DEPARTMENT OF INFORMATION TECHNOLOGY

SMT. PARMESHWARIDEVI DURGADUTT TIBREWALA

LIONS JUHU COLLEGE

OF ARTS, COMMERE AND SCIENCE

Affiliated to University of Mumbai

J.B. NAGAR, ANDHERI (E), MUMBAI-400059

Academic Year 2022-2023

Applied Artificial Intelligence

For

Semester III

Submitted By:

MR. TUFAIL KHAN

Msc.IT (Sem III)

SMT. PARMESHWARIDEVI DURGADUTT TIBREWALA

LIONS JUHU COLLEGE

OF ARTS, COMMERE AND SCIENCE

Affiliated to University of Mumbai

J.B. NAGAR, ANDHERI (E), MUMBAI-400059

DEPARTMENT OF INFORMATION TECHNOLOGY

Certificate of Approval

This is to certify that practical entitled "**Applied Artificial Intelligence**", Undertaken at **SMT.PARMESHWARIDEVI DURGADUTT TIBREWALA LIONS JUHU COLLEGE OF ARTS, COMMERECE & SCIENCE.** By **MR. TUFAIL KHAN Seat No. 3269731** in partial fulfilment of **M.Sc. (IT) master degree (Semester III)** Examination had not been submitted for any other examination and does not form of any other course undergone by the candidate. It is further certified that she has completed all required phases of the practical.

__________ __________

Internal Examiner External Examiner

______________ __________

HOD / In-Charge / Coordinator Signature/ Principal/Stamp

INDEX

Practical No. 1

Aim: Implement Bayes Theorem using Python.

Code:

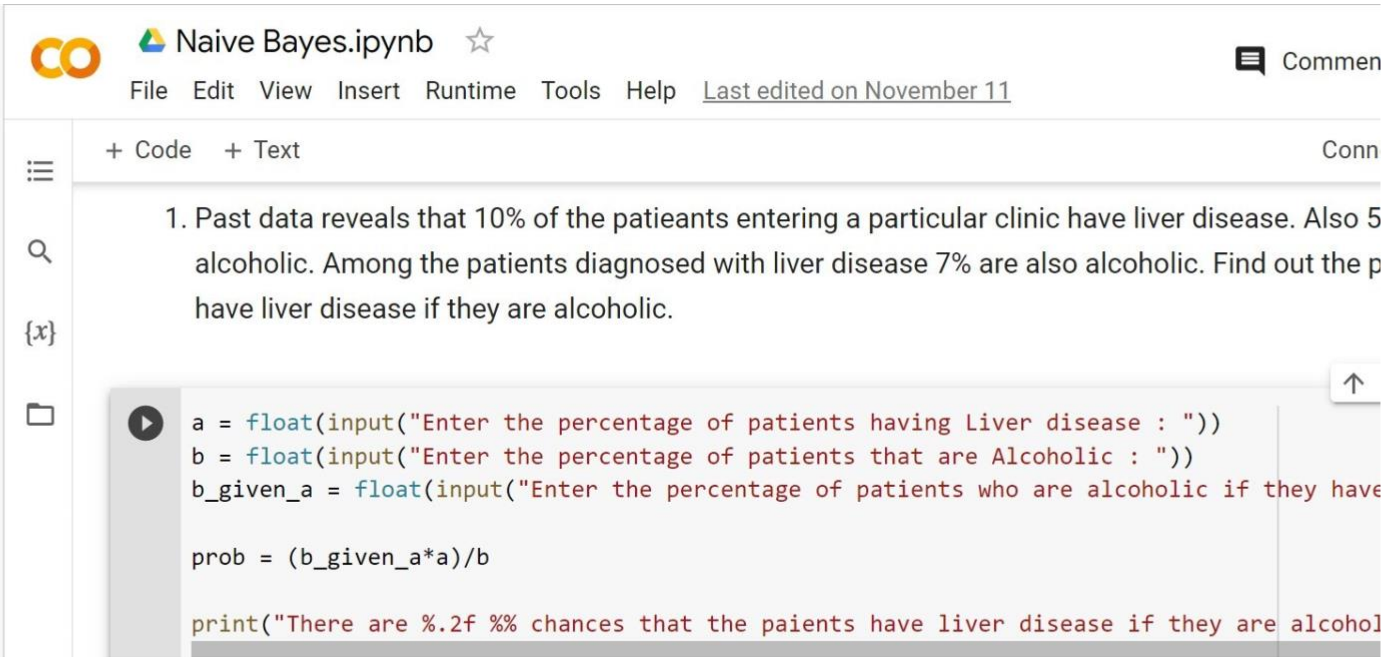
1. Past data reveals that 10% of the patients entering a particular clinic have liver disease. Also 5% of the patients are alcoholic. Among the patients diagnosed with liver disease 7% are also alcoholic. Find out the probability that the patients have liver disease if they are alcoholic.

```
a = float(input("Enter the percentage of patients having Liver disease : "))
b = float(input("Enter the percentage of patients that are Alcoholic : "))
b_given_a = float(input("Enter the percentage of patients who are alcoholic if they have liver
disease : "))
prob = (b_given_a*a)/b

print("There are %.2f %% chances that the paients have liver disease if they are alcoholic."%(prob))
```

2. Given that in a particular sample space, 1% of the patients have a certain genetic defect. 90% of the test for the gene detect the defect i.e., they are true positives. 9.6% of the test are false positives. If a person gets a positive test result, what are the chances that are actually have the genetic defect?

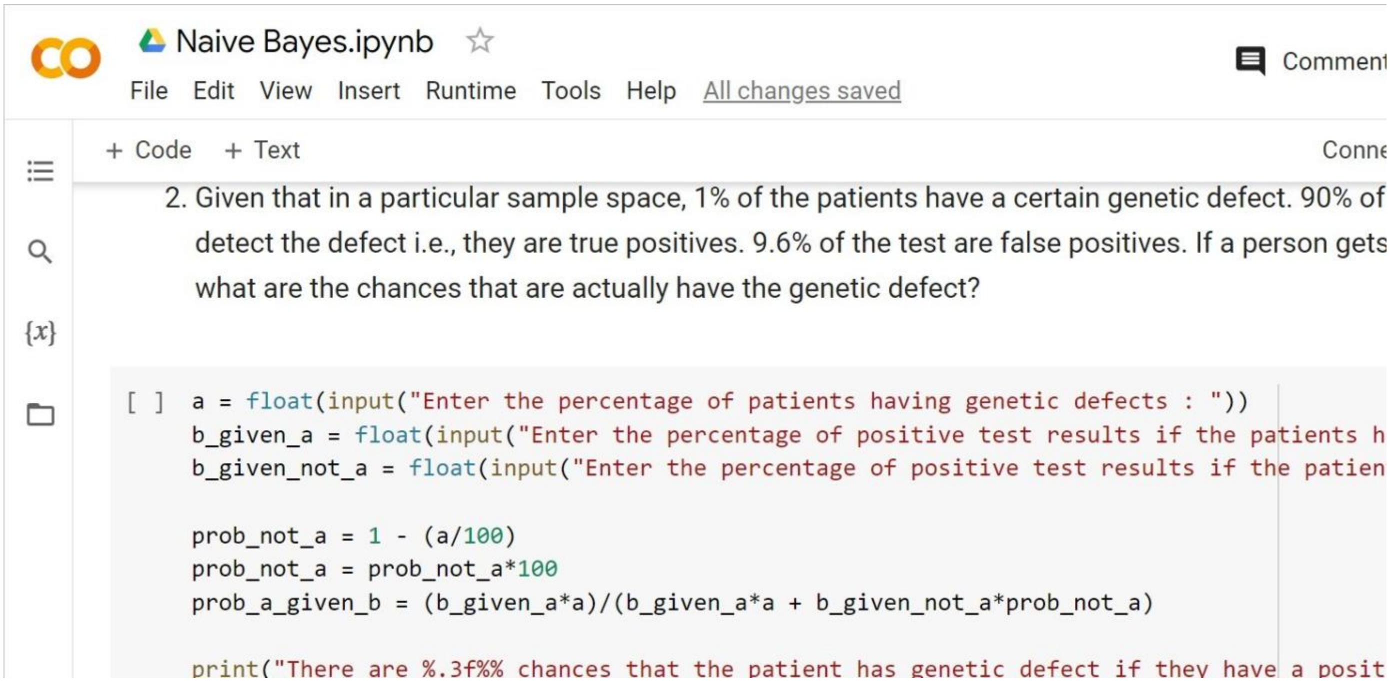
```
a = float(input("Enter the percentage of patients having genetic defects : "))
b_given_a = float(input("Enter the percentage of positive test results if the patients have the
genetic effect : "))
b_given_not_a = float(input("Enter the percentage of positive test results if the patients do not have
the genetic effect : "))

prob_not_a = 1 - (a/100)
prob_not_a = prob_not_a*100
prob_a_given_b = (b_given_a*a)/(b_given_a*a + b_given_not_a*prob_not_a)

print("There are %.3f%% chances that the patient has genetic defect if they have a positive test
result."%(prob_a_given_b))
```

Output:


Naive Bayes.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 11
Commen
+ Code + Text
Conn

1. Past data reveals that 10% of the patieants entering a particular clinic have liver disease. Also 5 alcoholic. Among the patients diagnosed with liver disease 7% are also alcoholic. Find out the p have liver disease if they are alcoholic.

```
a = float(input("Enter the percentage of patients having Liver disease : "))
b = float(input("Enter the percentage of patients that are Alcoholic : "))
b_given_a = float(input("Enter the percentage of patients who are alcoholic if they have

prob = (b_given_a*a)/b

print("There are %.2f %% chances that the paients have liver disease if they are alcohol
```



2. Given that in a particular sample space, 1% of the patients have a certain genetic defect. 90% of detect the defect i.e., they are true positives. 9.6% of the test are false positives. If a person gets what are the chances that are actually have the genetic defect?

```
[ ] a = float(input("Enter the percentage of patients having genetic defects : "))
    b_given_a = float(input("Enter the percentage of positive test results if the patients h
    b_given_not_a = float(input("Enter the percentage of positive test results if the patien

    prob_not_a = 1 - (a/100)
    prob_not_a = prob_not_a*100
    prob_a_given_b = (b_given_a*a)/(b_given_a*a + b_given_not_a*prob_not_a)

    print("There are %.3f%% chances that the patient has genetic defect if they have a posit
```

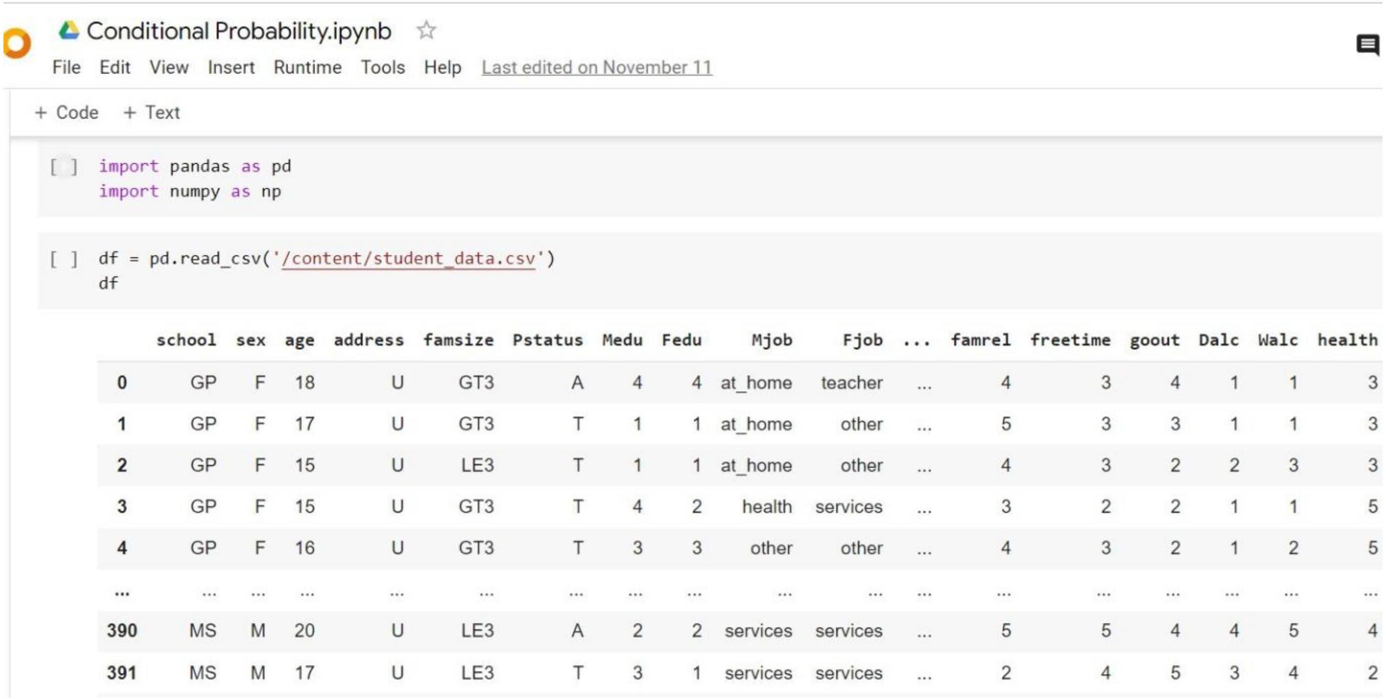
Practical No. 2

Aim: Implement Conditional Probability and Joint Probability using Python.

Code:

```
#Conditional Probability

import pandas as pd
import numpy as np

df = pd.read_csv('/content/student_data.csv')
df['G'] = round((df['G1']+df['G2']+df['G3'])/3)
df['Percentage'] = df['G'] * 5
df['O_grade'] = np.where(df['Percentage'] >= 80, 1, 0)
df['high_absentees'] = np.where(df['absences'] >= 10,1,0)
df['count'] = 1
df = df[['O_grade', 'high_absentees', 'count']]
ptable = pd.pivot_table(df, values='count', index = 'high_absentees', columns='O_grade', aggfunc= np.size, fill_value = 0)

total = 283+29+78+5
prob_a = (29+5)/total
prob_b = (78+5)/total
prob_a_intersect_b = 5/total
prob_a, prob_b, prob_a_intersect_b
prob_a_given_b = prob_a_intersect_b / prob_b

print("Probability of Students getting atleast 80% grade given they have missed 10 lectures or more is ", round(prob_a_given_b,2))

#Joint Probability

color = input('Enter the card colour : ')
number = input('Enter the card number : ')
prob_color = 26/52
prob_num = 4/52
print('Probability of drawing a ',color, 'card is ',round(prob_color,2))
print('Probability of drawing a card with number ',number, ' is ',prob_num)
prob_color_and_num = round(prob_color*prob_num, 2)
print('Probability of drawing ',color,' card with the number ',number,' from a normal deck of 52 playing cards is ',prob_color_and_num)
```

Output:

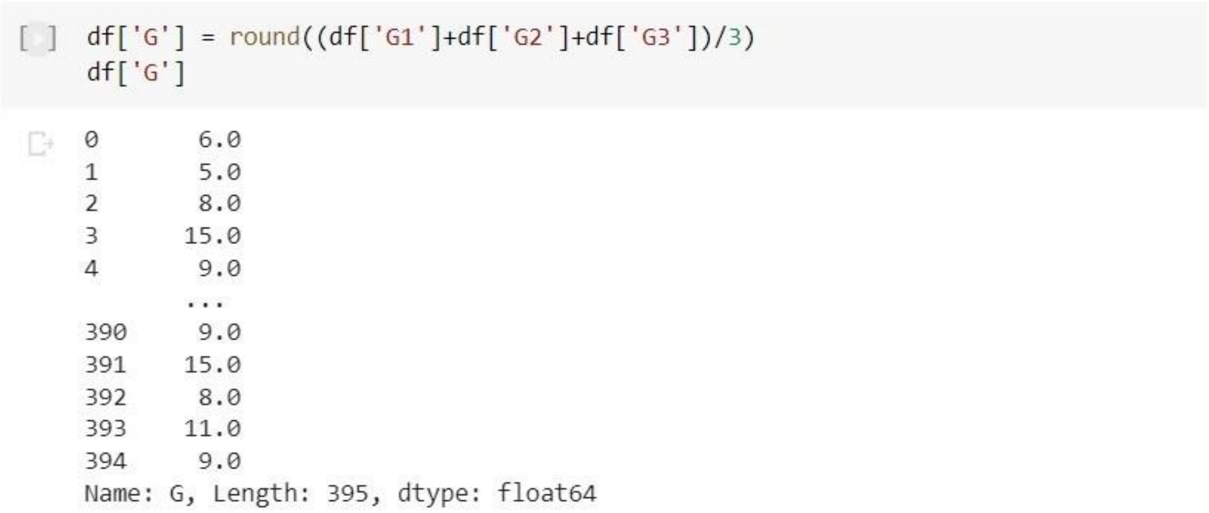


Conditional Probability.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 11
+ Code + Text

```
[ ] import pandas as pd
    import numpy as np
```

