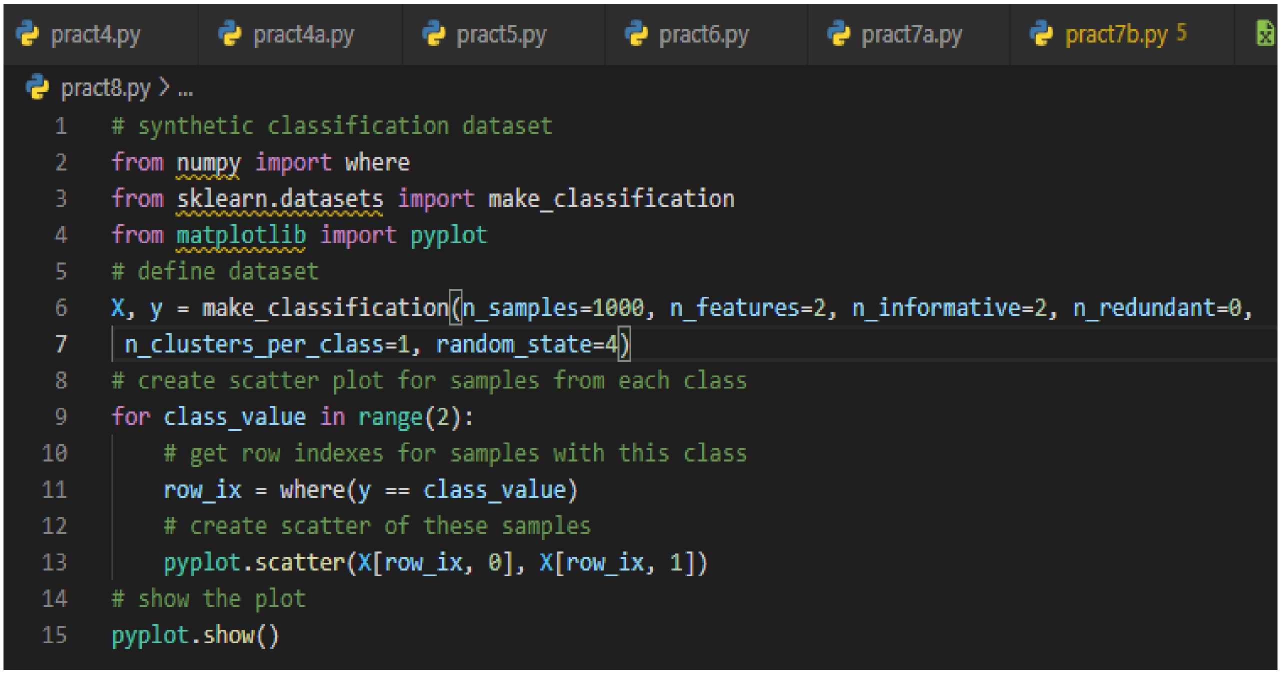
Aim : Write an application to implement Clustering algorithm.

Description :

What is Clustering?

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups.

