**Assignment No. 1**

**Code:-**

import numpy as np

class Perceptron:

def \_\_init\_\_(self,W,lr=0.5,epochs=100):

self.W=W

self.lr=lr

self.epochs=epochs

def activation\_fn(self,a):

if(a>=0):return 1

else:return 0

def predict(self,W,x):

y=np.dot(self.W,x)

z=self.activation\_fn(y)

return z

def fit(self,W,x,o):

for j in range(self.epochs):

for i in range(o.shape[0]):

x1=np.insert(x[i],0,1)

y=self.predict(self.W,x1)

loss=o[i]-y

self.W=self.W+(self.lr\*loss\*x1)

return self.W

input\_size=int(input("Enter input size:"))

w=np.zeros(np.array(input\_size+1))

# For AND

X=np.array([[0,0],[0,1],[1,0],[1,1]])

Y=np.array([0,0,0,1])

p=Perceptron(w)

w1=p.fit(w,X,Y)

print("For AND:-")

print("Trained Weights-",w1)

for i in range(4):

x1=np.insert(X[i],0,1)

O=p.predict(w1,x1)

print("For input as: ",X[i],"Output: ",O)

#For OR

X=np.array([[0,0],[0,1],[1,0],[1,1]])

Y=np.array([0,1,1,1])

p1=Perceptron(w)

w1=p1.fit(w,X,Y)

print("For OR:-")

print("Trained Weights-",w1)

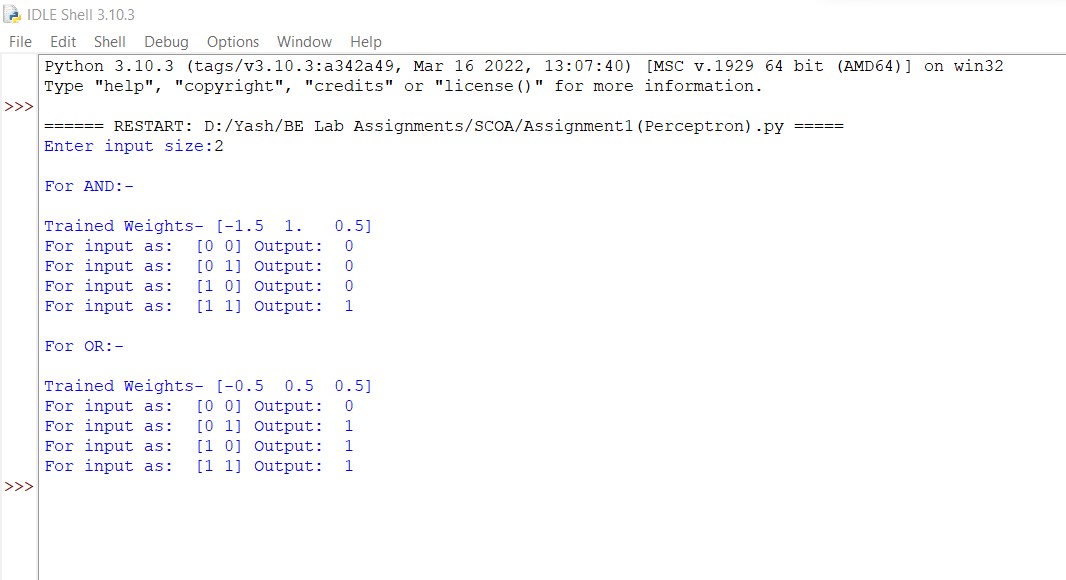
for i in range(4):

x1=np.insert(X[i],0,1)

O=p1.predict(w1,x1)

print("For input as: ",X[i],"Output: ",O)

**Output:-**



**Assignment No. 2**

**Code:-**

def Union(X,Y):

C={}

for k in X.keys():

if k in Y:

C[k]=max(X[k],Y[k])

Y[k]=-1

else:

C[k]=X[k]

for k in Y.keys():

if Y[k]!=-1:

C[k]=Y[k]

return C

def Intersection(X,Y):

C={}

for k in X.keys():

if k in Y:

C[k]=min(X[k],Y[k])

Y[k]=-1

else:

C[k]=X[k]

for k in Y.keys():

if Y[k]!=-1:

C[k]=Y[k]

return C

def Difference(X,Y):

C={}

for k in X.keys():

if k in Y:

C[k]=min(X[k],1-Y[k])

Y[k]=-1

else:

C[k]=X[k]

for k in Y.keys():

if Y[k]!=-1:

C[k]=Y[k]

return C

def Complement(X):

C={}

for k in X.keys():

C[k]=1-X[k]

return C

def Cartesian\_Product(X,Y):

C=list()

for i in range(len(X)):

for j in range(len(Y)):

C.append([X[i],Y[j]])

return C

A={"a":0.2, "b":0.45, "c":0.32, "d":0.12, "e":0.3}

B={"a":0.52, "b":0.5, "c":0.23, "d":0.6, "f":0.8}

print("Fuzzy set A ", A)

print("\nFuzzy set B ", B)

print("\nFuzzy Set Operations:-")

D=Union(A,B)

print("1]Union of Fuzzy set A and B: ", D)