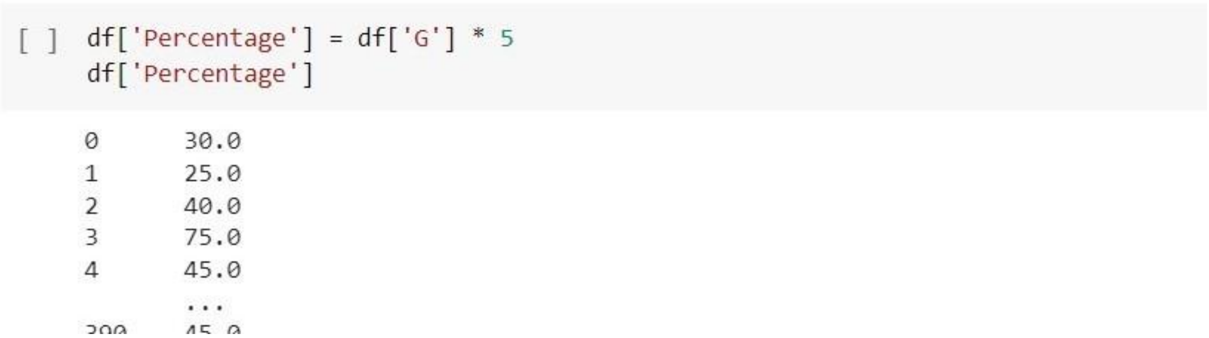
```
[ ] df = pd.read_csv('/content/student_data.csv')
    df
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4	4	5	4
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5	3	4	2



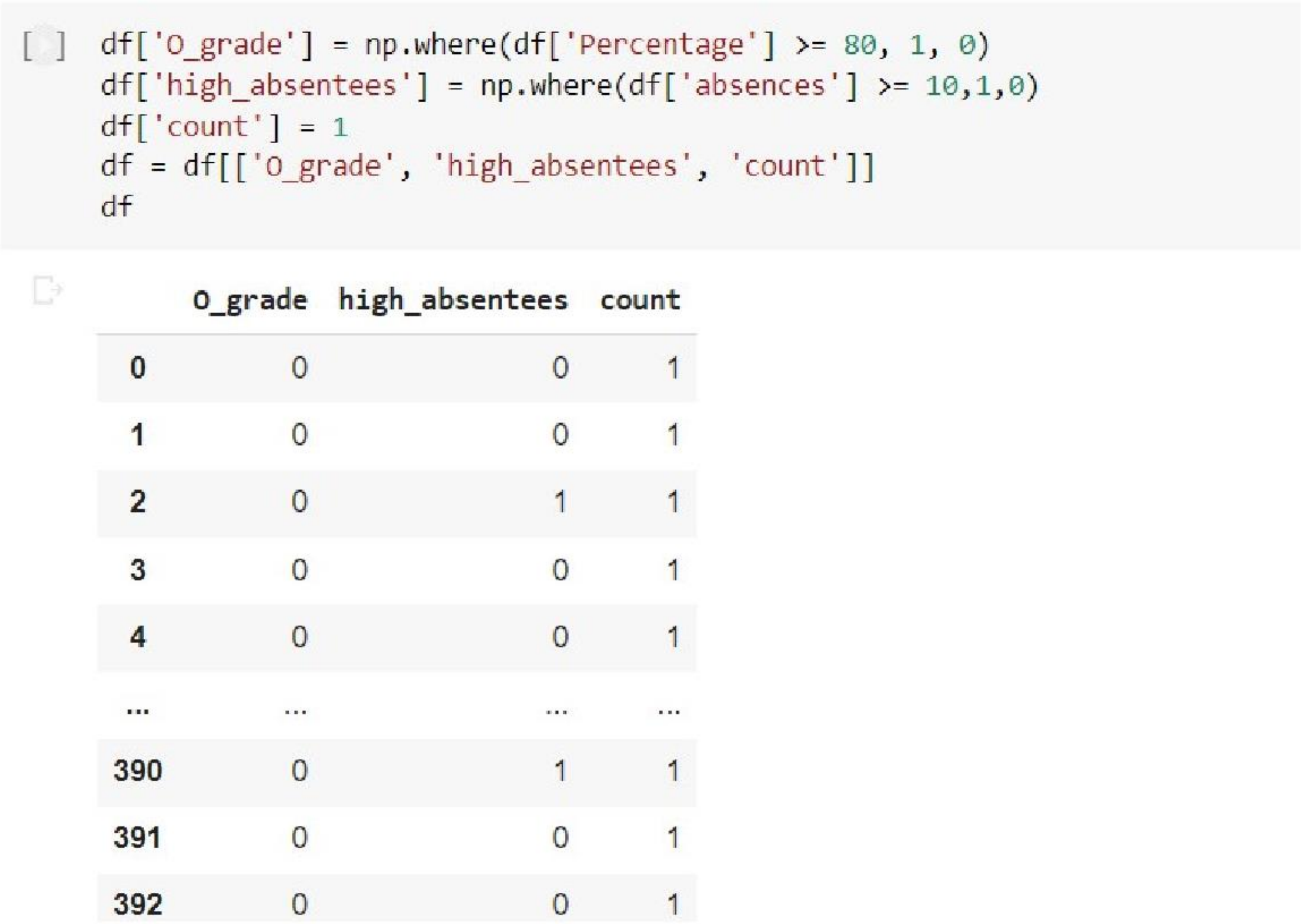
```
[ ] df['G'] = round((df['G1']+df['G2']+df['G3'])/3)
    df['G']
```

```
0       6.0
1       5.0
2       8.0
3      15.0
4       9.0
       ...
390     9.0
391    15.0
392     8.0
393    11.0
394     9.0
Name: G, Length: 395, dtype: float64
```



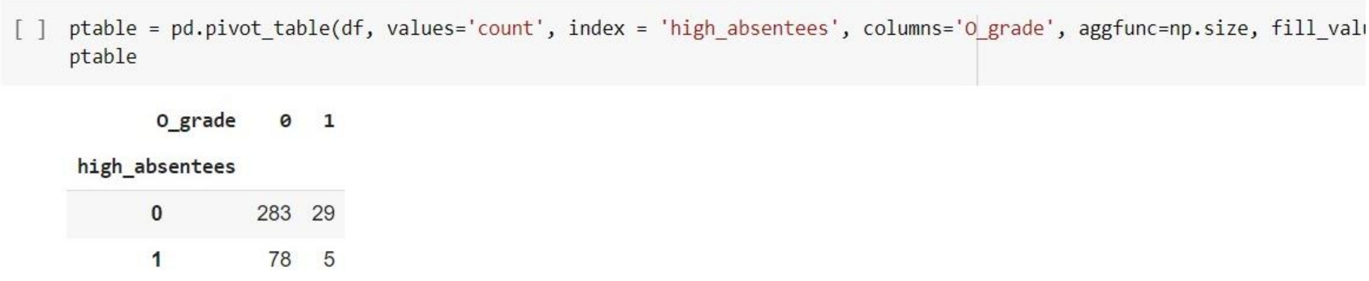
```
[ ] df['Percentage'] = df['G'] * 5
    df['Percentage']
```

```
0      30.0
1      25.0
2      40.0
3      75.0
4      45.0
       ...
390    45.0
```



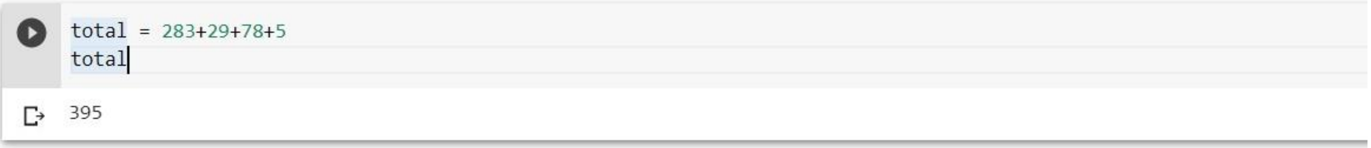
```
[ ] df['O_grade'] = np.where(df['Percentage'] >= 80, 1, 0)
    df['high_absentees'] = np.where(df['absences'] >= 10,1,0)
    df['count'] = 1
    df = df[['O_grade', 'high_absentees', 'count']]
    df
```

```
     O_grade  high_absentees  count
0          0               0      1
1          0               0      1
2          0               1      1
3          0               0      1
4          0               0      1
...      ...             ...    ...
390        0               1      1
391        0               0      1
392        0               0      1
```



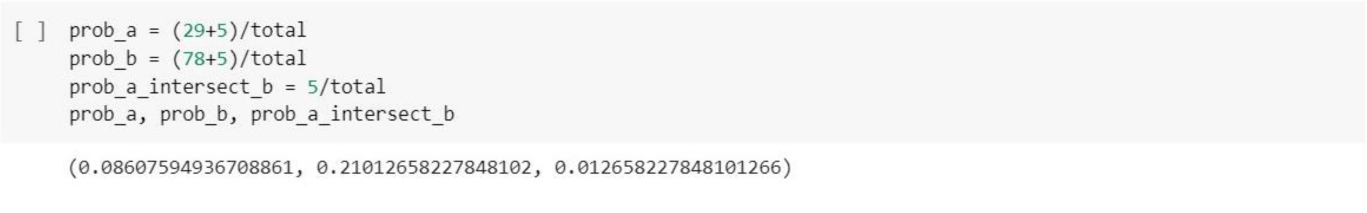
```
[ ] ptable = pd.pivot_table(df, values='count', index = 'high_absentees', columns='O_grade', aggfunc=np.size, fill_val
    ptable
```

```
       O_grade    0   1
high_absentees
             0  283  29
             1   78   5
```



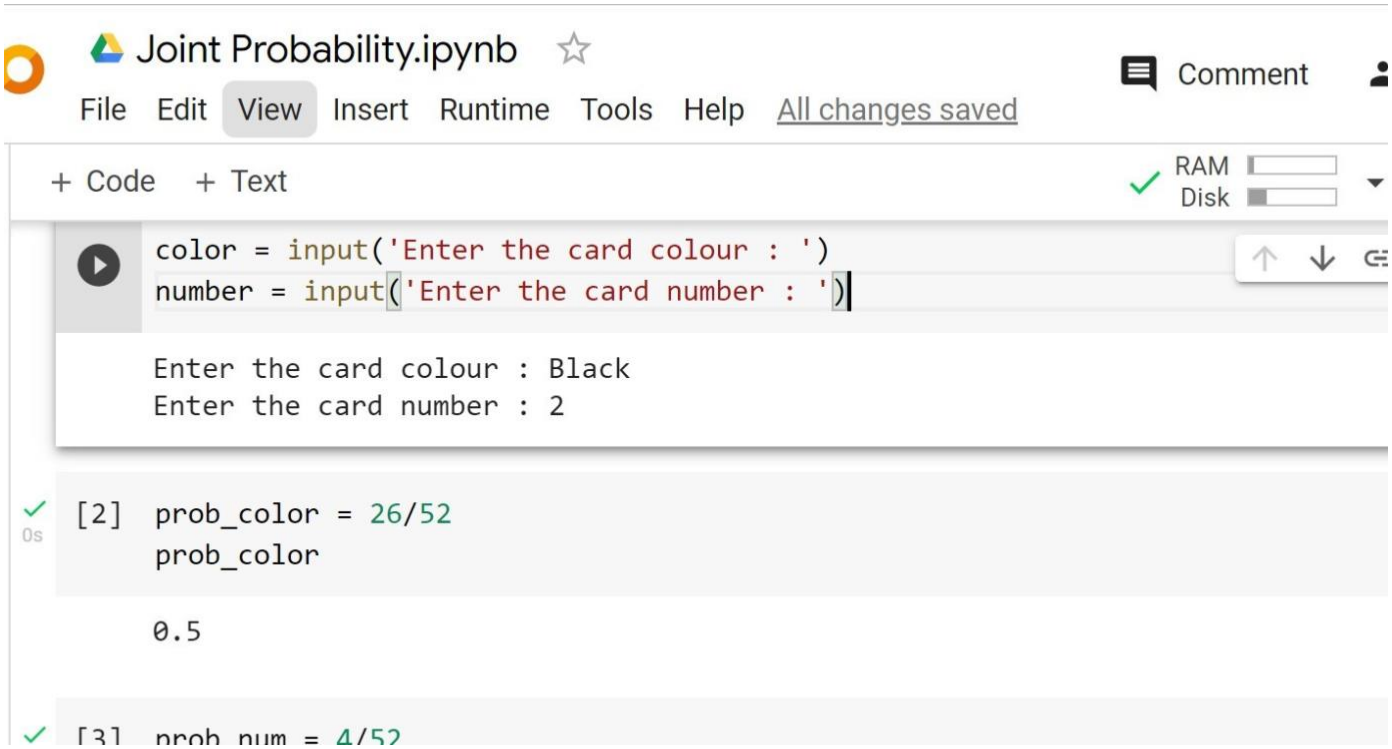
```
total = 283+29+78+5
total
```

```
395
```



```
[ ] prob_a = (29+5)/total
    prob_b = (78+5)/total
    prob_a_intersect_b = 5/total
    prob_a, prob_b, prob_a_intersect_b
```

```
(0.08607594936708861, 0.21012658227848102, 0.012658227848101266)
```



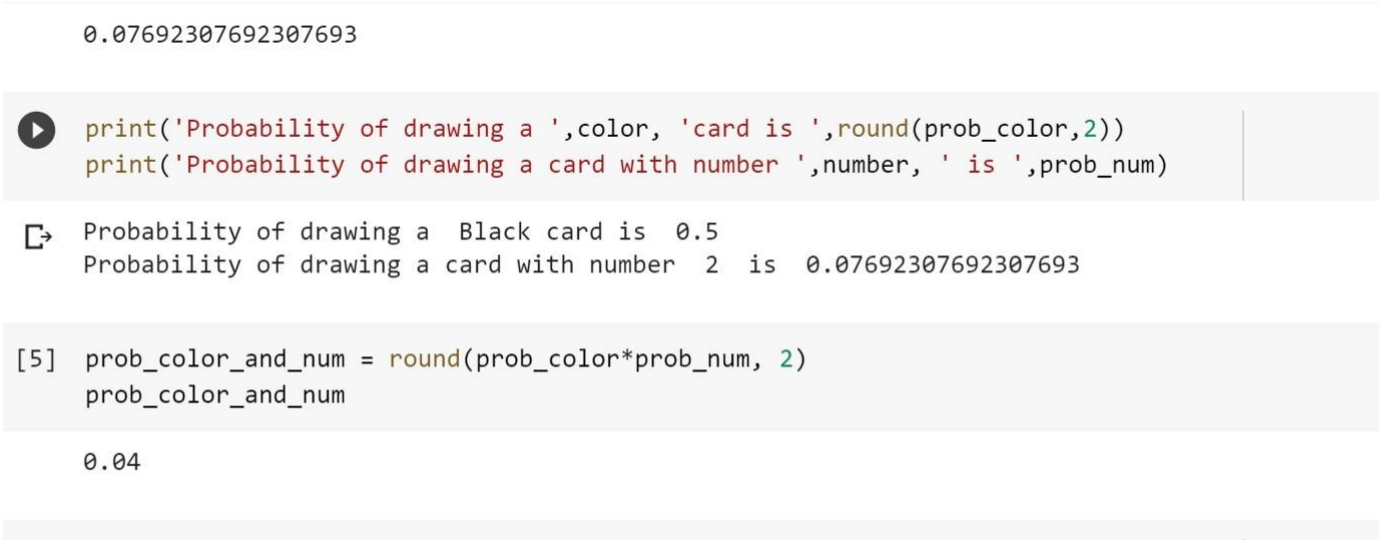
```
color = input('Enter the card colour : ')
number = input('Enter the card number : ')
```

```
Enter the card colour : Black
Enter the card number : 2
```

```
[2] prob_color = 26/52
    prob_color
```

```
0.5
```

```
[3] prob_num = 4/52
```



```
0.07692307692307693
```

```
print('Probability of drawing a ',color, 'card is ',round(prob_color,2))
print('Probability of drawing a card with number ',number, ' is ',prob_num)
```

```
Probability of drawing a  Black card is  0.5
Probability of drawing a card with number  2  is  0.07692307692307693
```

```
[5] prob_color_and_num = round(prob_color*prob_num, 2)
    prob_color_and_num
```

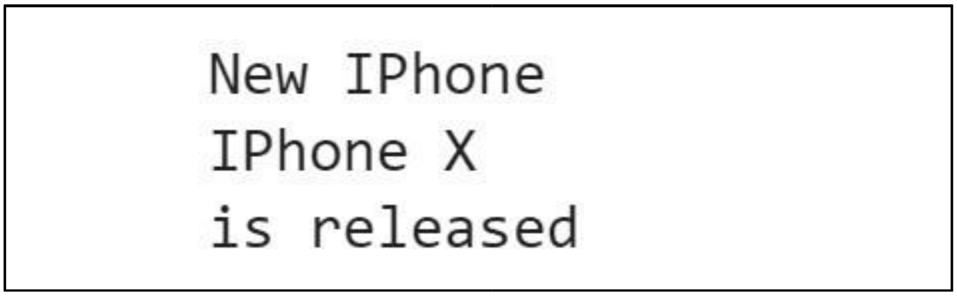
```
0.04
```

Practical No. 3

Aim: Write a program to implement Rule based system.

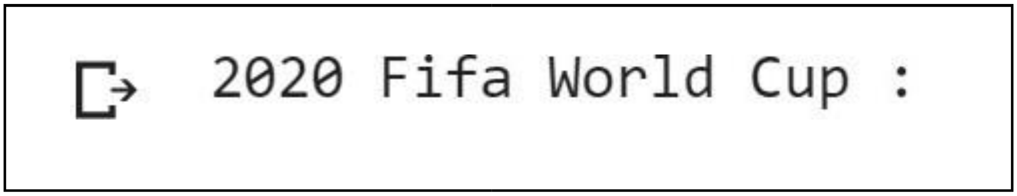
Code with Output:

