import numpy as np

class Perceptron:

def \_\_init\_\_(self, input\_size, learning\_rate=0.1, epochs=100):

self.learning\_rate = learning\_rate

self.epochs = epochs

self.weights = np.zeros(input\_size + 1)

self.bias = 0

def activation(self, x):

return 1 if x >= 0 else 0

def predict(self, inputs):

summation = np.dot(inputs, self.weights[1:]) + self.weights[0]

return self.activation(summation)

def train(self, X, y):

for epoch in range(self.epochs):

for inputs, label in zip(X, y):

prediction = self.predict(inputs)

error = label - prediction

self.weights[1:] += self.learning\_rate \* error \* inputs

self.weights[0] += self.learning\_rate \* error

print(f"Epoch {epoch+1}/{self.epochs} - Weights: {self.weights}, Bias: {self.weights[0]}")

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

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

perceptron = Perceptron(input\_size=2, learning\_rate=0.1, epochs=10)

perceptron.train(X, y)

print("\nTesting the trained perceptron:")

for inputs in X:

print(f"Input: {inputs} -> Predicted Output: {perceptron.predict(inputs)}")