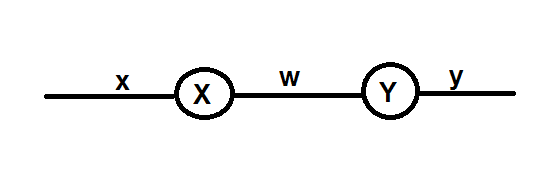
PRACTICAL NO -1

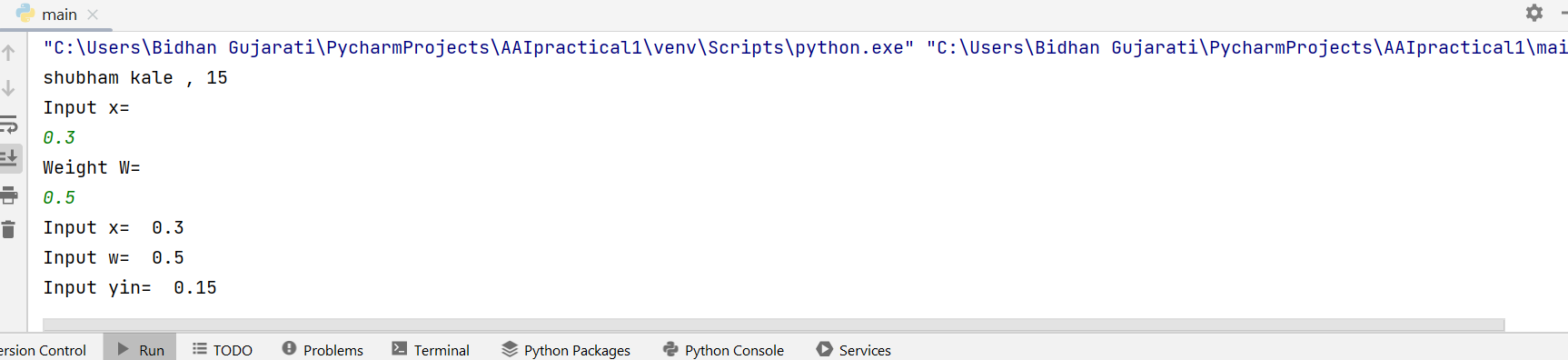
Q.1) A) Design a simple neural network with one input node and one output node. Calculate .net input of the network.



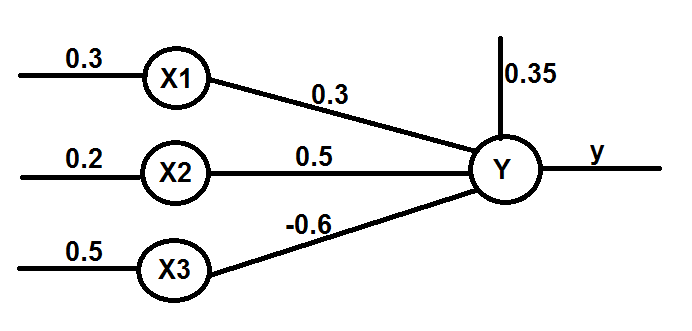
CODE:

print(‘shubham kale , 15)  
print("Input x=")  
x=float(input())  
print("Weight W= ")  
w=float(input())  
yin=x\*w  
print("Input x= ",x)  
print("Input w= ",w)  
print("Input yin= ",yin)

OUTPUT:

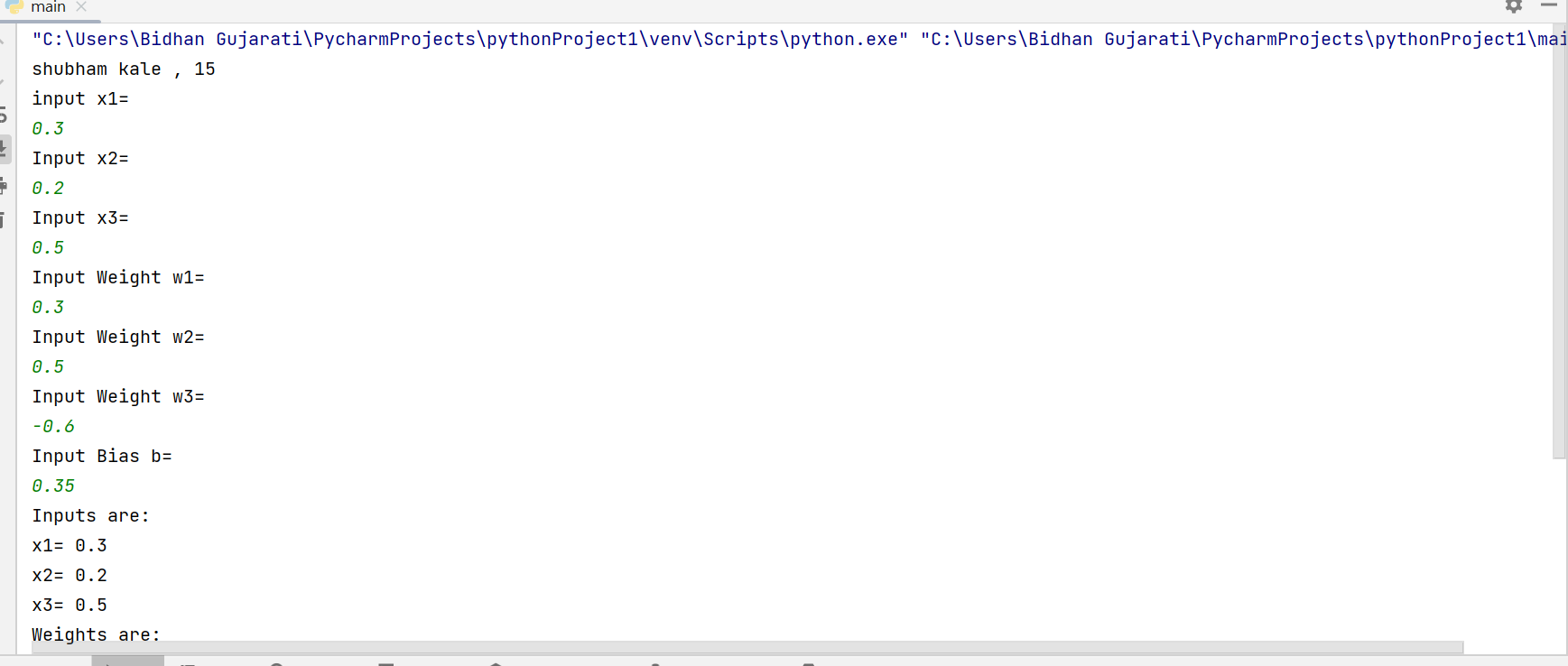


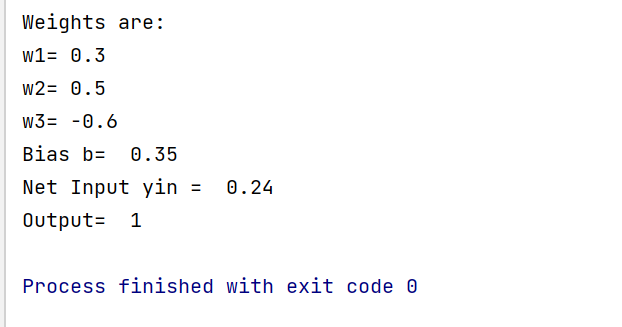
Q.1) B) Design the following neural network.



CODE:  
print(‘shubham kale , 15’)  
  
print("input x1= ")  
x1=float(input())  
print("Input x2= ")  
x2=float(input())  
print("Input x3= ")  
x3=float(input())  
  
print("Input Weight w1= ")  
w1=float(input())  
print("Input Weight w2= ")  
w2=float(input())  
print("Input Weight w3= ")  
w3=float(input())  
  
print("Input Bias b= ")  
b=float(input())  
  
yin=b+(x1\*w1+x2\*w2+x3\*w3)  
  
if yin>=0:  
 y=1  
else:  
 y=0  
  
print("Inputs are:")  
print("x1=",x1)  
print("x2=",x2)  
print("x3=",x3)  
  
print("Weights are:")  
print("w1=",w1)  
print("w2=",w2)  
print("w3=",w3)  
  
print("Bias b= ",b)  
  
print("Net Input yin = ",yin)  
print("Output= ",y)

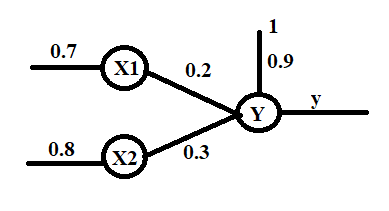
OUTPUT:





PRACTICAL No-2

1) Calculate the output of the neuron Y for the network shown below. Use binary and bipolar sigmodial activation function.



CODE:

print(‘shubham kale , 15’)

import math

print("Input x1= ")

x1=float(input())

print("Input x2= ")

x2=float(input())

print("Input weight w1= ")

w1=float(input())

print("Input weight w2= ")

w2=float(input())

print("Input Bias b=")

b=float(input())

print()

yin=b+(x1\*w1+x2\*w2)

y1=1/(1 + math.exp(-yin))

y2=(2/(1 + math.exp(-yin)))-1

print("Inputs are: ")

print("x1=",x1)

print("x2=",x2,'\n')

print("Weights are:")

print("w1=",w1)

print("w2=",w2,'\n')

print("Bias b=",b)

print("Net Input=",yin,'\n')

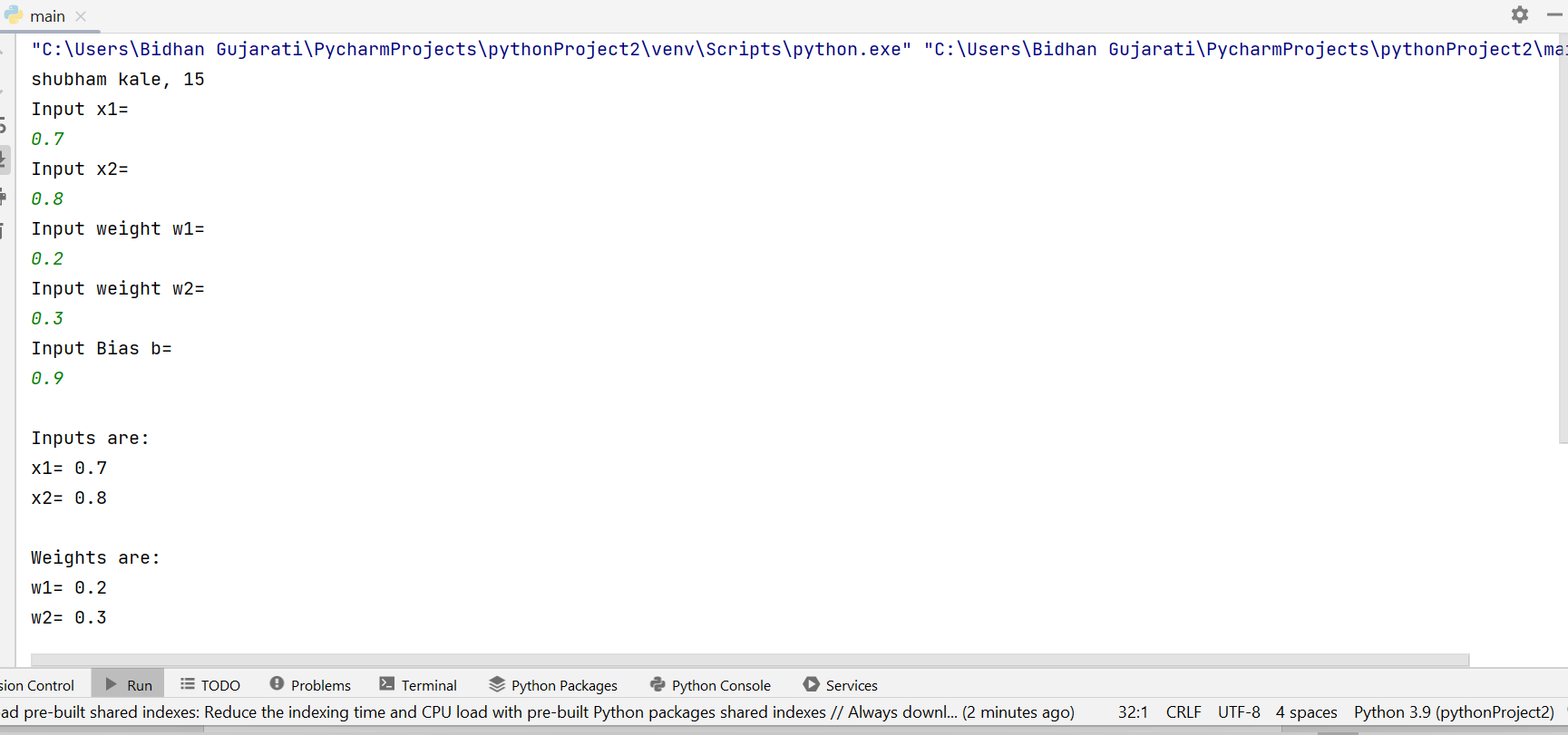
print("Binary Sigmodial Activation Function")