```
import spacy
from spacy.matcher import Matcher

nlp=spacy.load('en_core_web_sm')
matcher=Matcher(nlp.vocab)

doc = nlp("New IPhone X is released")
pattern=[{'ORTH':'New'}, {'ORTH':'IPhone'}]
matcher.add('Iphone_pattern',[pattern])
matches = matcher(doc)

for match_id, start, end in matches:
  matched_span = doc[start:end]
  print(matched_span.text)
```



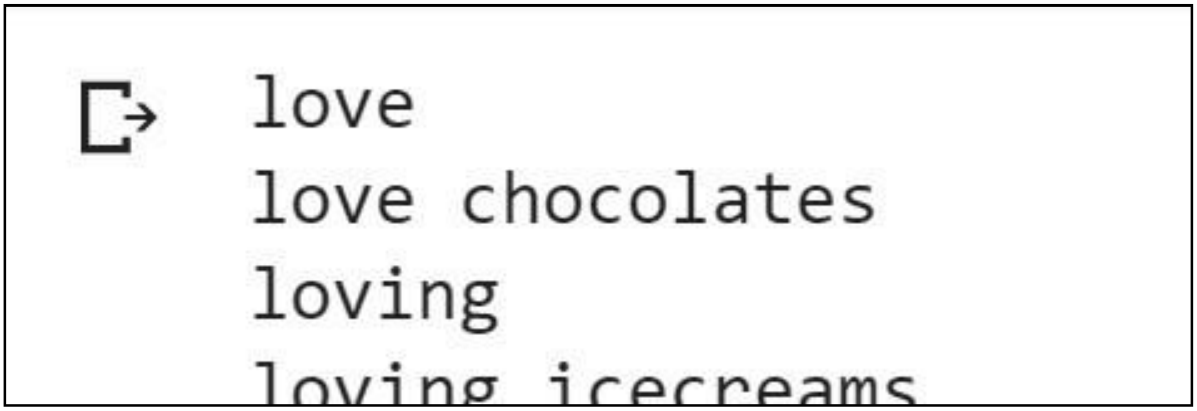
```
New IPhone
IPhone X
is released
```

```
doc = nlp("2020 Fifa World Cup : India Wins")
pattern=[{'IS_DIGIT':True}, {'LOWER':'fifa'}, {'LOWER':'world'}, {'LOWER':'cup'}, {'IS_PUNCT':True}]
matcher.add('FIFA_PATTERN',[pattern])

matches = matcher(doc)

for match_id, start, end in matches:
  matched_span = doc[start:end]
  print(matched_span.text,"\n")
```



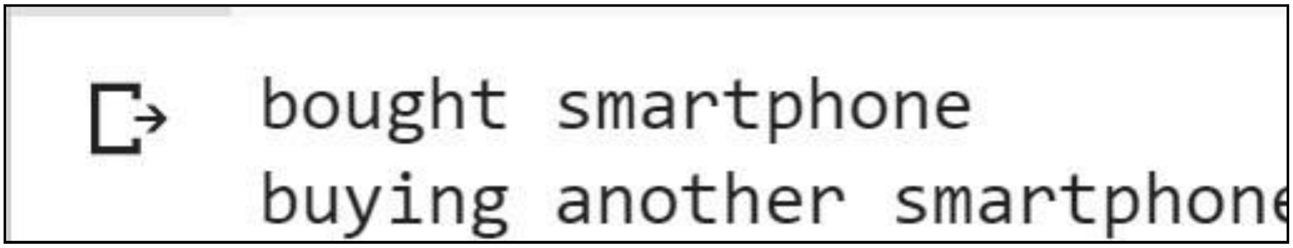
```
2020 Fifa World Cup :
```

```
doc = nlp("I love chocolates but now I loving icecreams more")
pattern=[{'LEMMA':'love'}, {'POS':'NOUN'}]
matcher.add('EAT_PATTERN',[pattern])

matches = matcher(doc)

for match_id, start, end in matches:
  matched_span = doc[start:end]
  print(matched_span.text)
```



```
love
love chocolates
loving
loving icecreams
```

```
doc = nlp("I bought smartphone now I am buying another smartphone")
pattern=[{'LEMMA':'buy'}, {'POS':'DET', "OP":'?'}, {'POS':'NOUN'}]
matcher.add('EA_PATTERN',[pattern])

matches = matcher(doc)

for match_id, start, end in matches:
  matched_span = doc[start:end]
  print(matched_span.text)
```



```
bought smartphone
buying another smartphone
```

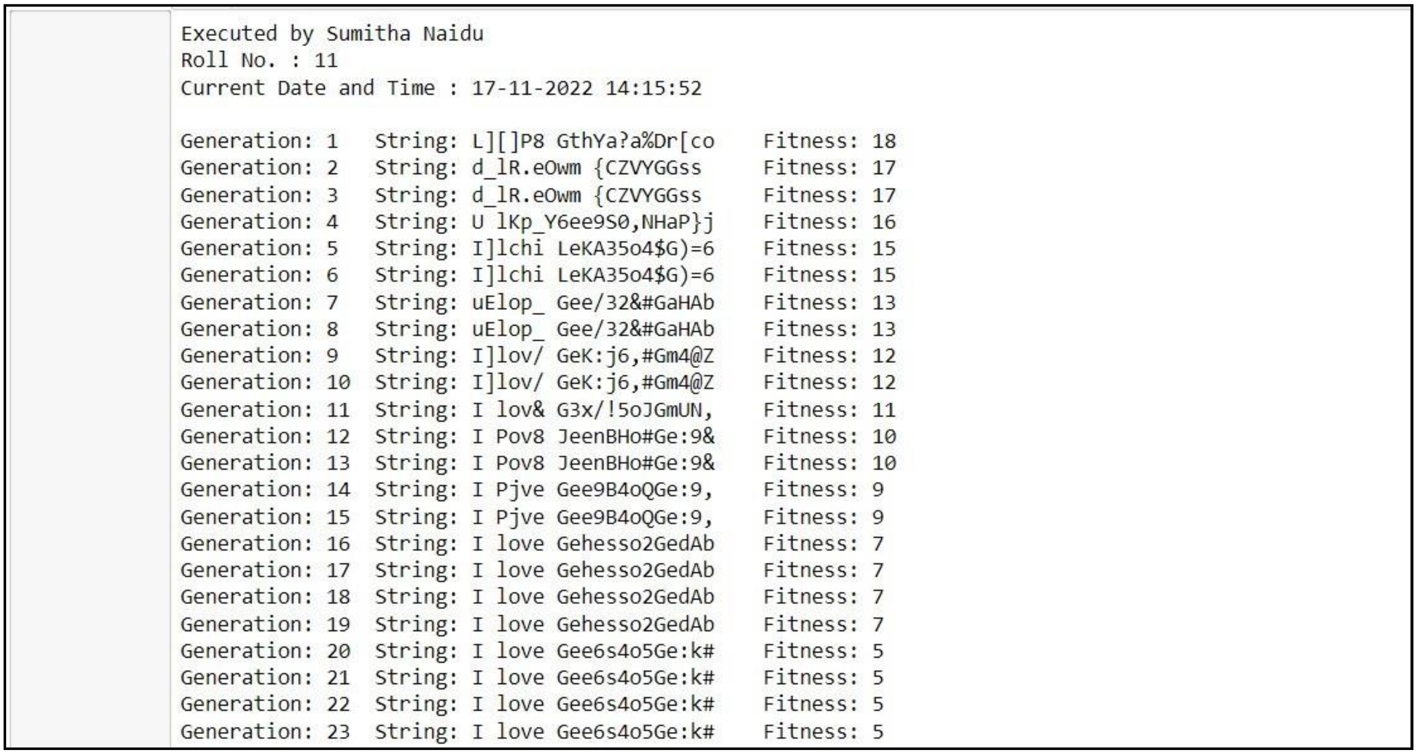
Practical No. 4

Aim: Simulate Genetic Algorithm with suitable example using Python.

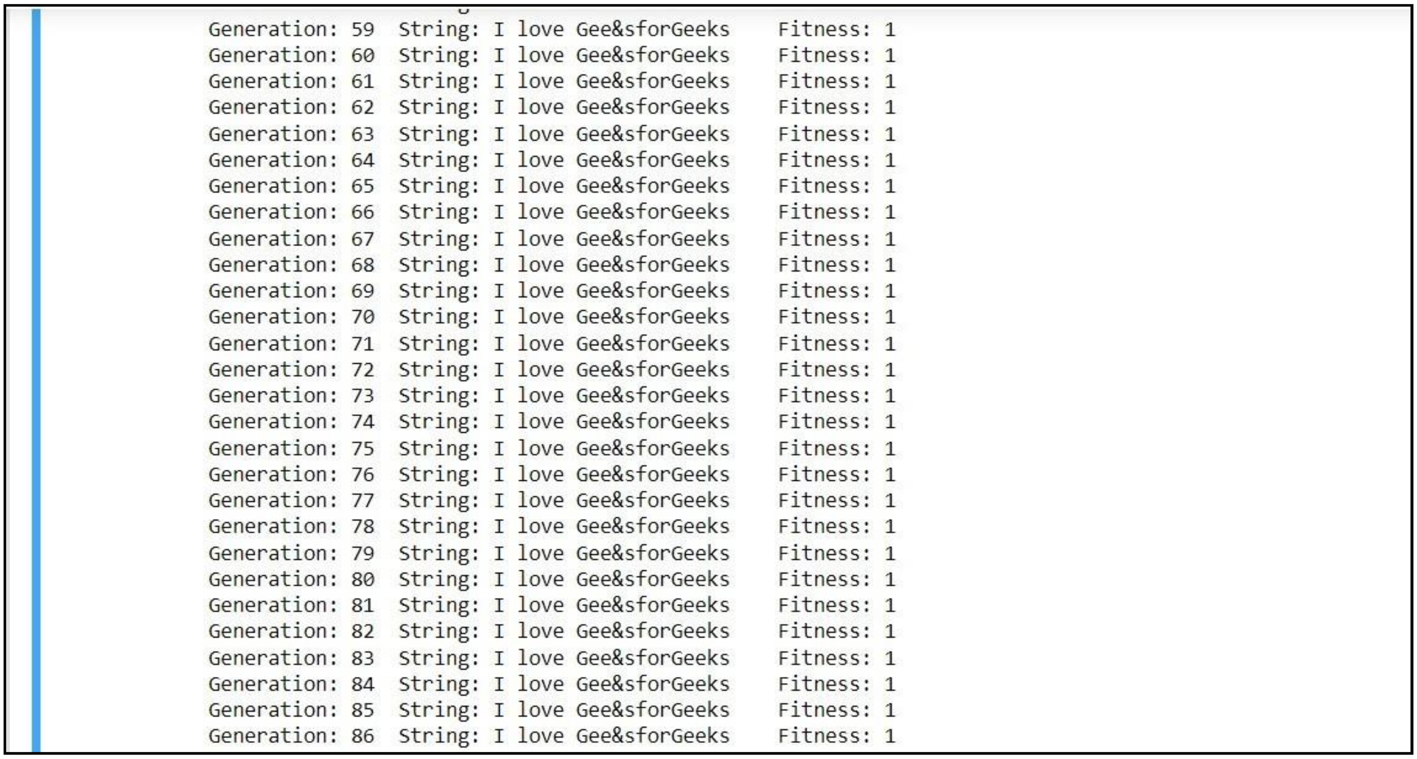
Code:

```
import datetime as dt
import random

# Number of individuals in each generation
POPULATION_SIZE = 100
# Valid genes
GENES = '''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890, .-
;:_!"#%&/()=?@${[]}'''
# Target string to be generated
TARGET = "I love GeeksforGeeks"

class Individual(object):
    def_init_(self, chromosome):
        self.chromosome = chromosome
        self.fitness = self.cal_fitness()
    @classmethod
    def mutated_genes(self):
        global GENES
        gene = random.choice(GENES)
        return gene
    @classmethod
    def create_gnome(self):
        global  TARGET
        gnome_len = len(TARGET)
        return [self.mutated_genes() for _ in range(gnome_len)]
    def mate(self, par2):
        child_chromosome = []
        for gp1, gp2 in zip(self.chromosome, par2.chromosome):
            prob = random.random()
            if prob < 0.45:
                child_chromosome.append(gp1)
            elif prob < 0.90:
                child_chromosome.append(gp2)
            else:
                child_chromosome.append(self.mutated_genes())
        return Individual(child_chromosome)
    def cal_fitness(self):
        global TARGET
        fitness = 0
        for gs, gt in zip(self.chromosome, TARGET):
            if gs != gt: fitness+= 1
        return fitness

# Driver code
def main():
    global POPULATION_SIZE
    #current generation
    generation = 1
    found = False
    population = []
    # create initial population
    for _ in range(POPULATION_SIZE):
        gnome = Individual.create_gnome()
        population.append(Individual(gnome))

    while not found:
        population = sorted(population, key = lambda x:x.fitness)
        if population[0].fitness <= 0:
            found = True
            break
        new_generation = []
        s = int((10*POPULATION_SIZE)/100)
        new_generation.extend(population[:s])
        s = int((90*POPULATION_SIZE)/100)
        for _ in range(s):
            parent1 = random.choice(population[:50])
            parent2 = random.choice(population[:50])
            child = parent1.mate(parent2)
            new_generation.append(child)
        population = new_generation
        print("Generation: {}\tString: {}\tFitness:
{}".format(generation,"".join(population[0].chromosome), population[0].fitness))
        generation += 1
    print("Generation: {}\tString: {}\tFitness: {}".format(generation, "".join(population[0].chromosome),
population[0].fitness))

if_name_== '_main_':
    main()
```

Output:



```
Executed by Sumitha Naidu
Roll No. : 11
Current Date and Time : 17-11-2022 14:15:52

Generation: 1   String: L][]P8 GthYa?a%Dr[co    Fitness: 18
Generation: 2   String: d_lR.eOwm {CZVYGGss     Fitness: 17
Generation: 3   String: d_lR.eOwm {CZVYGGss     Fitness: 17
Generation: 4   String: U lKp_Y6ee9S0,NHaP}j    Fitness: 16
Generation: 5   String: I]lchi LeKA35o4$G)=6    Fitness: 15
Generation: 6   String: I]lchi LeKA35o4$G)=6    Fitness: 15
Generation: 7   String: uElop_ Gee/32&#GaHAb    Fitness: 13
Generation: 8   String: uElop_ Gee/32&#GaHAb    Fitness: 13
Generation: 9   String: I]lov/ GeK:j6,#Gm4@Z    Fitness: 12
Generation: 10  String: I]lov/ GeK:j6,#Gm4@Z    Fitness: 12
Generation: 11  String: I lov& G3x/!5oJGmUN,    Fitness: 11
Generation: 12  String: I Pov8 JeenBHo#Ge:9&    Fitness: 10
Generation: 13  String: I Pov8 JeenBHo#Ge:9&    Fitness: 10
Generation: 14  String: I Pjve Gee9B4oQGe:9,    Fitness: 9
Generation: 15  String: I Pjve Gee9B4oQGe:9,    Fitness: 9
Generation: 16  String: I love Gehesso2GedAb    Fitness: 7
Generation: 17  String: I love Gehesso2GedAb    Fitness: 7
Generation: 18  String: I love Gehesso2GedAb    Fitness: 7
Generation: 19  String: I love Gehesso2GedAb    Fitness: 7
Generation: 20  String: I love Gee6s4o5Ge:k#    Fitness: 5
Generation: 21  String: I love Gee6s4o5Ge:k#    Fitness: 5
Generation: 22  String: I love Gee6s4o5Ge:k#    Fitness: 5
Generation: 23  String: I love Gee6s4o5Ge:k#    Fitness: 5
```