A={"a":0.2, "b":0.45, "c":0.32, "d":0.12, "e":0.3}

B={"a":0.52, "b":0.5, "c":0.23, "d":0.6, "f":0.8}

D=Intersection(A,B)

print("\n2]Intersection of Fuzzy set A and B: ", D)

A={"a":0.2, "b":0.45, "c":0.32, "d":0.12, "e":0.3}

B={"a":0.52, "b":0.5, "c":0.23, "d":0.6, "f":0.8}

D=Difference(A,B)

print("\n3]Difference of Fuzzy set A and B: ", D)

A={"a":0.2, "b":0.45, "c":0.32, "d":0.12, "e":0.3}

B={"a":0.52, "b":0.5, "c":0.23, "d":0.6, "f":0.8}

D=Complement(A)

E=Complement(B)

print("\n4a]Complement of A: ",D)

print("4b]Complement of B: ",E)

# Cartesian Product of two fuzzy sets

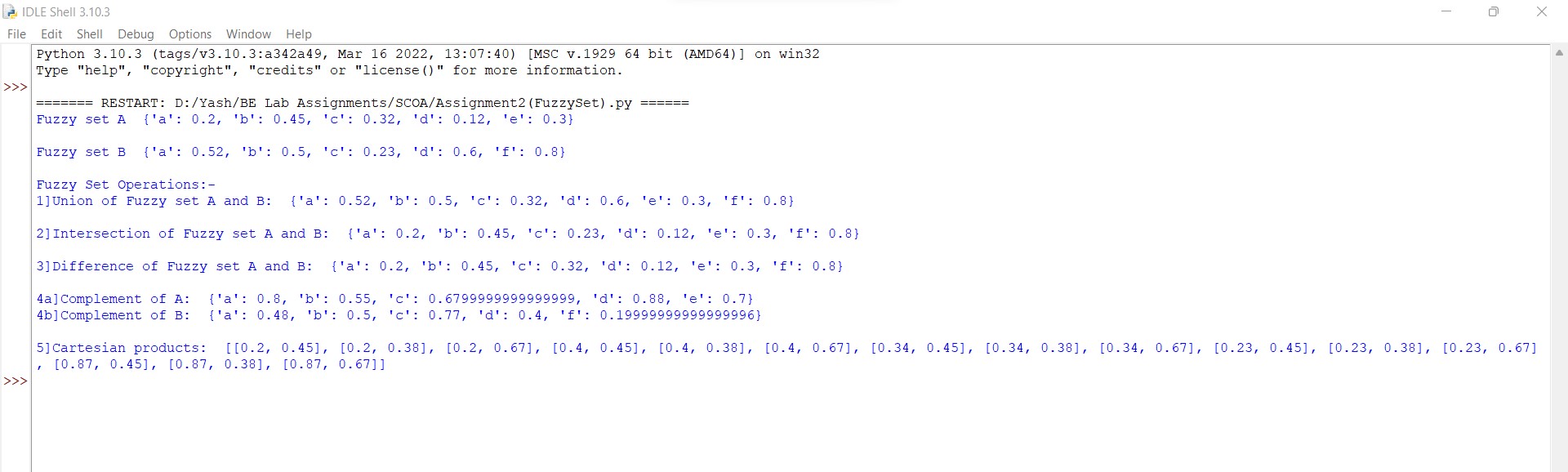
A=[0.2,0.4,0.34,0.23,0.87]

B=[0.45,0.38,0.67]

D=Cartesian\_Product(A,B)

print("\n5]Cartesian products: ",D)

**Output:-**

****

**Assignment No. 1**

**Code:-**

import numpy as np

class Perceptron:

def \_\_init\_\_(self,W,lr=0.5,epochs=100):

self.W=W

self.lr=lr

self.epochs=epochs

def activation\_fn(self,a):

if(a>=0):return 1

else:return 0

def predict(self,W,x):

y=np.dot(self.W,x)

z=self.activation\_fn(y)

return z

def fit(self,W,x,o):

for j in range(self.epochs):

for i in range(o.shape[0]):

x1=np.insert(x[i],0,1)

y=self.predict(self.W,x1)

loss=o[i]-y

self.W=self.W+(self.lr\*loss\*x1)

return self.W

input\_size=int(input("Enter input size:"))

w=np.zeros(np.array(input\_size+1))

# For AND

X=np.array([[0,0],[0,1],[1,0],[1,1]])

Y=np.array([0,0,0,1])

p=Perceptron(w)

w1=p.fit(w,X,Y)

print("For AND:-")

print("Trained Weights-",w1)

for i in range(4):

x1=np.insert(X[i],0,1)

O=p.predict(w1,x1)

print("For input as: ",X[i],"Output: ",O)

#For OR

X=np.array([[0,0],[0,1],[1,0],[1,1]])

Y=np.array([0,1,1,1])

p1=Perceptron(w)

w1=p1.fit(w,X,Y)

print("For OR:-")

print("Trained Weights-",w1)

for i in range(4):

x1=np.insert(X[i],0,1)

O=p1.predict(w1,x1)

print("For input as: ",X[i],"Output: ",O)

**Output:-**