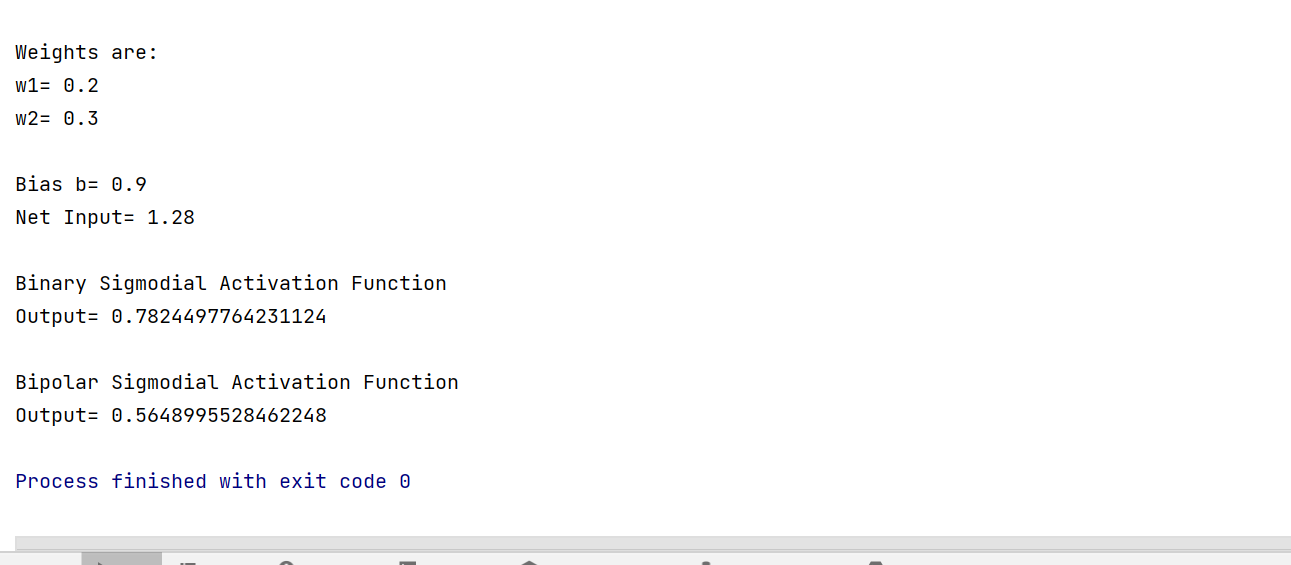
print("Output=",y1,'\n')

print("Bipolar Sigmodial Activation Function")

print("Output=",y2)

OUTPUT:





Practical No-3

Q.3) Design a simple machine learning model to train the training instances and test the same

Code:

print(‘shubham kale, 15)

import random;

from sklearn.linear\_model import LinearRegression

*#Create an empty list for the feature data set 'X' and target data set 'Y'*

feature\_set=[]

target\_set=[]

*#get the number of rows wanted for data set*

number\_of\_rows=200

*#limit the possible values in data set*

random\_number\_limit=2000

*#Create the training data*

*#Create and append a randomly generated data set to input and output*

for i in range(0,number\_of\_rows):

x = random.randint(0,random\_number\_limit)

y = random.randint(0, random\_number\_limit)

z = random.randint(0, random\_number\_limit)

print("x=",x,"\t y=",y,"\t z=",z)

function=(10\*x)+(2\*y)+(3\*z)

feature\_set.append([x,y,z])

target\_set.append(function)

*#Create linear regression model/object*

model=LinearRegression()

model.fit(feature\_set,target\_set)

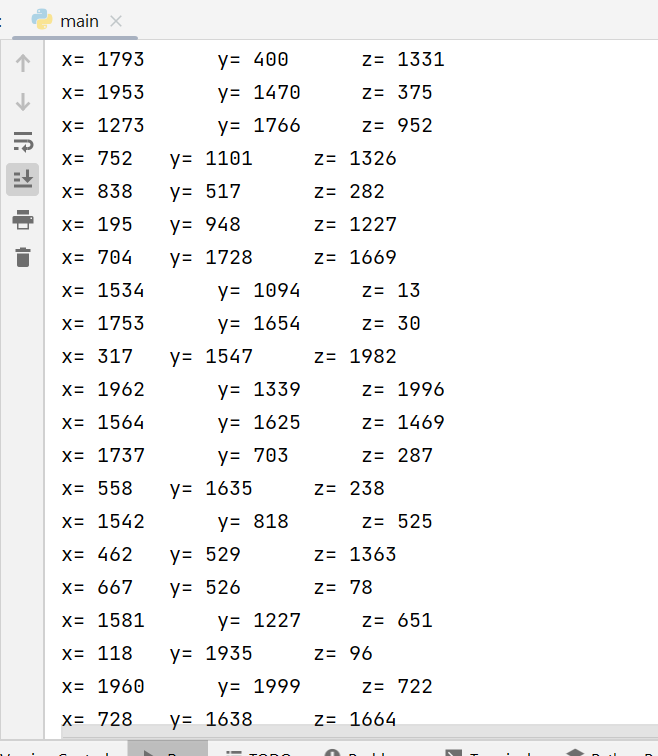
test\_set=[[1,3,5]]