```
Generation: 59  String: I love Gee&sforGeeks    Fitness: 1
Generation: 60  String: I love Gee&sforGeeks    Fitness: 1
Generation: 61  String: I love Gee&sforGeeks    Fitness: 1
Generation: 62  String: I love Gee&sforGeeks    Fitness: 1
Generation: 63  String: I love Gee&sforGeeks    Fitness: 1
Generation: 64  String: I love Gee&sforGeeks    Fitness: 1
Generation: 65  String: I love Gee&sforGeeks    Fitness: 1
Generation: 66  String: I love Gee&sforGeeks    Fitness: 1
Generation: 67  String: I love Gee&sforGeeks    Fitness: 1
Generation: 68  String: I love Gee&sforGeeks    Fitness: 1
Generation: 69  String: I love Gee&sforGeeks    Fitness: 1
Generation: 70  String: I love Gee&sforGeeks    Fitness: 1
Generation: 71  String: I love Gee&sforGeeks    Fitness: 1
Generation: 72  String: I love Gee&sforGeeks    Fitness: 1
Generation: 73  String: I love Gee&sforGeeks    Fitness: 1
Generation: 74  String: I love Gee&sforGeeks    Fitness: 1
Generation: 75  String: I love Gee&sforGeeks    Fitness: 1
Generation: 76  String: I love Gee&sforGeeks    Fitness: 1
Generation: 77  String: I love Gee&sforGeeks    Fitness: 1
Generation: 78  String: I love Gee&sforGeeks    Fitness: 1
Generation: 79  String: I love Gee&sforGeeks    Fitness: 1
Generation: 80  String: I love Gee&sforGeeks    Fitness: 1
Generation: 81  String: I love Gee&sforGeeks    Fitness: 1
Generation: 82  String: I love Gee&sforGeeks    Fitness: 1
Generation: 83  String: I love Gee&sforGeeks    Fitness: 1
Generation: 84  String: I love Gee&sforGeeks    Fitness: 1
Generation: 85  String: I love Gee&sforGeeks    Fitness: 1
Generation: 86  String: I love Gee&sforGeeks    Fitness: 1
```

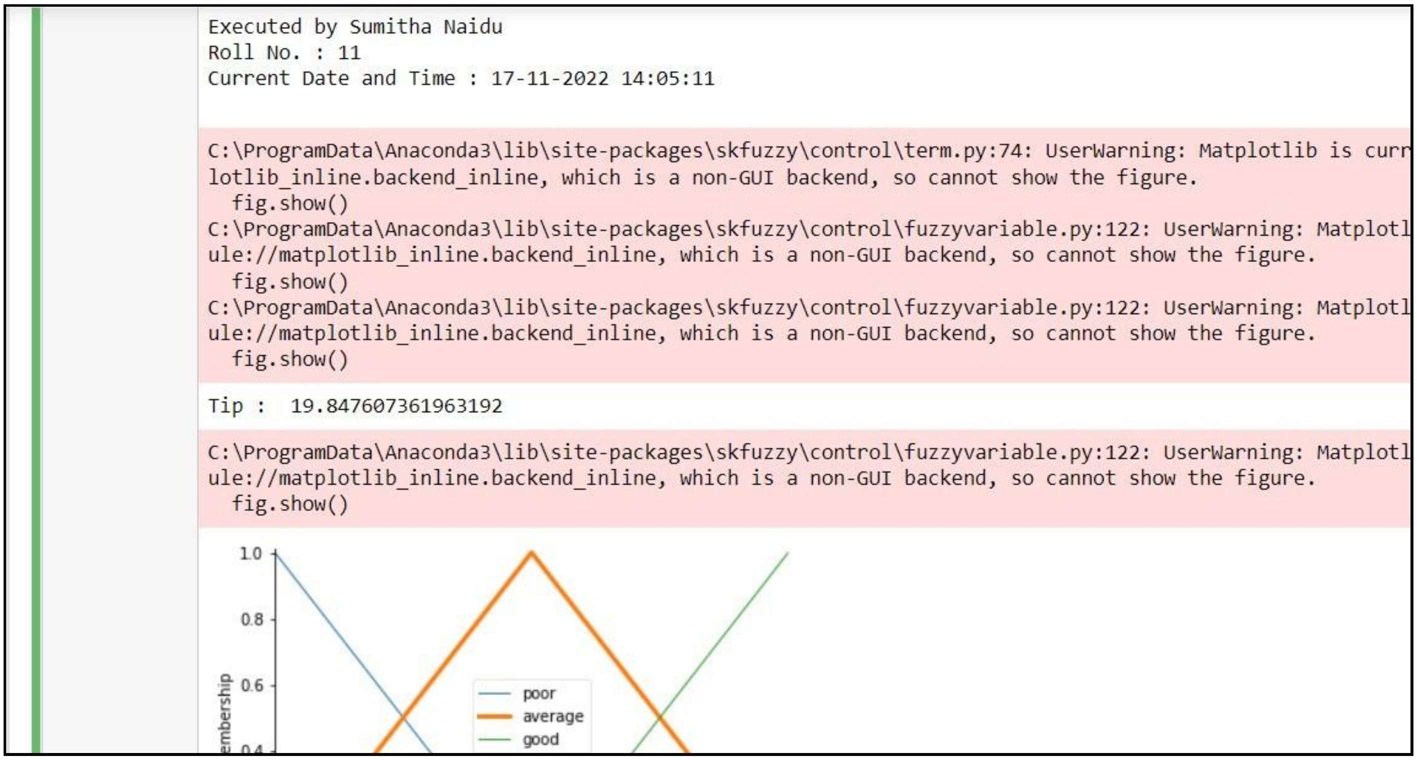
Practical No. 5

Aim: Design a Fuzzy based application using Python.

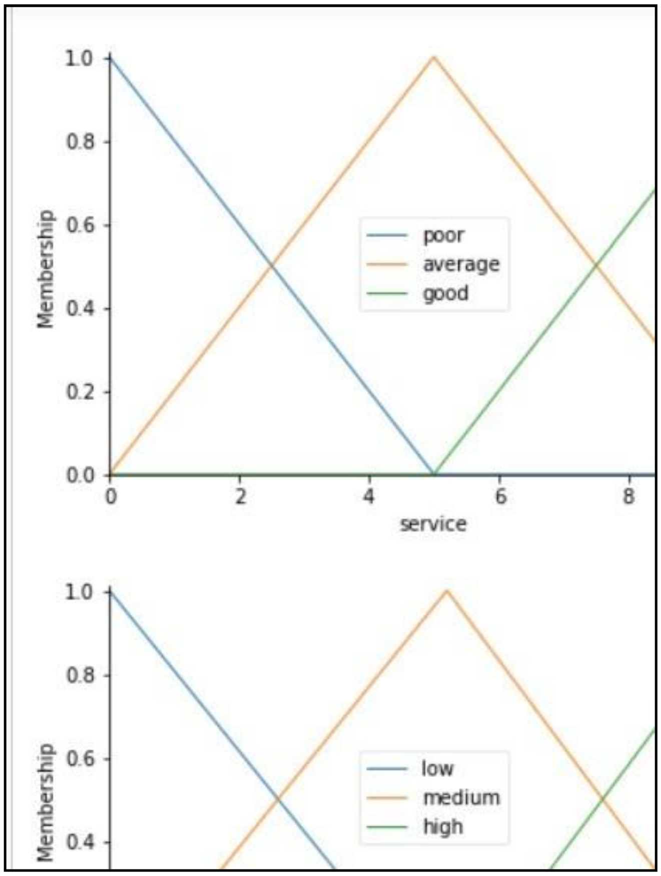
Code:

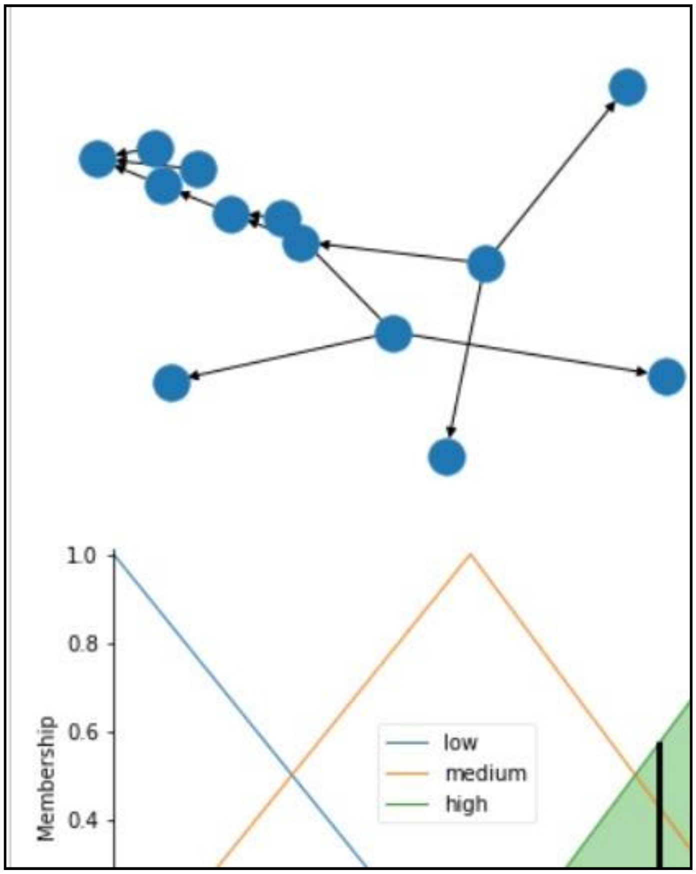
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and membership functions
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')

# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)

# Custom membership functions can be built interactively with a familiar, Pythonic API
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])

# You can see how these look with .view()
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()

tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
# Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
# Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8
# Crunch the numbers
tipping.compute()
print (tipping.output['tip'])
tip.view(sim=tipping)
```

Output:



```
Executed by Sumitha Naidu
Roll No. : 11
Current Date and Time : 17-11-2022 14:05:11

C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\term.py:74: UserWarning: Matplotlib is curr
lotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotl
ule://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotl
ule://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()

Tip :  19.847607361963192

C:\ProgramData\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotl
ule://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```





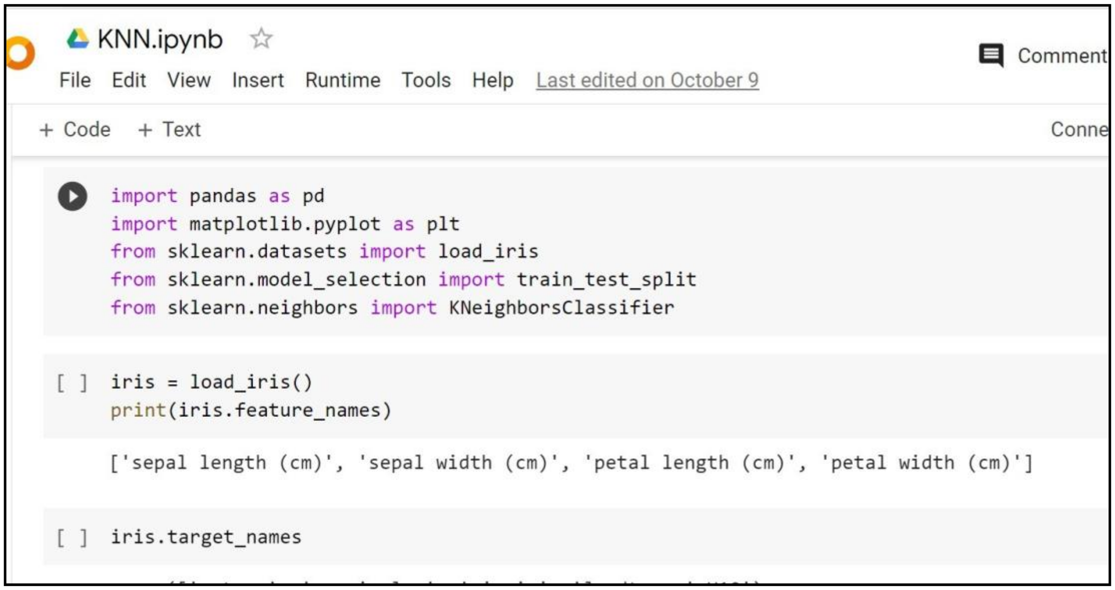
Practical No. 6

Aim: Write an application to implement supervised and unsupervised learning model.

Code:

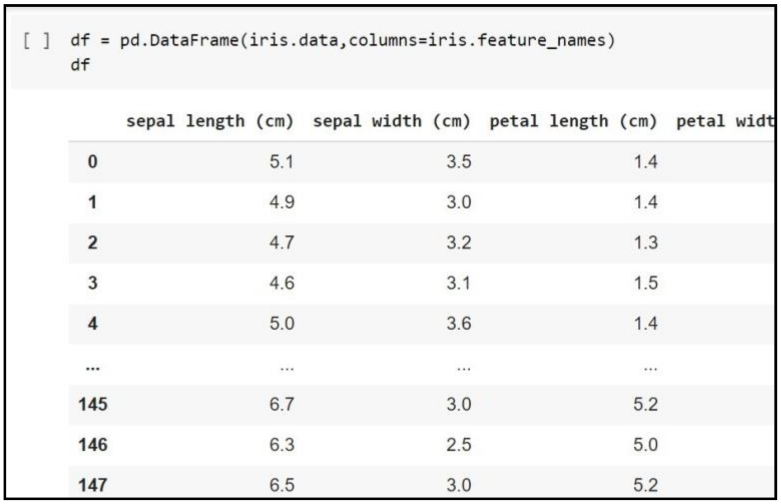
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()
print(iris.feature_names)
iris.target_names
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df['target'] = iris.target
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
print(df)

df0 =  df[:50]
df1 = df[50:100]
df2 = df[100:]

plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')

plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",s=100,marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')

X = df.drop(['target','flower_name'], axis='columns')
y = df.target
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn.score(X_test, y_test)
```

Output:


KNN.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 9
Comment
+ Code + Text
Conne

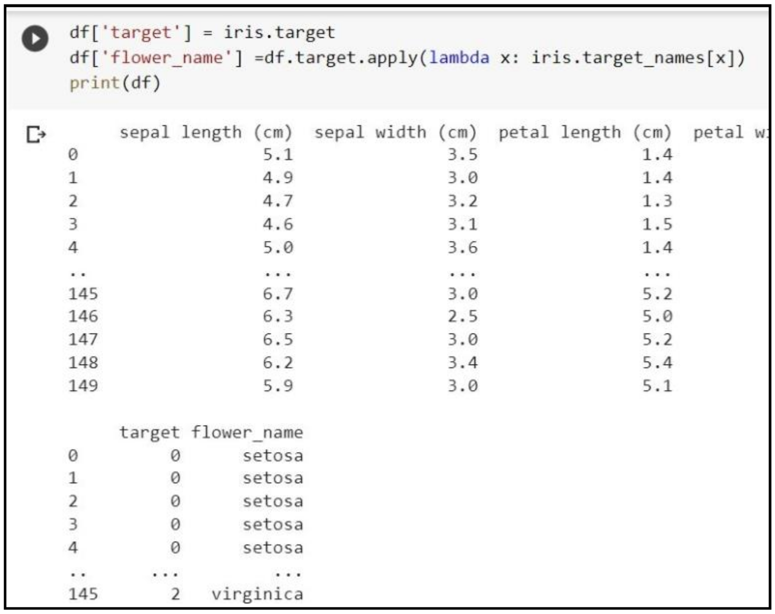
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
[ ] iris = load_iris()
    print(iris.feature_names)

    ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

[ ] iris.target_names
```



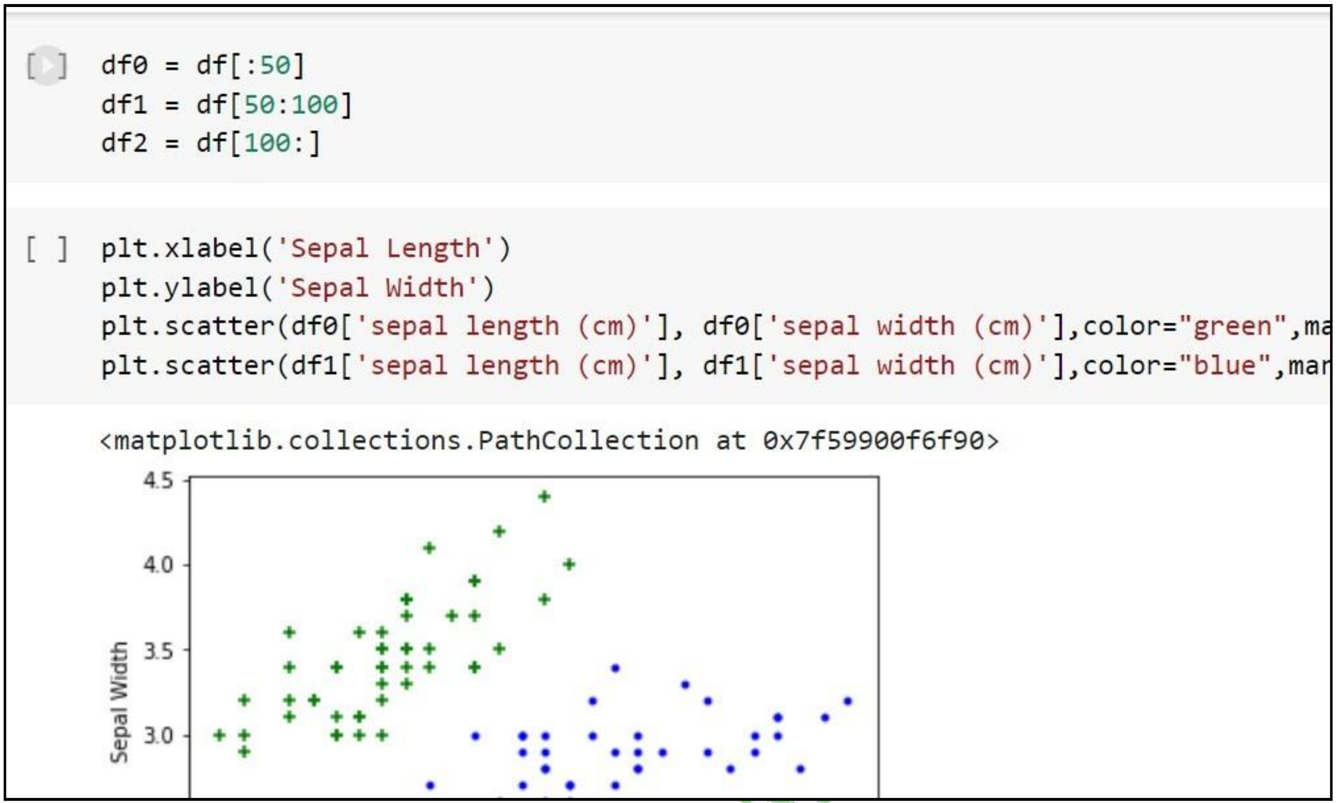
```
[ ] df = pd.DataFrame(iris.data,columns=iris.feature_names)
    df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal widt
0	5.1	3.5	1.4	
1	4.9	3.0	1.4	
2	4.7	3.2	1.3	
3	4.6	3.1	1.5	
4	5.0	3.6	1.4	
...	...	...	...	
145	6.7	3.0	5.2	
146	6.3	2.5	5.0	
147	6.5	3.0	5.2	



```
df['target'] = iris.target
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
print(df)
```