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 with this 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()
```

Ouput :

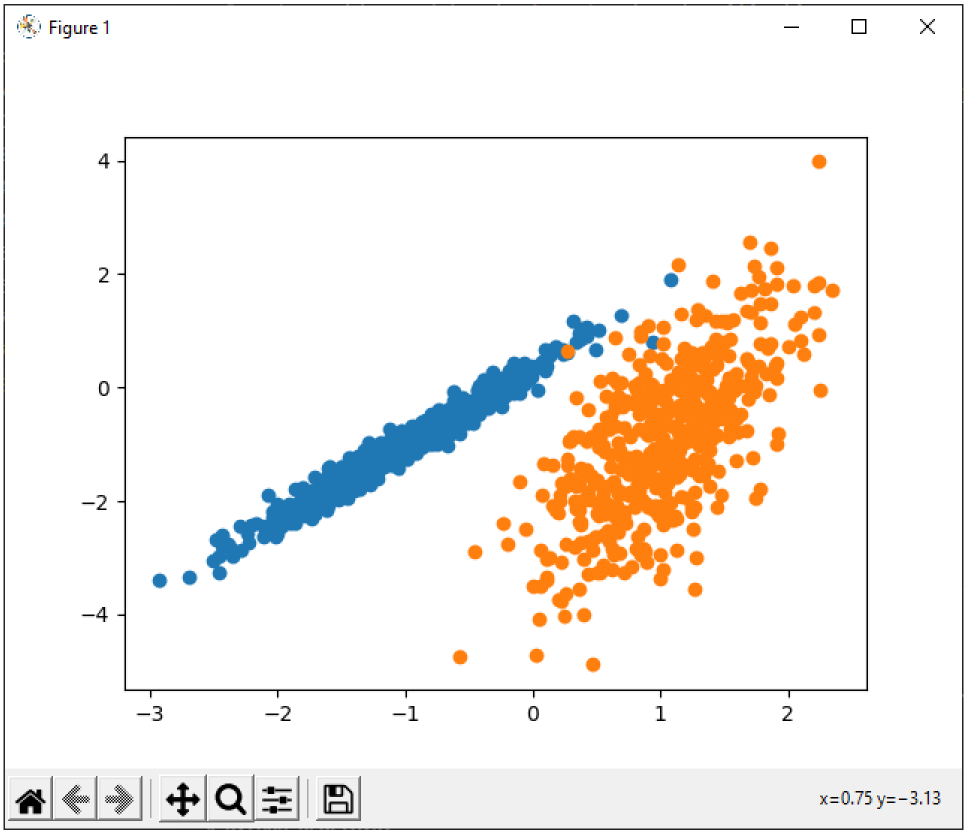


Figure 1
4
2
0
−2
−4
−3
−2
−1
0
1
2
x=0.75 y=−3.13

Practical : 9

Aim : Write an Program to implement BFS algorithm.

Description :

What is Breadth-First Search?

Breadth-First Search (BFS) is an algorithm used for traversing graphs or trees. Traversing means visiting each node of the graph. Breadth-First Search is a recursive algorithm to search all the vertices of a graph or a tree. BFS in python can be implemented by using data structures like a dictionary and lists. As breadth-first