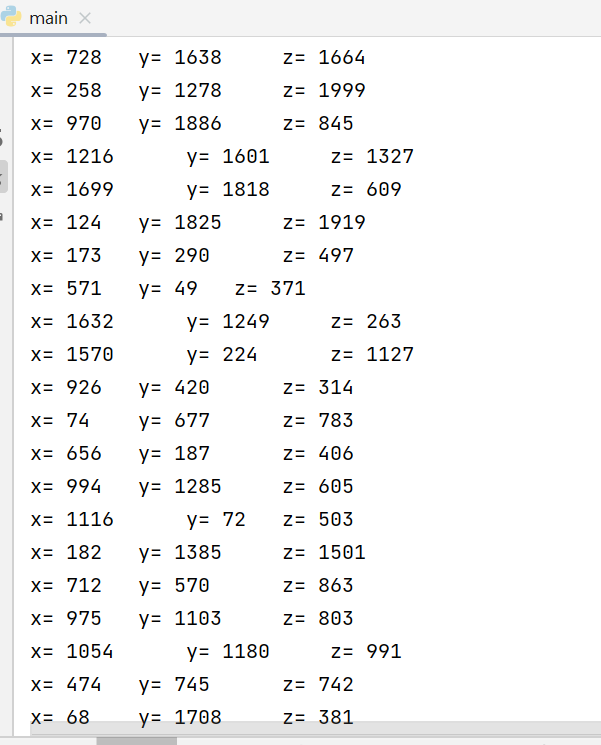
prediction=model.predict(test\_set)

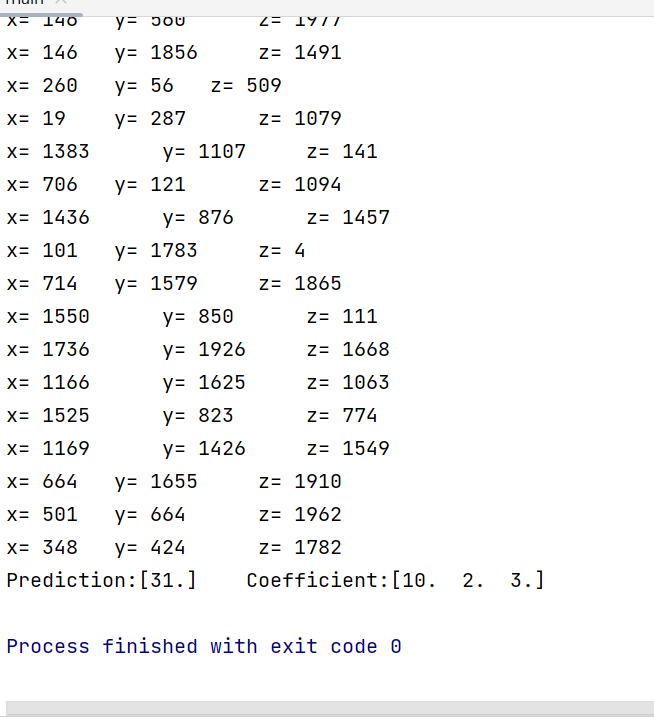
print("Prediction:"+str(prediction)+'\t'+'Coefficient:'+str(model.coef\_))

OUTPUT:









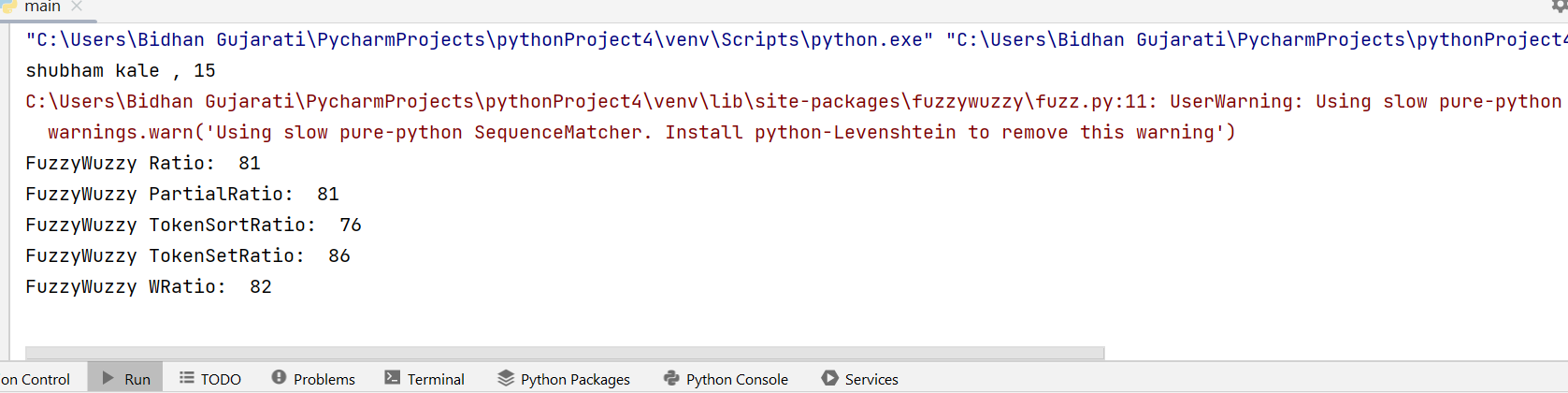
Practical No-4

Q.4) Find Ratios using Fuzzy logic and Calculate the levenshtein distance similarity ratios between the two strings (Sequence).