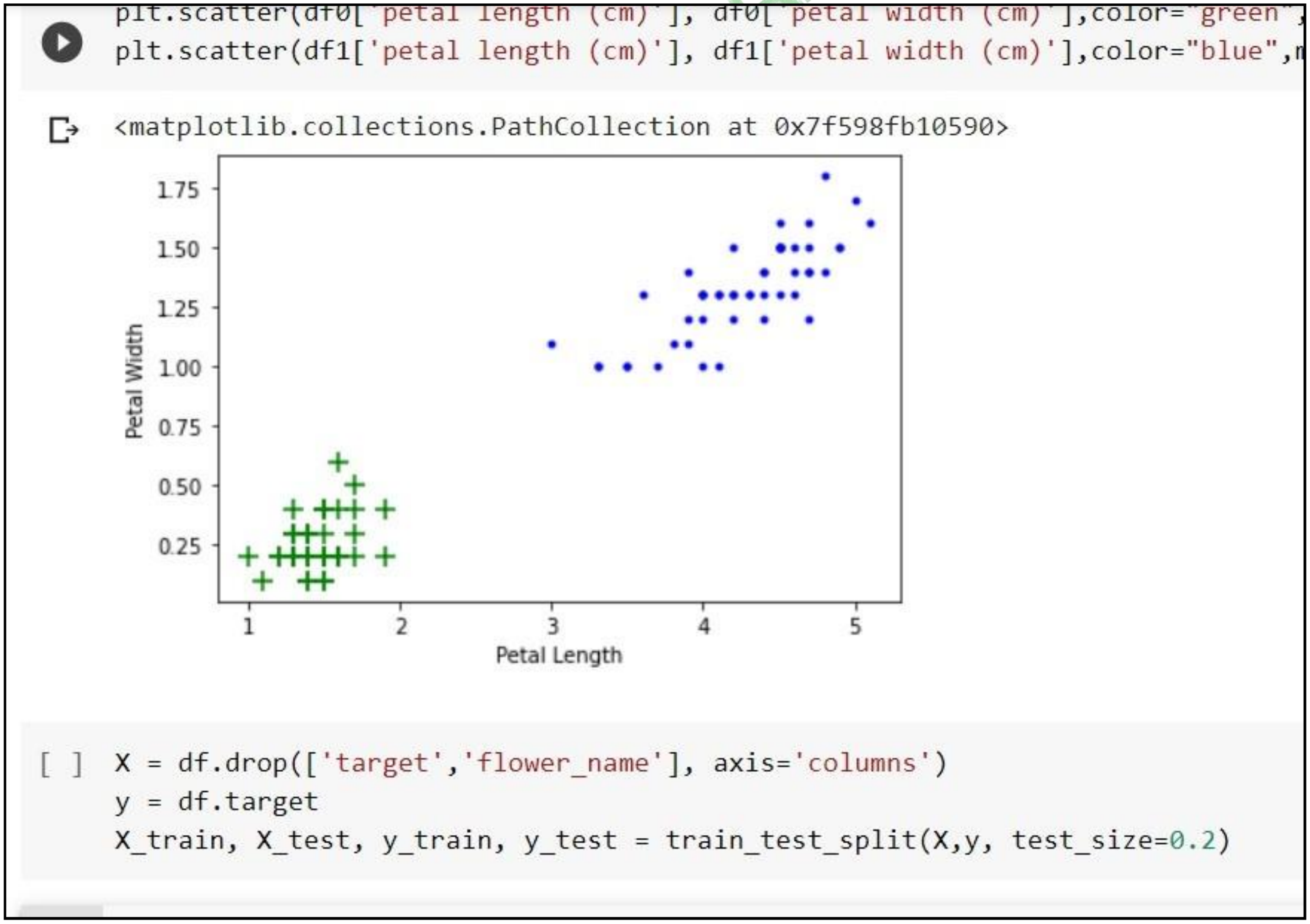
```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal wi
0                  5.1               3.5                1.4
1                  4.9               3.0                1.4
2                  4.7               3.2                1.3
3                  4.6               3.1                1.5
4                  5.0               3.6                1.4
..                 ...               ...                ...
145                6.7               3.0                5.2
146                6.3               2.5                5.0
147                6.5               3.0                5.2
148                6.2               3.4                5.4
149                5.9               3.0                5.1

     target flower_name
0         0      setosa
1         0      setosa
2         0      setosa
3         0      setosa
4         0      setosa
..      ...         ...
145       2   virginica
```



```
[ ] df0 = df[:50]
    df1 = df[50:100]
    df2 = df[100:]
```

```
[ ] plt.xlabel('Sepal Length')
    plt.ylabel('Sepal Width')
    plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",ma
    plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",mar
```

```
<matplotlib.collections.PathCollection at 0x7f59900f6f90>
```



```
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",m
```

```
<matplotlib.collections.PathCollection at 0x7f598fb10590>
```

```
[ ] X = df.drop(['target','flower_name'], axis='columns')
    y = df.target
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)
```

Practical No. 7

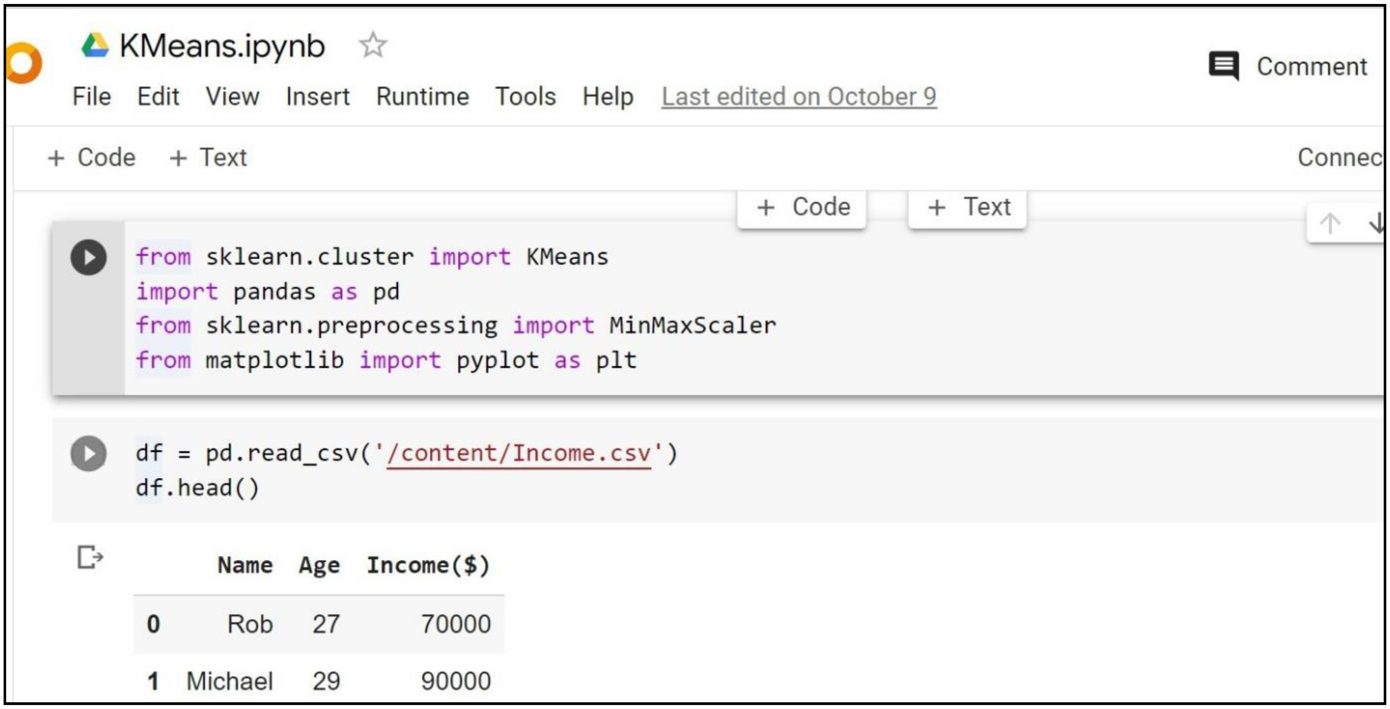
Aim: Write an application to implement clustering algorithm (K Means).

Code:

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
df = pd.read_csv('/content/Income.csv')
df.head()
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age', 'Income($)']])
df['cluster'] = predicted
df.head()
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
df3 = df[df.cluster == 2]
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='purple', marker='*', label='Centroid')
plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

df.head()
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age', 'Income($)']])
df['cluster'] = predicted
df.head()
```

```
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
df3 = df[df.cluster == 2]
plt.scatter(df1['Age'], df1['Income($)'], color='green')
plt.scatter(df2['Age'], df2['Income($)'], color='red')
plt.scatter(df3['Age'], df3['Income($)'], color='blue')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='purple', marker='*', label='Centroid')
plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
#Elbow Plot (For checking)
sse = []
k_range = range(1,10)
for k in k_range:
  km = KMeans(n_clusters=k)
  km.fit(df[['Age', 'Income($)']])
  sse.append(km.inertia_) # Calculating the distance between centroids and the nearest point

plt.xlabel('K')
plt.ylabel('Sum of Squared error')
plt.plot(k_range,sse)
```

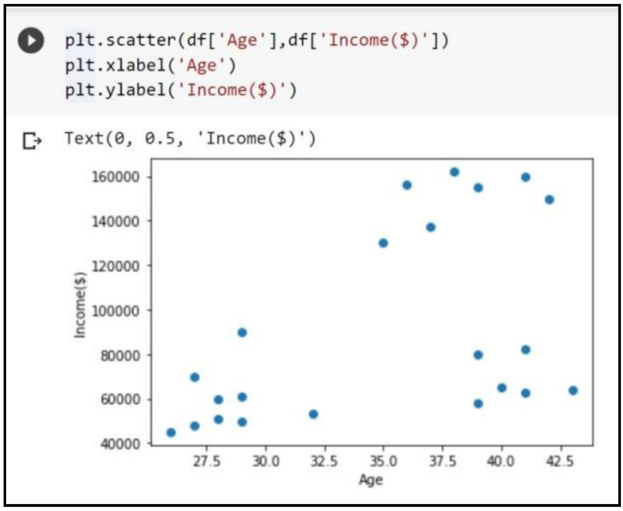
Output:


KMeans.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 9
Comment
+ Code + Text
Connec
+ Code + Text

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
```

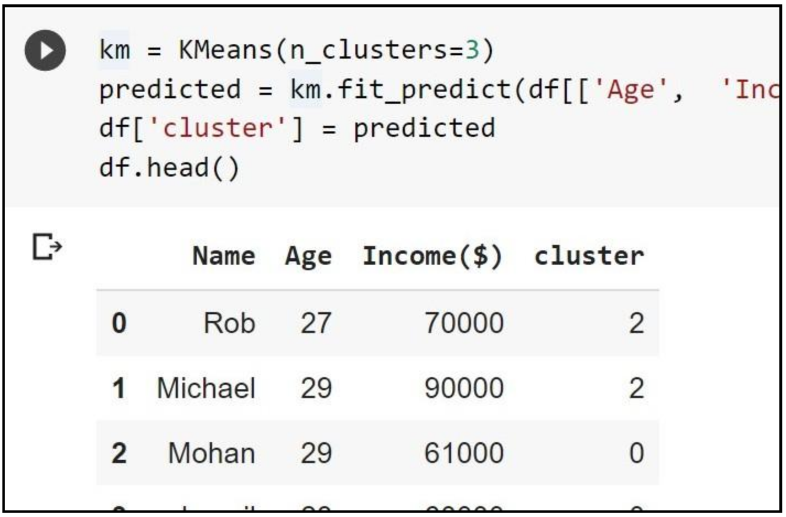
```
df = pd.read_csv('/content/Income.csv')
df.head()
```

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000



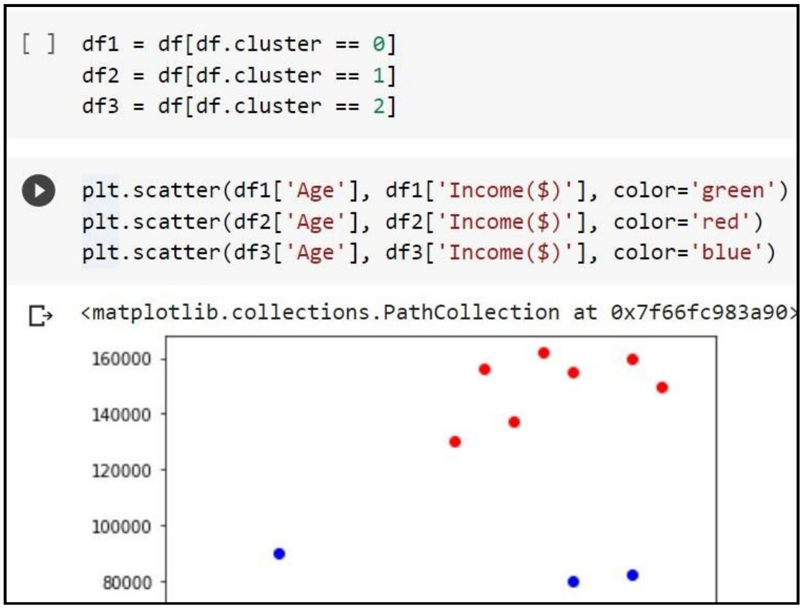
```
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
```

```
Text(0, 0.5, 'Income($)')
```



```
km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age',  'Inc
df['cluster'] = predicted
df.head()
```

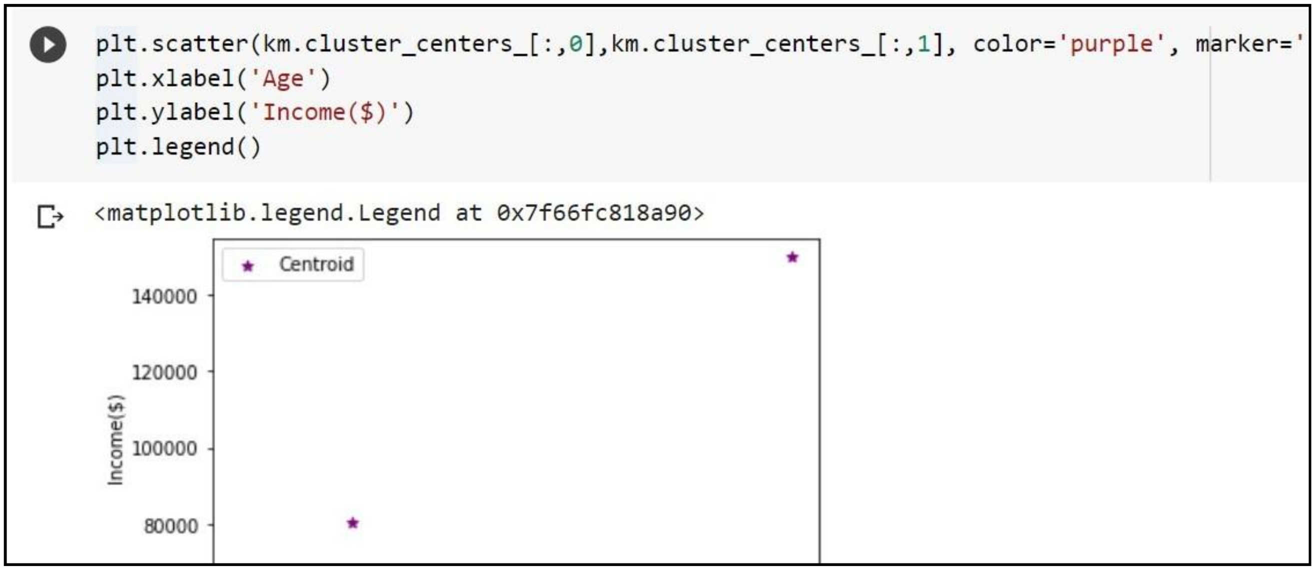
	Name	Age	Income($)	cluster
0	Rob	27	70000	2
1	Michael	29	90000	2
2	Mohan	29	61000	0



```
[ ] df1 = df[df.cluster == 0]
    df2 = df[df.cluster == 1]
    df3 = df[df.cluster == 2]
```