search is the process of traversing each node of the graph, a standard BFS algorithm traverses each vertex of the graph into two parts:

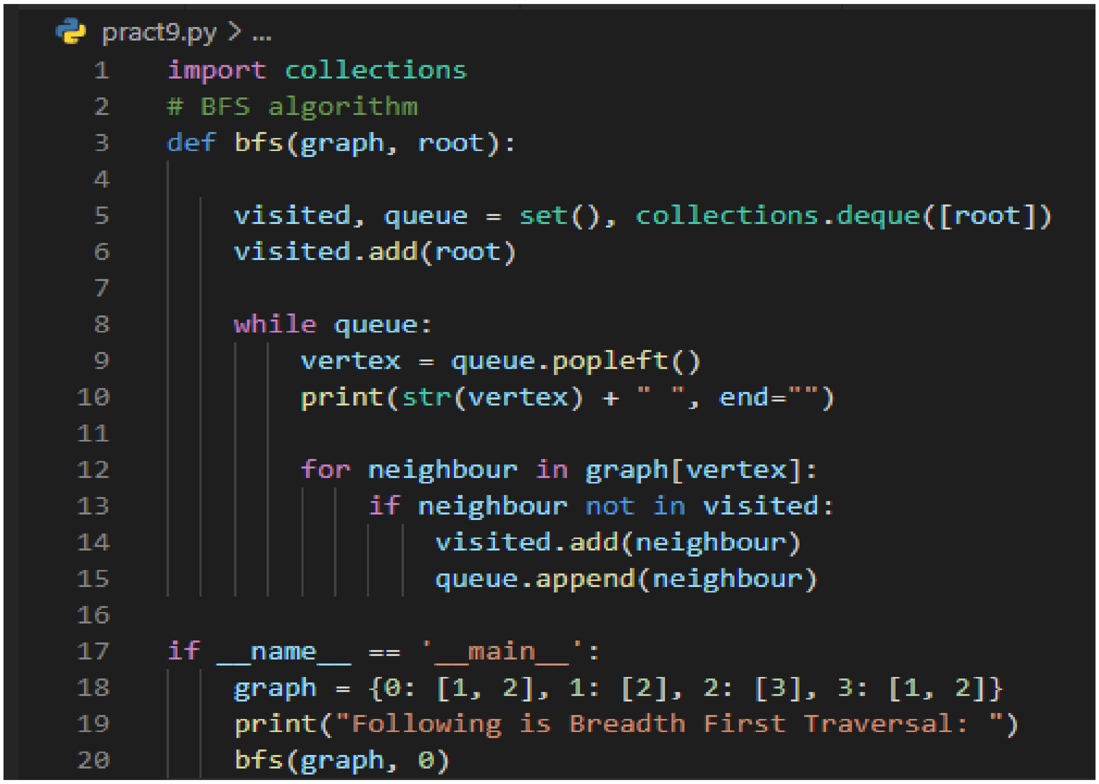
1) Visited

2) Not Visited. So, the purpose of the algorithm is to visit all the vertex while avoiding cycles.

The steps of the algorithm work as follow:

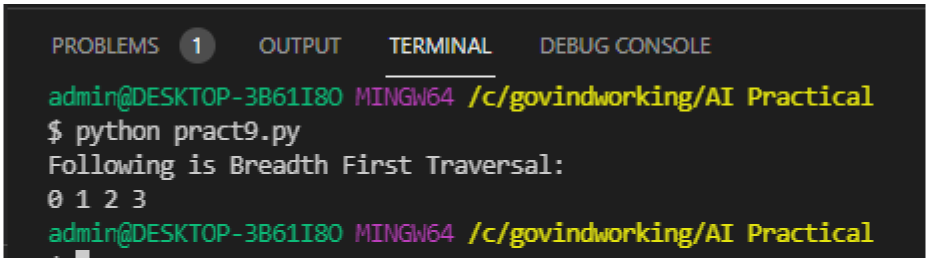
1. Start by putting any one of the graph's vertices at the back of the queue.
2. Now take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add those which are not within the visited list to the rear of the queue.
4. Keep continuing steps two and three till the queue is empty.

Code:



```python
import collections
# BFS algorithm
def bfs(graph, root):

    visited, queue = set(), collections.deque([root])
    visited.add(root)

    while queue:
        vertex = queue.popleft()
        print(str(vertex) + " ", end="")

        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)

if __name__ == '__main__':
    graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
    print("Following is Breadth First Traversal: ")
    bfs(graph, 0)
```

Output:



```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ python pract9.py
Following is Breadth First Traversal:
0 1 2 3
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
```

Practical : 10

Aim : Write an Program to implement DFS algorithm.

Description :

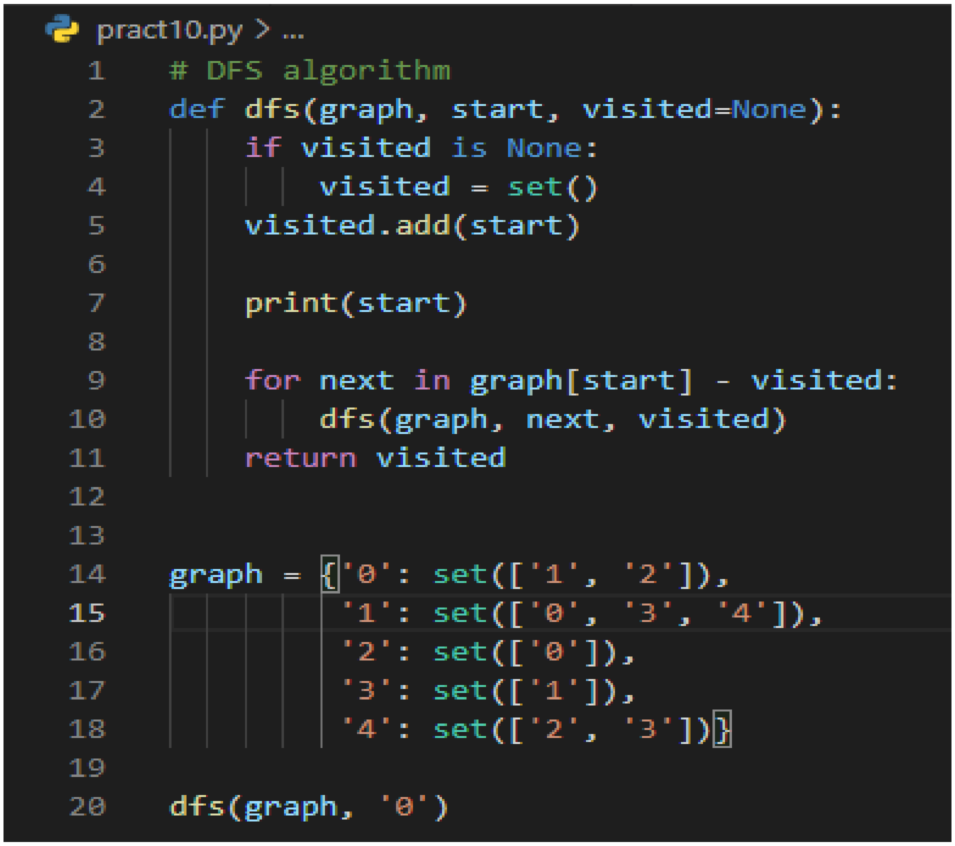
What is **Depth-First Search ?**

The Depth-First Search is a recursive algorithm that uses the concept of backtracking. It involves thorough searches of all the nodes by going ahead if potential, else by backtracking. Here, the word backtrack means once you are moving forward and there are not any more nodes along the present path, you progress backward on an equivalent path to seek out nodes to traverse.

Algorithm:

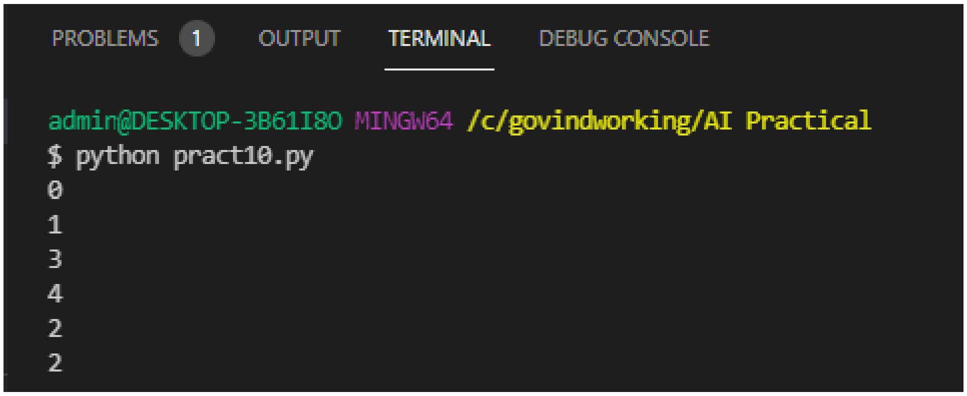
- Create a recursive function that takes the index of the node and a visited array.
- Mark the current node as visited and print the node.
- Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.

Code :



```
# 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 :



```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/AI Practical
$ python pract10.py
0
1
3
4
2
2
```