CODE:

print(‘shubham kale , 15)  
  
from fuzzywuzzy import fuzz  
from fuzzywuzzy import process  
  
s1="I like soft computing"  
s2="I like hard computing"  
print("FuzzyWuzzy Ratio: ",fuzz.ratio(s1,s2))  
print("FuzzyWuzzy PartialRatio: ",fuzz.partial\_ratio(s1,s2))  
print("FuzzyWuzzy TokenSortRatio: ",fuzz.token\_sort\_ratio(s1,s2))  
print("FuzzyWuzzy TokenSetRatio: ",fuzz.token\_set\_ratio(s1,s2))  
print("FuzzyWuzzy WRatio: ",fuzz.WRatio(s1,s2),'\n\n')

OUTPUT:



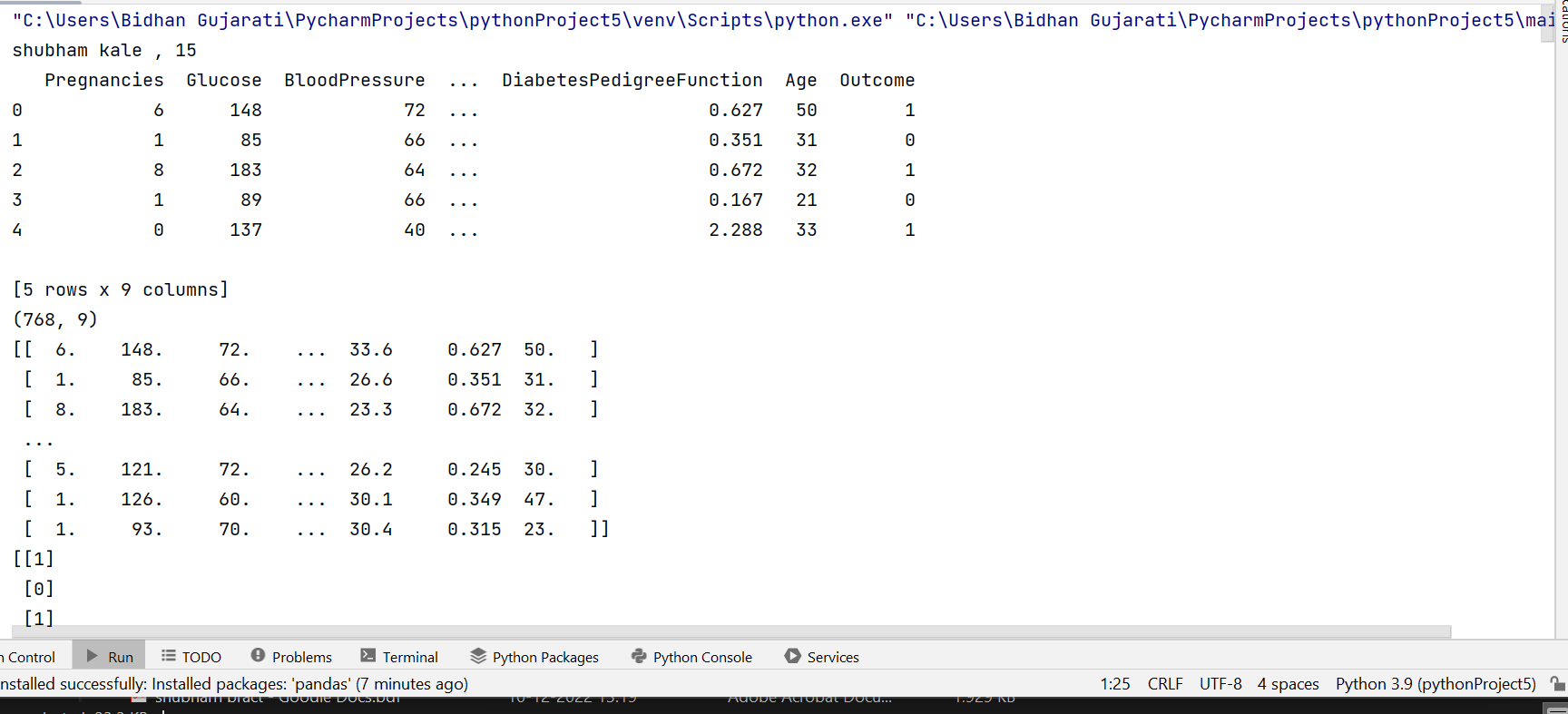
PRATICAL NO -5

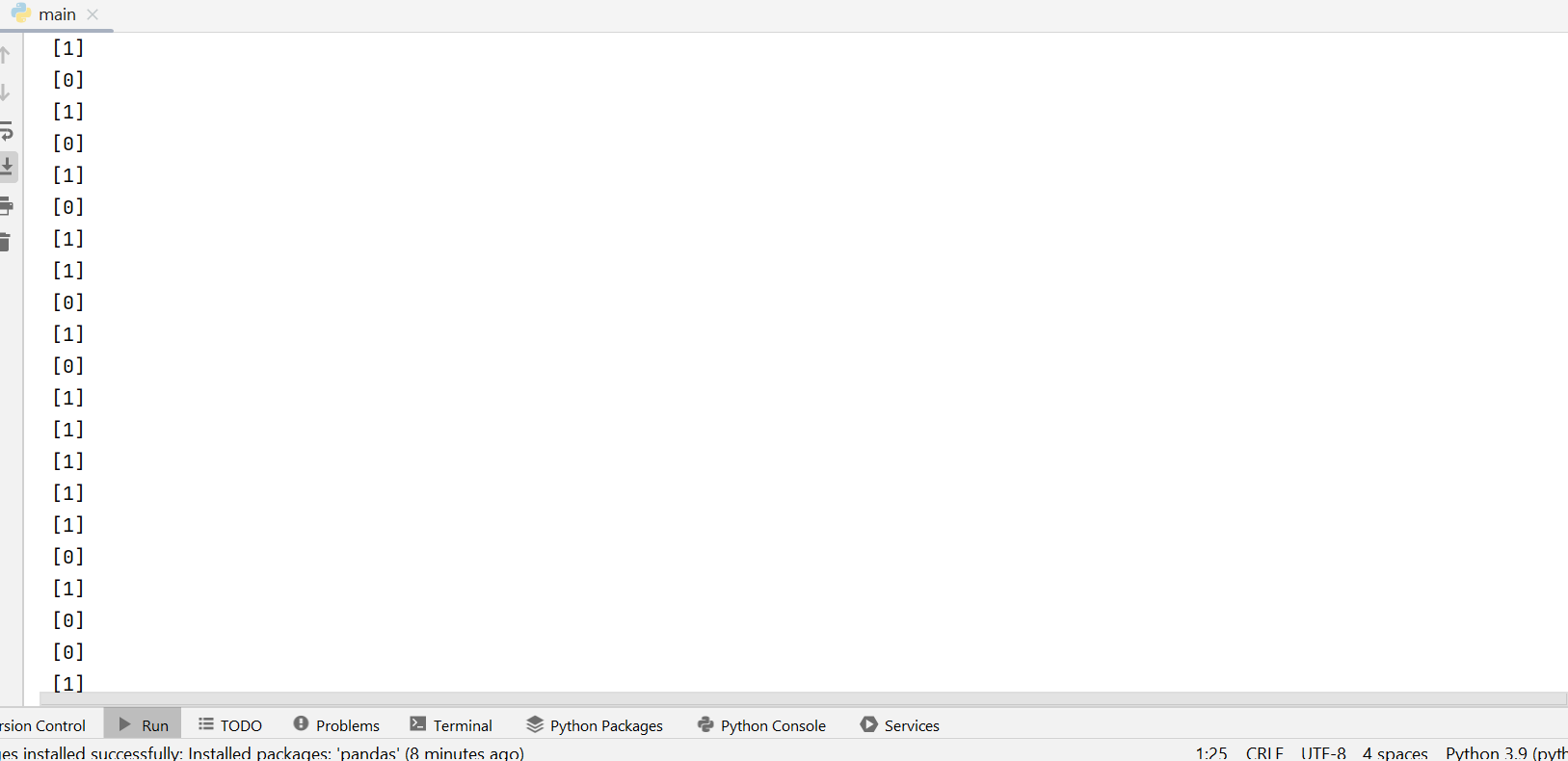
Q5) Write an application using python to stimulate supervised learning model (Logistic Regression)

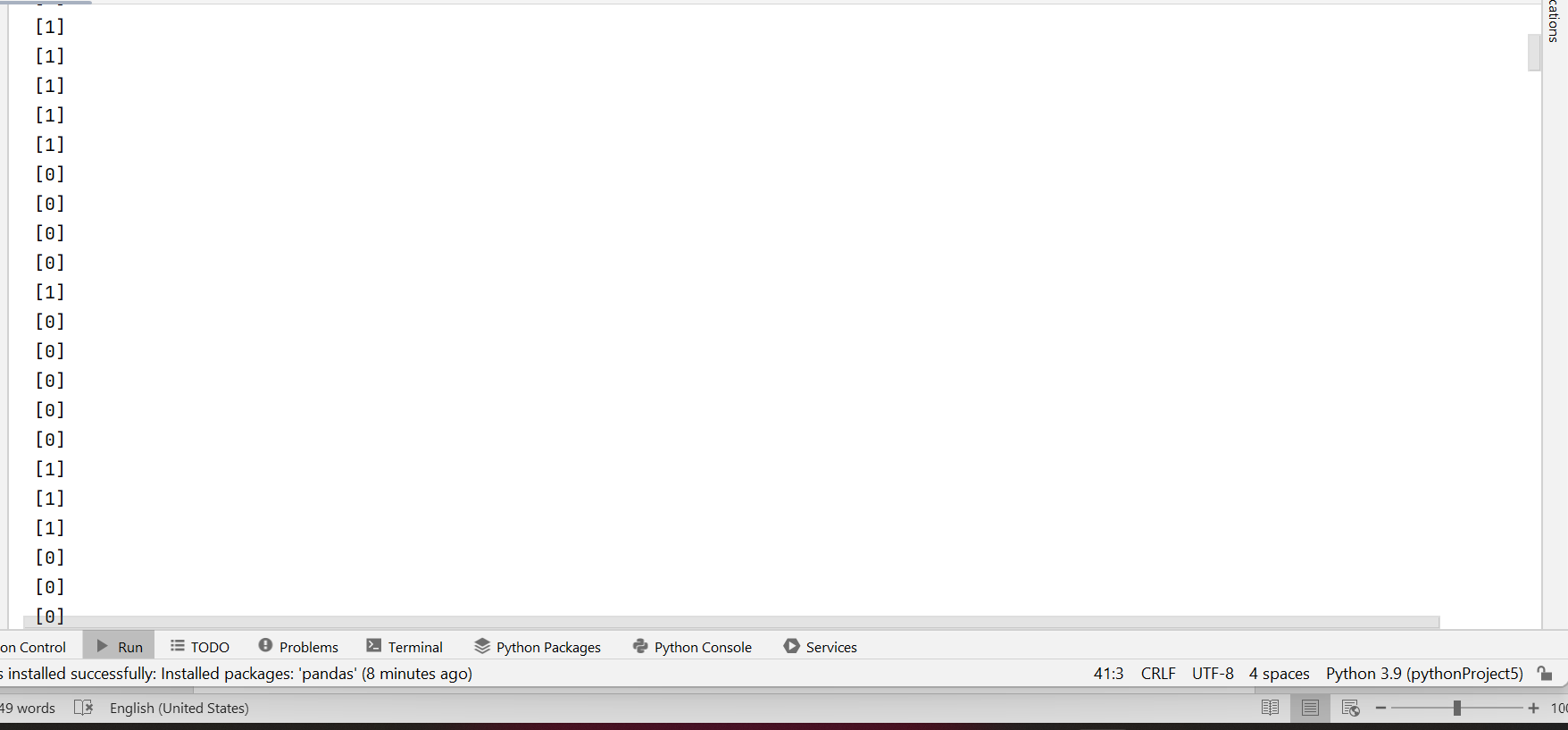
CODE:

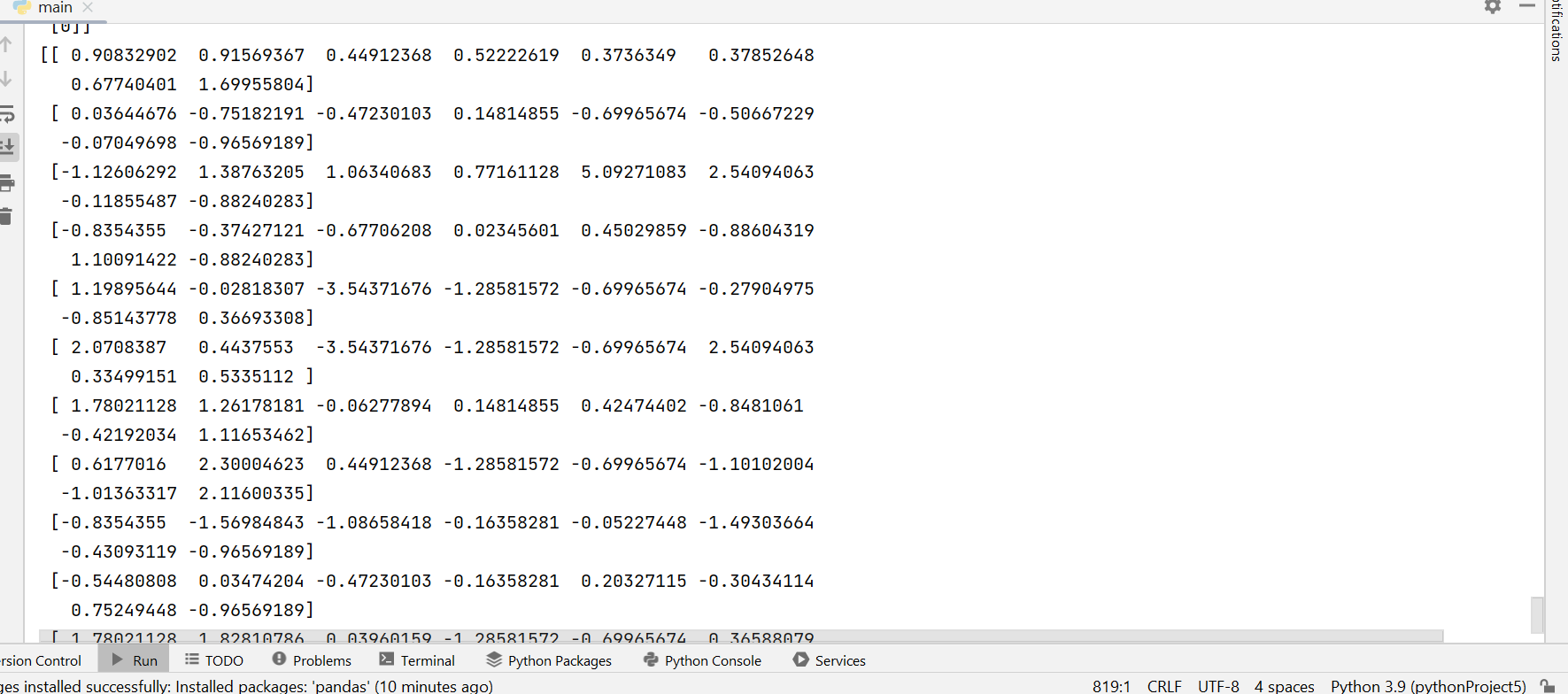
print(‘shubham kale, 15)  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix, accuracy\_score  
  
*#Pandas used for reading the data*dataset=pd.read\_csv("diabetes.csv")  
print(dataset.head())  
print(dataset.shape)  
x= dataset.iloc[:,[0,1,2,3,4,5,6,7]].values  
y=dataset.iloc[:,[-1]].values  
print(x)  
print(y)  
  
*#sklearn model used to split training and testing data*x\_train, x\_test, y\_train, y\_test=train\_test\_split(x,y, test\_size=0.2,random\_state=0)  
  
*#Standard scalar used to give structure to the dataset*sc= StandardScaler()  
x\_train\_new=sc.fit\_transform(x\_train)  
y\_test\_new=sc.fit\_transform(y\_test)  
  
print(x\_train\_new[0:15,:])