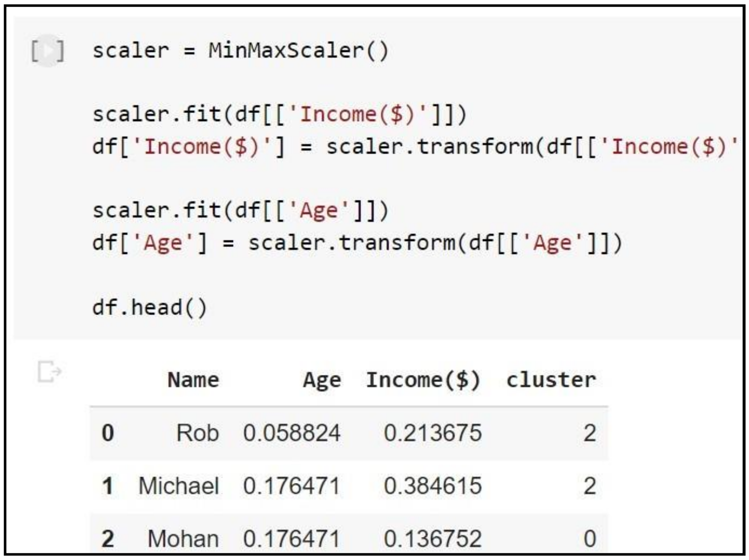
```
plt.scatter(df1['Age'], df1['Income($)'], color='green')
plt.scatter(df2['Age'], df2['Income($)'], color='red')
plt.scatter(df3['Age'], df3['Income($)'], color='blue')
```

```
<matplotlib.collections.PathCollection at 0x7f66fc983a90>
```



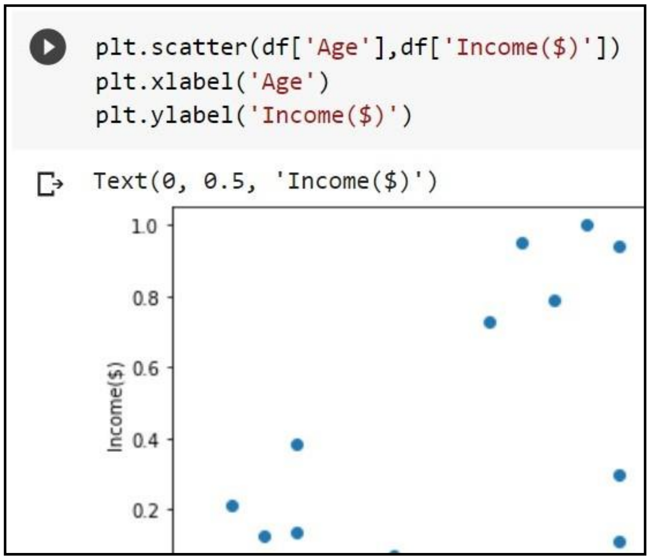
```
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='purple', marker='
plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f66fc818a90>
```



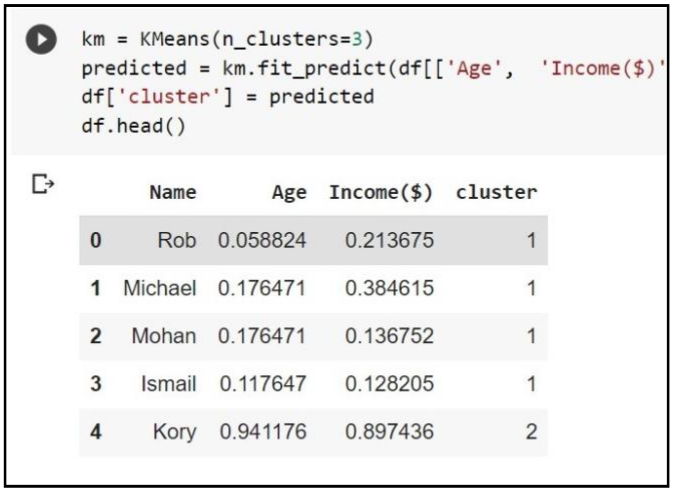
```
[ ] scaler = MinMaxScaler()

    scaler.fit(df[['Income($)']])
    df['Income($)'] = scaler.transform(df[['Income($)'

    scaler.fit(df[['Age']])
    df['Age'] = scaler.transform(df[['Age']])

    df.head()
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	2
1	Michael	0.176471	0.384615	2
2	Mohan	0.176471	0.136752	0



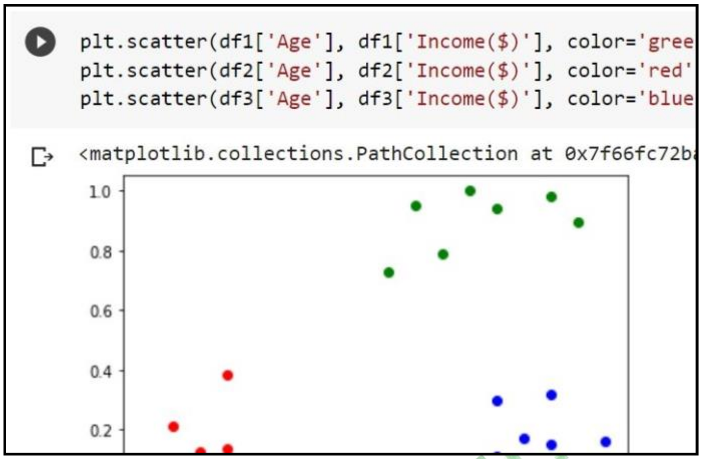
```
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
```

```
Text(0, 0.5, 'Income($)')
```



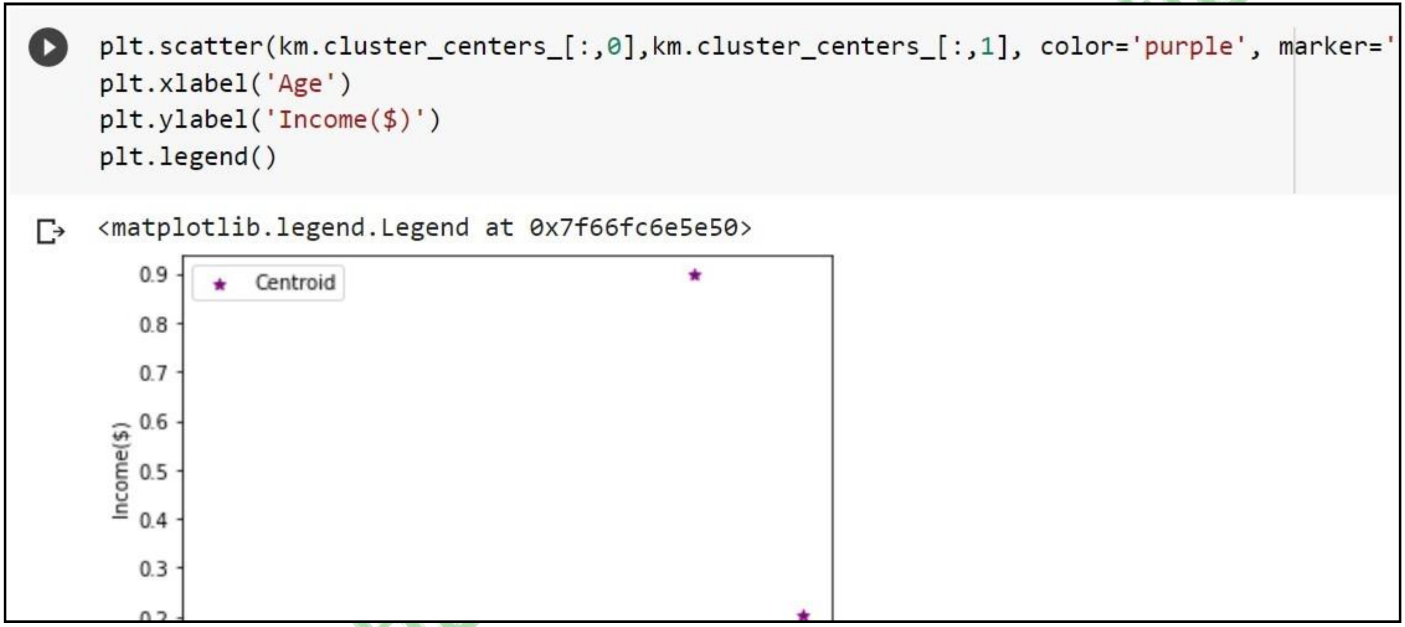
```
km = KMeans(n_clusters=3)
predicted = km.fit_predict(df[['Age',  'Income($)'
df['cluster'] = predicted
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	1
1	Michael	0.176471	0.384615	1
2	Mohan	0.176471	0.136752	1
3	Ismail	0.117647	0.128205	1
4	Kory	0.941176	0.897436	2



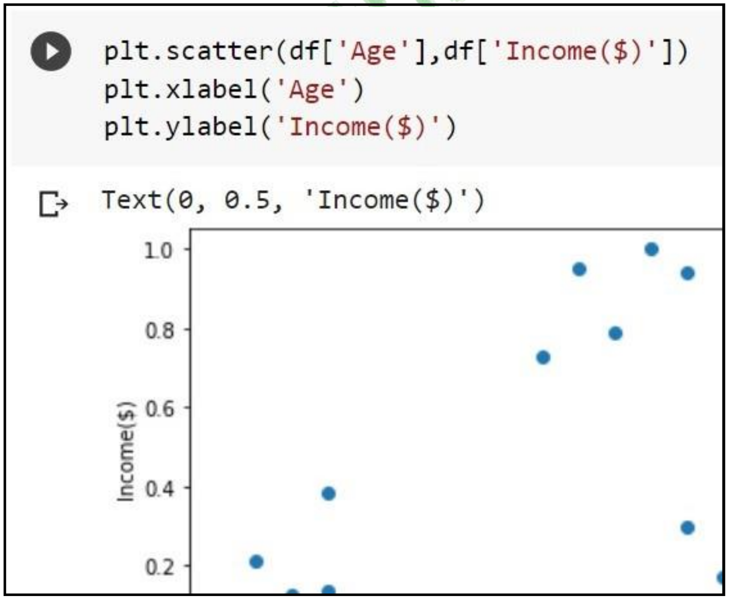
```
plt.scatter(df1['Age'], df1['Income($)'], color='gree
plt.scatter(df2['Age'], df2['Income($)'], color='red'
plt.scatter(df3['Age'], df3['Income($)'], color='blue
```

```
<matplotlib.collections.PathCollection at 0x7f66fc72b
```



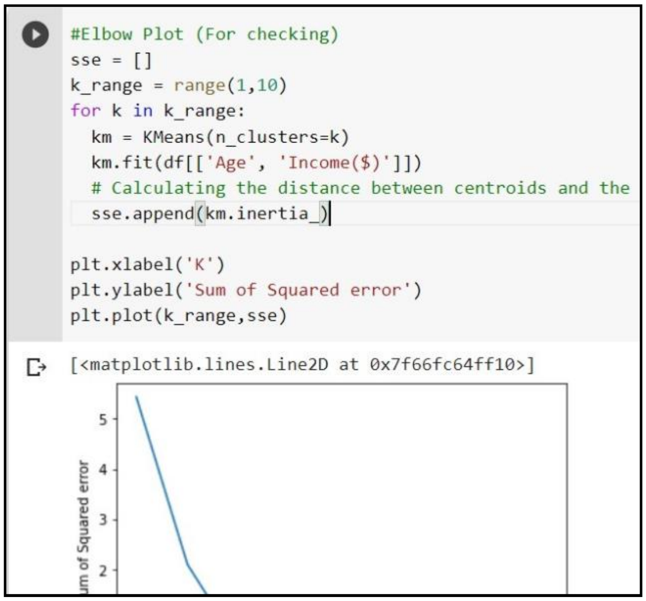
```
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='purple', marker='
plt.xlabel('Age')
plt.ylabel('Income($)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f66fc6e5e50>
```



```
plt.scatter(df['Age'],df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
```

```
Text(0, 0.5, 'Income($)')
```



```
#Elbow Plot (For checking)
sse = []
k_range = range(1,10)
for k in k_range:
  km = KMeans(n_clusters=k)
  km.fit(df[['Age', 'Income($)']])
  # Calculating the distance between centroids and the
  sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of Squared error')
plt.plot(k_range,sse)
```

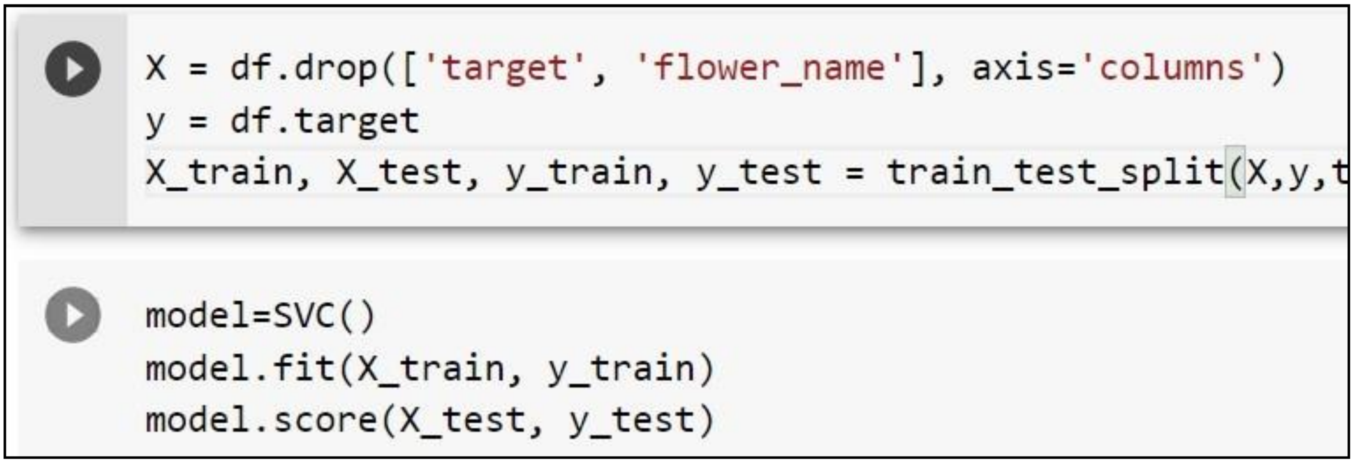
```
[<matplotlib.lines.Line2D at 0x7f66fc64ff10>]
```

Practical No. 8

Aim: Write an application to implement support vector machine algorithm.

Code:

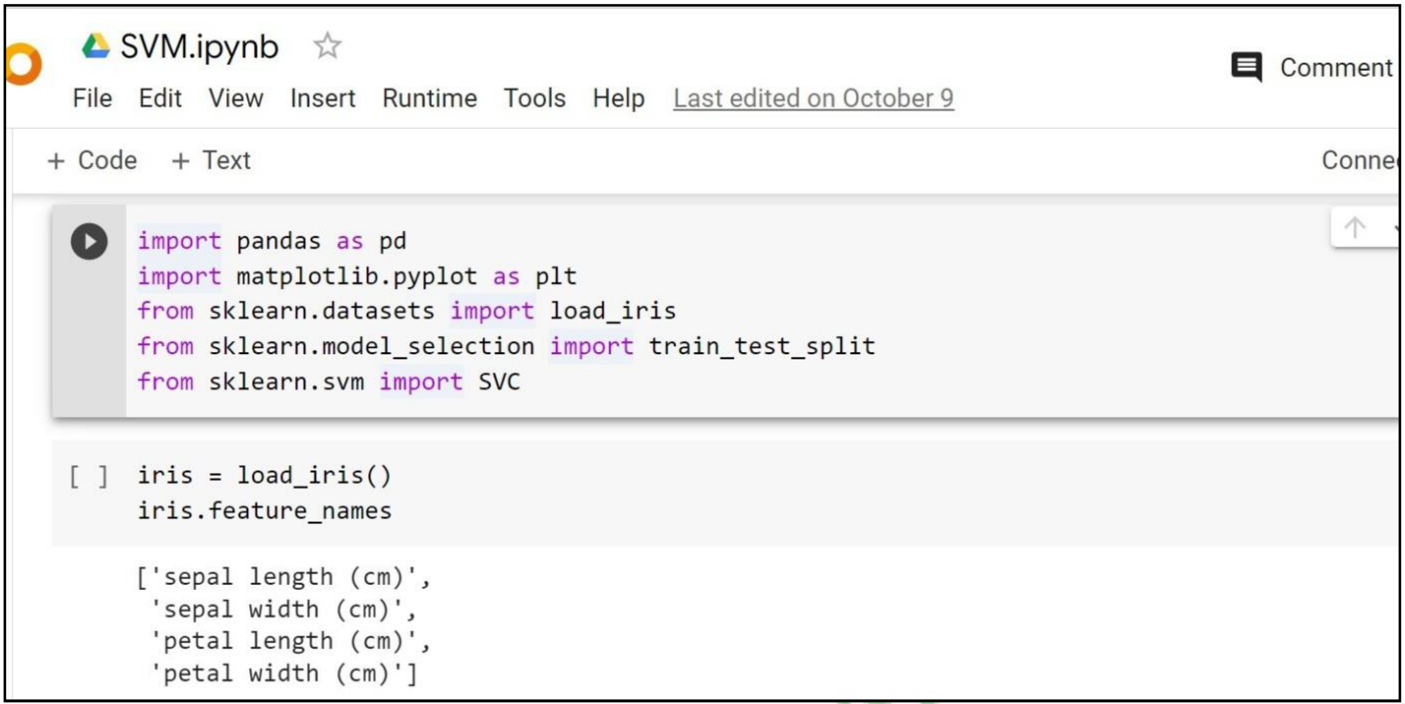
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

iris = load_iris()
iris.feature_names
iris.target_names
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
df['flower_name'] = df.target.apply(lambda x : iris.target_names[x])
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:150]
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'], color='green', marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'], color='red', marker='.')

X = df.drop(['target', 'flower_name'], axis='columns')
y = df.target
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
model=SVC()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```



```
X = df.drop(['target', 'flower_name'], axis='columns')
y = df.target
X_train, X_test, y_train, y_test = train_test_split(X,y,t
```

```
model=SVC()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

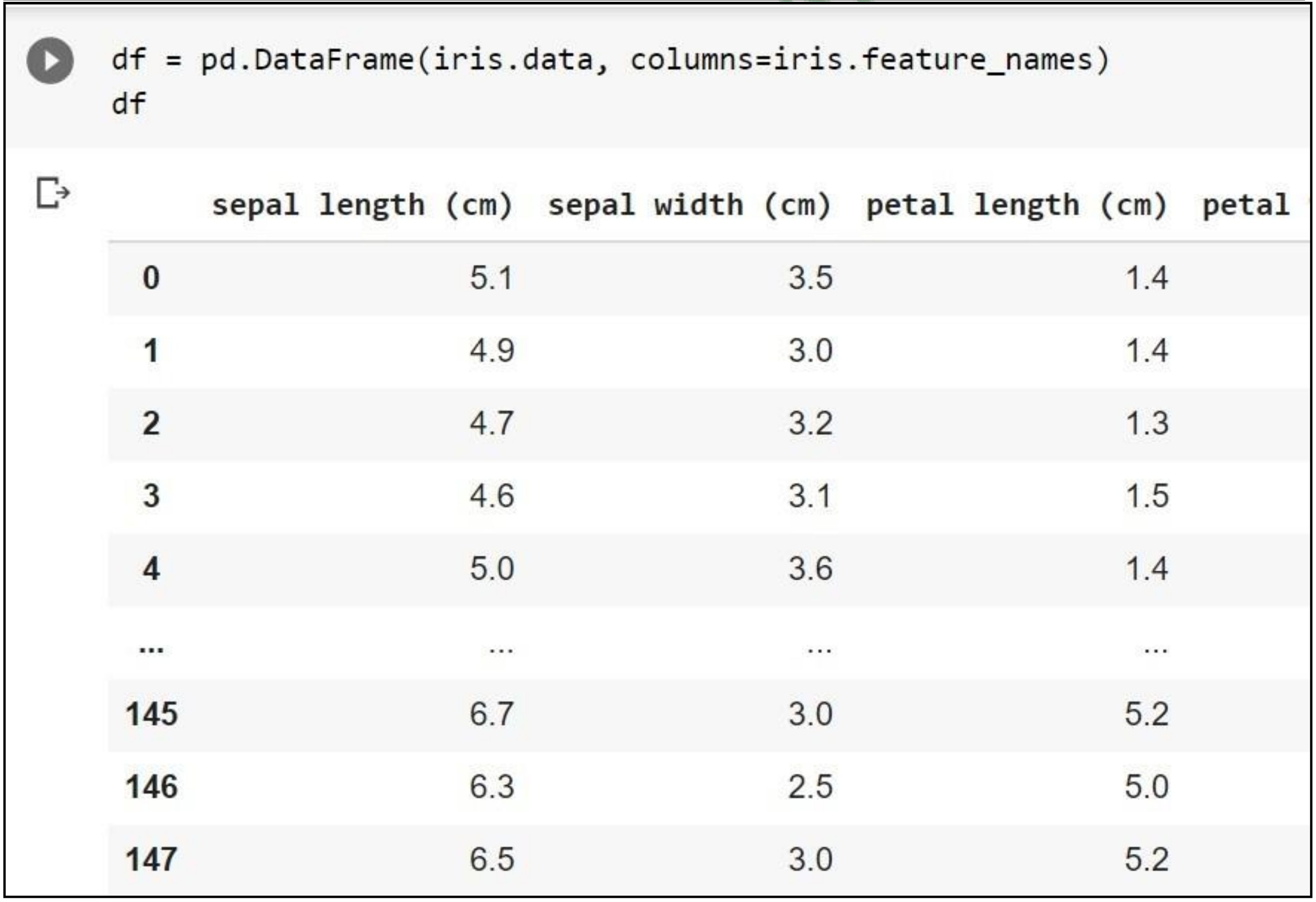
Output:


SVM.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 9
Comment
+ Code + Text
Conne

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

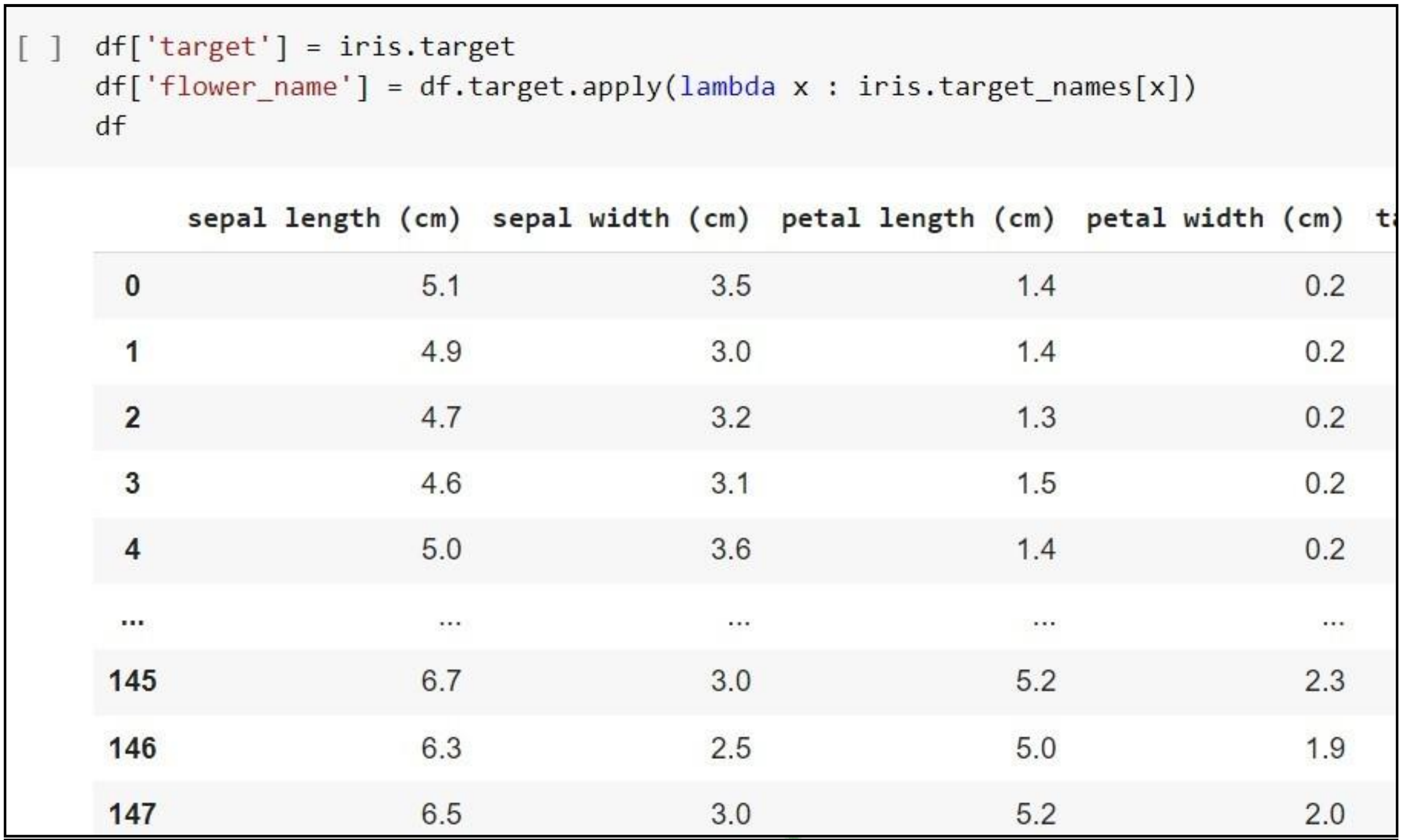
```
[ ] iris = load_iris()
    iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```



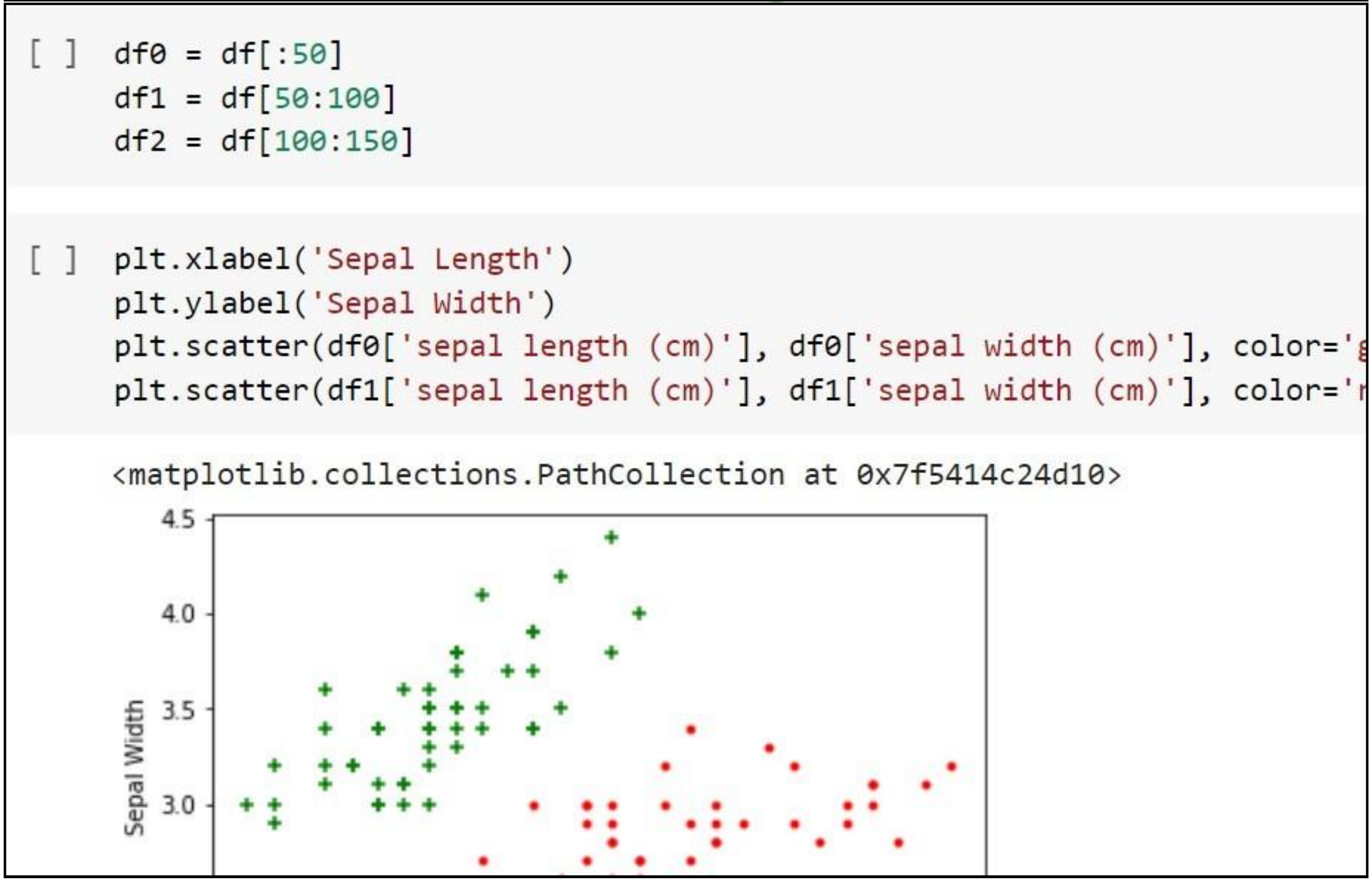
```
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal
0	5.1	3.5	1.4	
1	4.9	3.0	1.4	
2	4.7	3.2	1.3	
3	4.6	3.1	1.5	
4	5.0	3.6	1.4	
...	...	...	...	
145	6.7	3.0	5.2	
146	6.3	2.5	5.0	
147	6.5	3.0	5.2	



```
[ ]  df['target'] = iris.target
     df['flower_name'] = df.target.apply(lambda x : iris.target_names[x])
     df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	t
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
...	...	...	...	...	
145	6.7	3.0	5.2	2.3	
146	6.3	2.5	5.0	1.9	
147	6.5	3.0	5.2	2.0	



```
[ ]  df0 = df[:50]
     df1 = df[50:100]
     df2 = df[100:150]
```

```
[ ]  plt.xlabel('Sepal Length')
     plt.ylabel('Sepal Width')
     plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'], color='g
     plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'], color='r
```

```
<matplotlib.collections.PathCollection at 0x7f5414c24d10>
```

Practical No. 9

Aim: Design a bot using AIML.

Code with Output:

Install the following packages

- pip install aiml
- pip install python-aiml
- pip3 install aiml
- pip3 install python-aiml

# sillybot.py

```
import aiml

kernel = aiml.Kernel()
kernel.learn("std-startup.xml")
kernel.respond("load aiml b")
while True:
   inputText = input(" > Human : ")
   response = kernel.respond(inputText)
   print(" > Bot : "+response)
```

# std-startup.xml

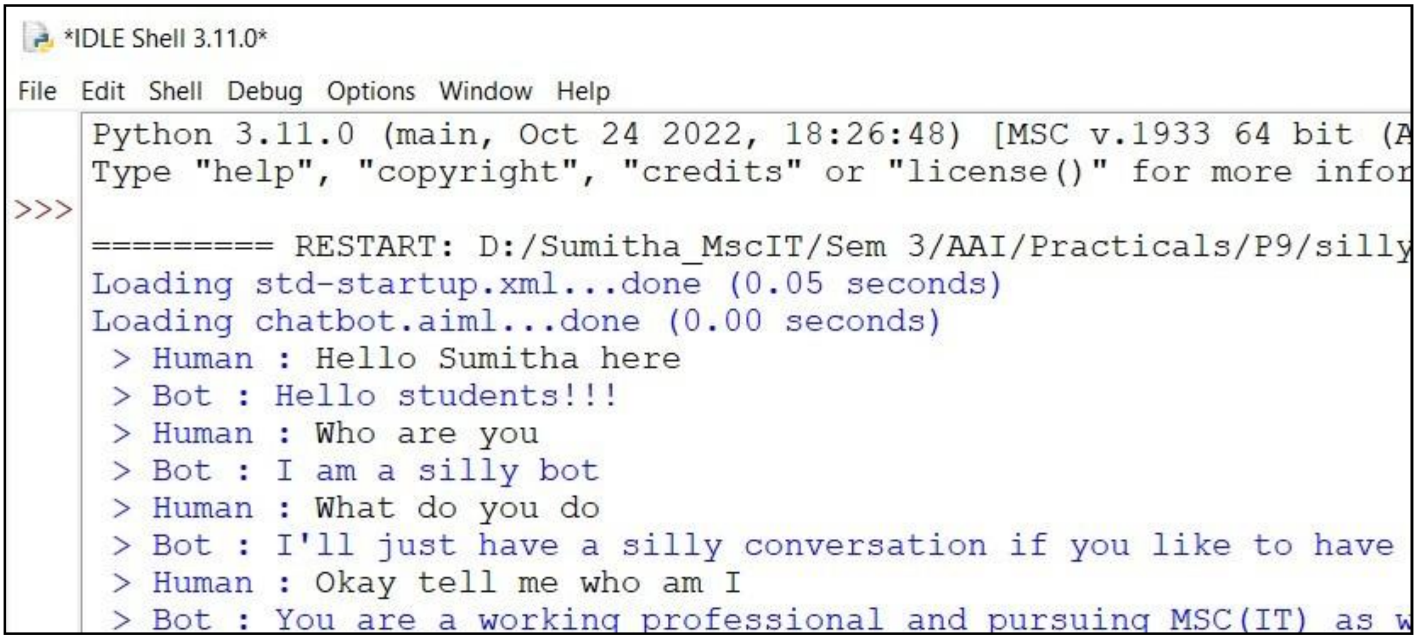
```
<aiml encoding="UTF-8" version="1.0.1">
   <category>
      <pattern>LOAD AIML B</pattern>
      <template>
         <learn>chatbot.aiml</learn>
      </template>
   </category>
</aiml>
```

# chatbot.aiml

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

<category>
```

```
<pattern>WHO ARE YOU</pattern>
<template>I am a silly bot</template>
</category>

<category>
<pattern>WHAT DO YOU DO</pattern>
<template>I'll just have a silly conversation if you like to have</template>
</category>

<category>
<pattern>OKAY TELL ME WHO AM I</pattern>
<template>You are a working professional and pursuing MSC(IT) as well</template>
</category>

<category>
<pattern>WELL BYE SEE YOU AGAIN</pattern>
<template>Bye, Take Care!!</template>
</category>
</aiml>
```

Output:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (A
Type "help", "copyright", "credits" or "license()" for more infor
>>>
========= RESTART: D:/Sumitha_MscIT/Sem 3/AAI/Practicals/P9/silly
Loading std-startup.xml...done (0.05 seconds)
Loading chatbot.aiml...done (0.00 seconds)
 > Human : Hello Sumitha here
 > Bot : Hello students!!!
 > Human : Who are you
 > Bot : I am a silly bot
 > Human : What do you do
 > Bot : I'll just have a silly conversation if you like to have
 > Human : Okay tell me who am I
 > Bot : You are a working professional and pursuing MSC(IT) as w
```

Practical No. 10

Aim: Design an Expert System using AIML.