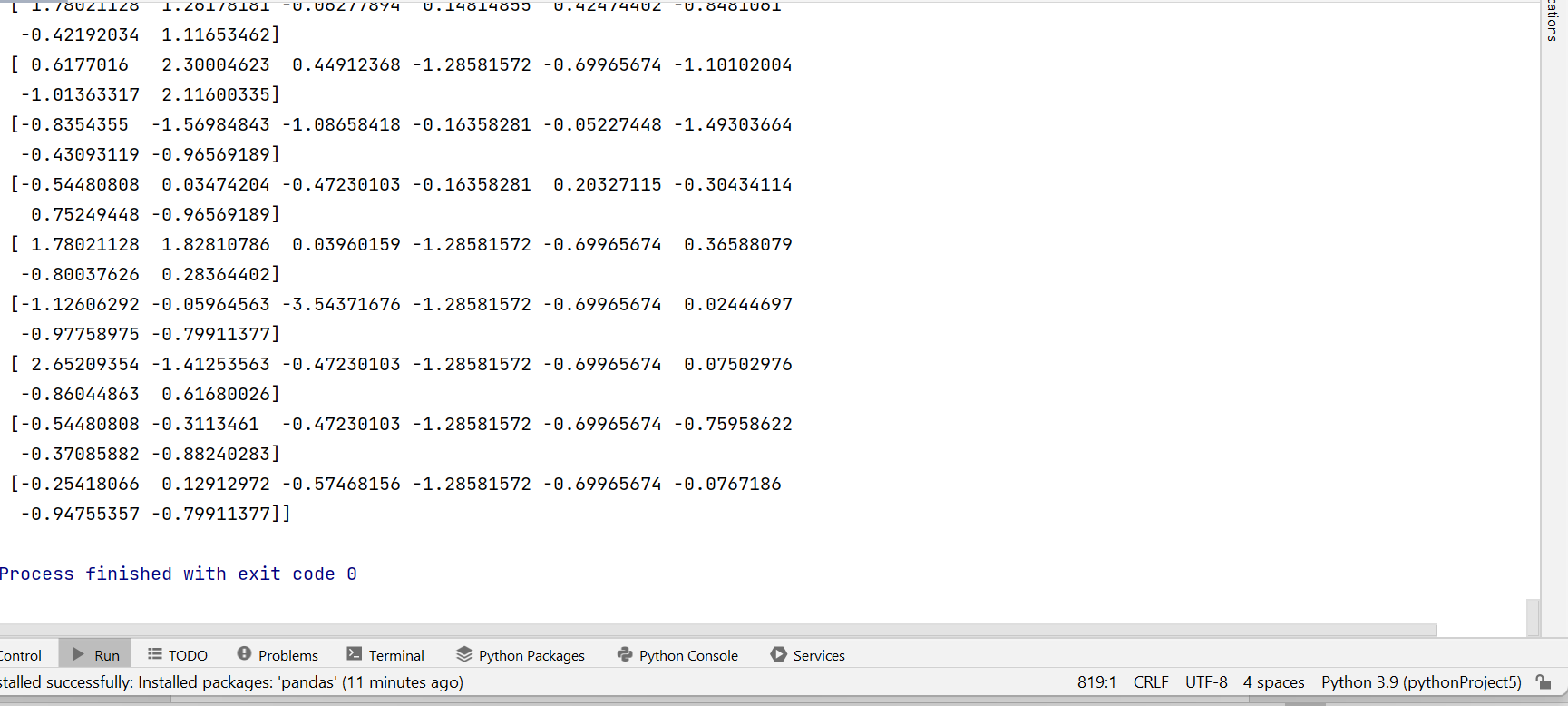
OUTPUT:











Practical 6

Q6) Fuzzy Logic : Membership in and not in and Identity Operators is, is not.

Code:-

print(‘shubham kale, 15’)

a = 10

b = 20

list = [1, 2, 3, 4, 5 ];

if a in list:

print ("Line 1 - a is available in the given list")

else:

print ("Line 1 - a is not available in the given list")

if b not in list :

print ("Line 2 - b is not available in the given list")

else:

print ("Line 2 - b is available in the given list")

a = 2

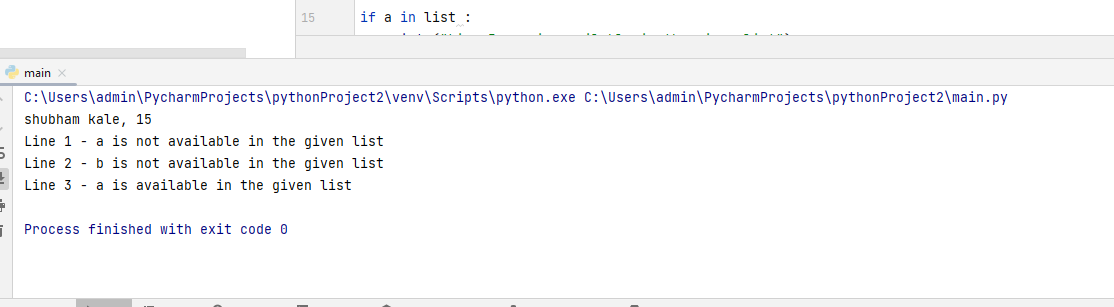
if a in list :

print ("Line 3 - a is available in the given list")

else:

print ("Line 3 - a is not available in the given list")

Output:-



Practical 7

Q7) Write an application to simulate supervised

Code:-

print(‘shubham kale, 15’)

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix, accuracy\_score  
from sklearn import datasets

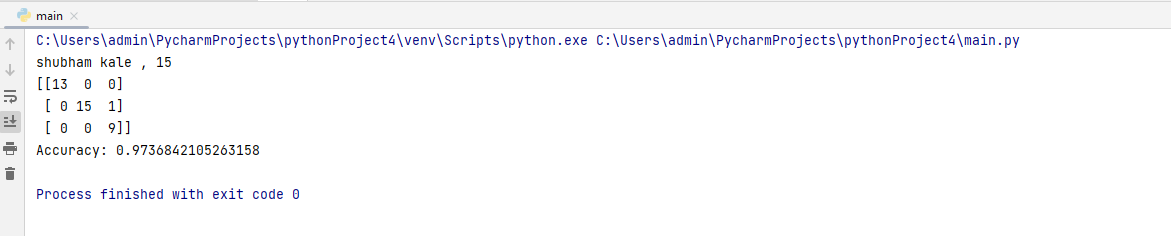
*#inporting the datasets*datasets = pd.read\_csv("iris.csv")  
datasets.describe()

*#spliting the datasets into the traning set and test set*x = datasets.iloc[:, [0,1,2,3]].values  
y = datasets.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 traning 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)  
from sklearn.metrics import confusion\_matrix  
cm = confusion\_matrix(y\_test, y\_pred)  
print(cm)  
print("Accuracy:",accuracy\_score(y\_test,y\_pred))

Output :-



Practical 8

Q8) Write a Program to implement BFS algorithm.

Code :-

print(‘shubham kale, 15’)

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 :-