Code :

Install the following packages

- pip install aiml
- pip install python-aiml
- pip3 install aiml
- pip3 install python-aiml

# doctorchat.py

```
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)
```

# std-startup.xml

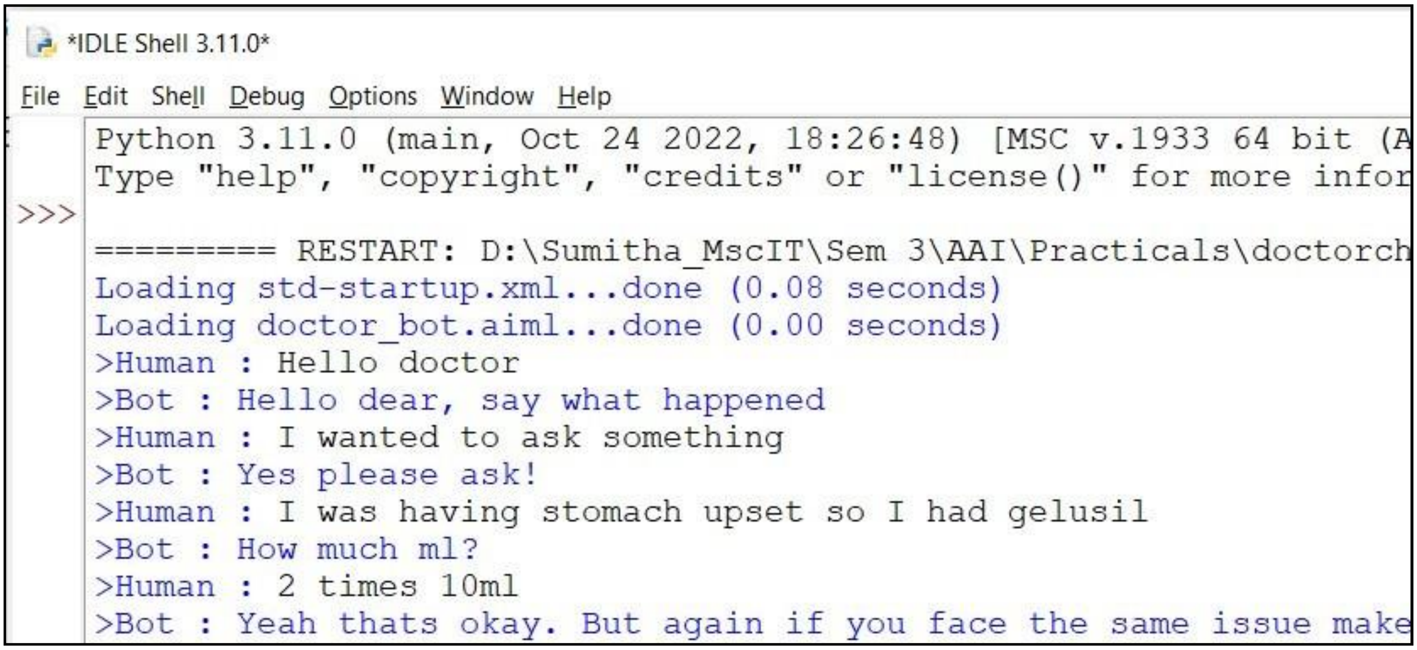
```
<aiml encoding="UTF-8" version="1.0.1">
<category>
<pattern>LOAD AIML B</pattern>
<template>
<learn>doctor_bot.aiml</learn>
</template>
</category>
</aiml>
```

# doctor_bot.aiml

```
<aiml version="1.0.1" encoding="UTF-8">
<category>
<pattern>HELLO DOCTOR</pattern>
<template>Hello dear, say what happened</template>
</category>

<category>
<pattern>I WANTED TO ASK SOME SOMETHING</pattern>
```

```
<template>Yes please ask!</template>
</category>

<category>
<pattern>I was having stomach upset so I had gelusil</pattern>
<template>How much ml?</template>
</category>

<category>
<pattern>2 times 10ml</pattern>
<template>Yeah thats okay. But again if you face the same issue make sure you consult
doctor.</template>
</category>

<category>
<pattern>OKAY DOCTOR THANK YOU</pattern>
<template>WELL. Take Care</template>
</category>
</aiml>
```

Output:



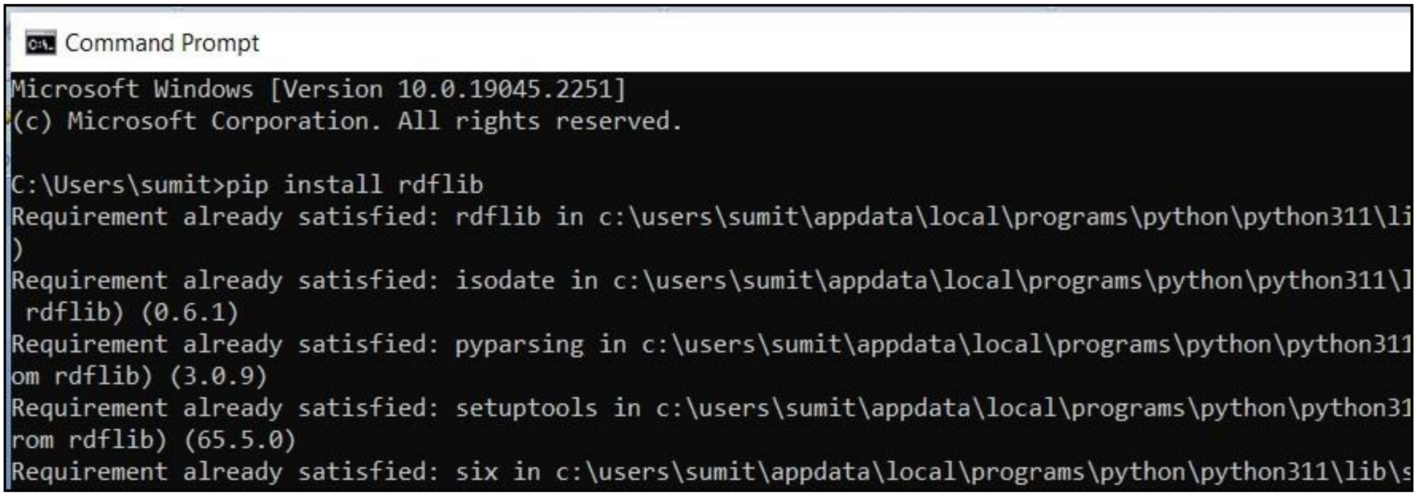
```
    Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (A
    Type "help", "copyright", "credits" or "license()" for more infor
>>>
    ========= RESTART: D:\Sumitha_MscIT\Sem 3\AAI\Practicals\doctorch
    Loading std-startup.xml...done (0.08 seconds)
    Loading doctor_bot.aiml...done (0.00 seconds)
    >Human : Hello doctor
    >Bot : Hello dear, say what happened
    >Human : I wanted to ask something
    >Bot : Yes please ask!
    >Human : I was having stomach upset so I had gelusil
    >Bot : How much ml?
    >Human : 2 times 10ml
    >Bot : Yeah thats okay. But again if you face the same issue make
```

Practical No. 11

Aim: Design an application to simulate Semantic Web.

Code :

Install the following package



```
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sumit>pip install rdflib
Requirement already satisfied: rdflib in c:\users\sumit\appdata\local\programs\python\python311\li
)
Requirement already satisfied: isodate in c:\users\sumit\appdata\local\programs\python\python311\l
 rdflib) (0.6.1)
Requirement already satisfied: pyparsing in c:\users\sumit\appdata\local\programs\python\python311
om rdflib) (3.0.9)
Requirement already satisfied: setuptools in c:\users\sumit\appdata\local\programs\python\python31
rom rdflib) (65.5.0)
Requirement already satisfied: six in c:\users\sumit\appdata\local\programs\python\python311\lib\s
```

# websemantic.py

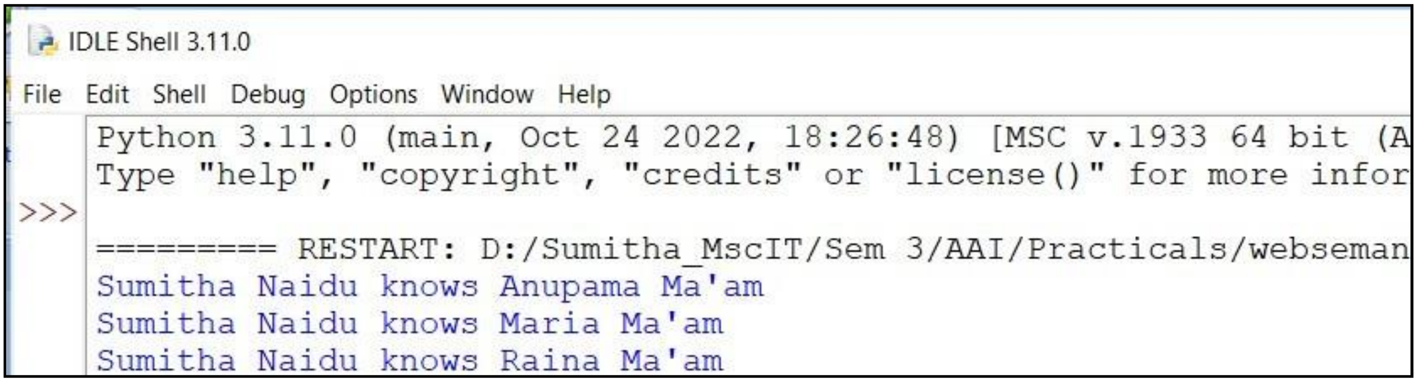
```
import rdflib

myGraph = rdflib.Graph()
myGraph.parse("myfoaf.rdf")
qres=myGraph.query(
"""SELECT DISTINCT ?fname ?lname
WHERE{
?a foaf:knows ?b .
?a foaf:name ?fname .
?b foaf:name ?lname .
}""")
for row in qres:
    print("%s knows %s"%row)
```

# myfoaf.rdf

```
 <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:admin="http://webns.net/mvcb/">
```

```
    <foaf:Person rdf:nodeID="me">
        <foaf:name>Sumitha Naidu</foaf:name>
        <foaf:knows>
            <foaf:Person>
                <foaf:name>Anupama Ma'am</foaf:name>
            </foaf:Person>
        </foaf:knows>
        <foaf:knows>
            <foaf:Person>
                <foaf:name>Maria Ma'am</foaf:name>
            </foaf:Person>
        </foaf:knows>
        <foaf:knows>
            <foaf:Person>
                <foaf:name>Raina Ma'am</foaf:name>
            </foaf:Person>
        </foaf:knows>
    </foaf:Person>

</rdf:RDF>
```

Output:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (A
Type "help", "copyright", "credits" or "license()" for more infor
>>>
========= RESTART: D:/Sumitha_MscIT/Sem 3/AAI/Practicals/webseman
Sumitha Naidu knows Anupama Ma'am
Sumitha Naidu knows Maria Ma'am
Sumitha Naidu knows Raina Ma'am
```

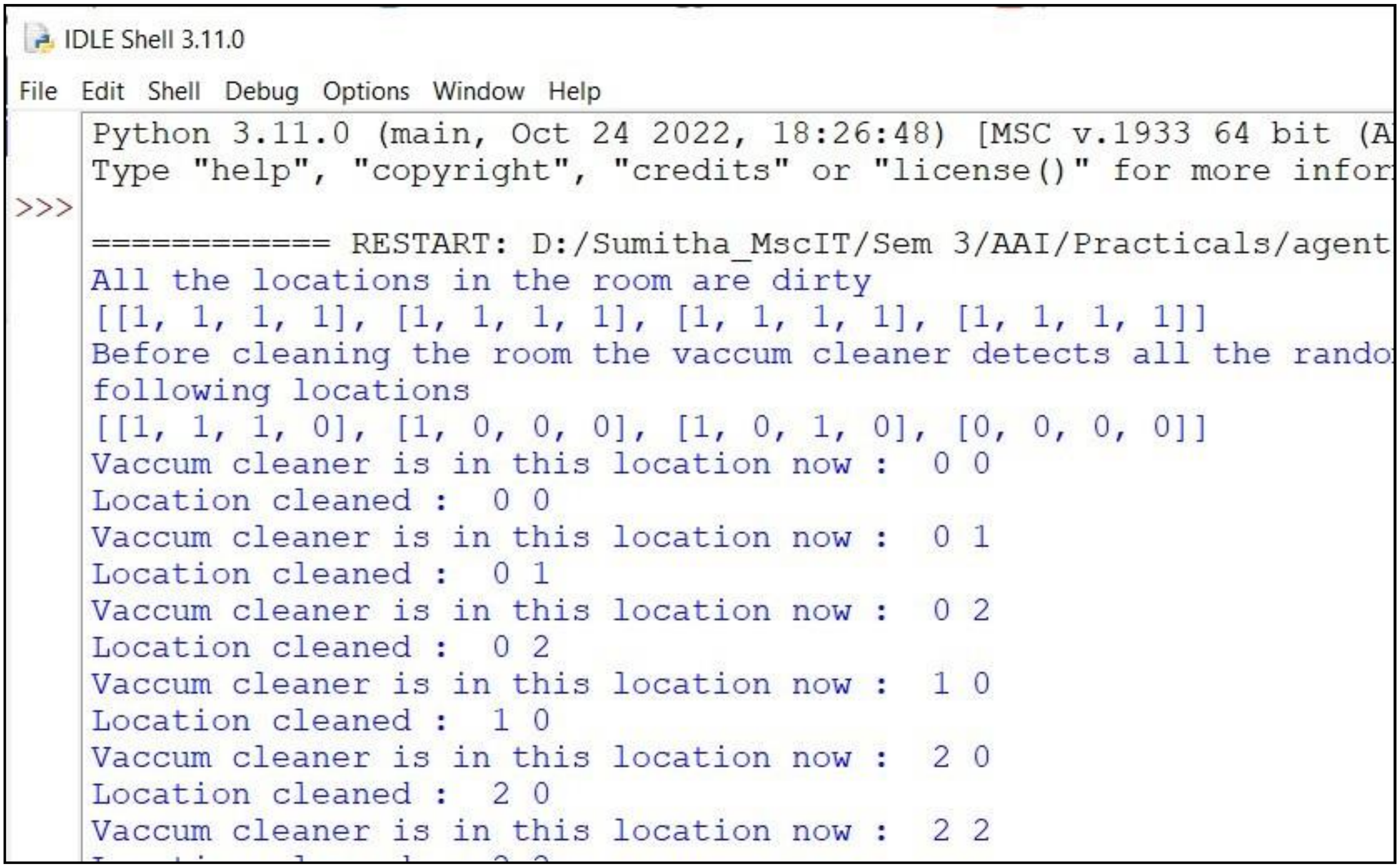
Practical No. 12

Aim: Design an Artificial Intelligence application to implement Intelligent Agent.

Code:

```
import random

def display(room):
    print(room)

# 1 means dirty location
# 0 means clean location

room = [
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    ]

print("All the locations in the room are dirty")
display(room)

x=0 #rows
y=0 #cols
while x<4:
    while y<4:
        room[x][y] = random.choice([0,1])
        y+=1
    x+=1
    y=0

print("Before cleaning the room the vaccum cleaner detects all the random dirts in the following
locations")
display(room)

x=0
y=0
z=0 #number of rooms cleaned

#Agent code
while x<4:
    while y<4:
        if(room[x][y] == 1):
            print("Vaccum cleaner is in this location now : ",x,y)
            room[x][y] = 0
            print("Location cleaned : ",x,y)
            z+=1
        y+=1

    x+=1
    y=0

print("Number of locations cleaned : ",z)

performance = (100-((z/16)*100))
print("Room is clean now")
display(room)
print("Cleaning performance : ",performance,"%")
```

Output:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (A
Type "help", "copyright", "credits" or "license()" for more infor
>>>
============ RESTART: D:/Sumitha_MscIT/Sem 3/AAI/Practicals/agent
All the locations in the room are dirty
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
Before cleaning the room the vaccum cleaner detects all the rando
following locations
[[1, 1, 1, 0], [1, 0, 0, 0], [1, 0, 1, 0], [0, 0, 0, 0]]
Vaccum cleaner is in this location now :  0 0
Location cleaned :  0 0
Vaccum cleaner is in this location now :  0 1
Location cleaned :  0 1
Vaccum cleaner is in this location now :  0 2
Location cleaned :  0 2
Vaccum cleaner is in this location now :  1 0
Location cleaned :  1 0
Vaccum cleaner is in this location now :  2 0
Location cleaned :  2 0
Vaccum cleaner is in this location now :  2 2
```