```
import numpy as np
import random


#activation function (unti step function)
def activation(x):
  return 1 if x >= 0 else 0


#perceptron
def perceptron(input , expected_output):
  global w0 , w1 , b
  w0 = random.uniform(-1,1)
  w1 = random.uniform(-1,1)
  b = random.uniform(-1,1)

  print("Initilized weights and bias are : ",w0 , w1 , b)
  print()

  epochs = 100
  lr = 0.1

  for epoch in range(epochs):
    total_error = 0
    for i in range(len(input)):
      X = input[i]
      y = expected_output[i]

      z = w0 * X[0] + w1 * X[1] + b
      y_pred = activation(z)  #forward pass

      error = y - y_pred # loss

      w0 = w0 + lr * error * X[0]
      w1 = w1 + lr * error * X[1]
      b = b + lr * error

      print(f'{i}/{epoch}: w0={w0}, w1={w1},  b={b}')
      total_error += abs(error)

    print()
    if total_error == 0:
      print(f"Training completed in {epoch+1} epochs.")
      break

  print("After trainig the weights and bias are : ",w0 , w1 , b)





#testing
def testing(input):
  X = input
  predicted = activation(w0 * X[0] + w1 * X[1] + b)
  return predicted


#Inputs and Expected_output
AND_inputs = [(0,0), (0,1), (1,0), (1,1)]
AND_expected_outputs = [0, 0, 0, 1]



#training the AND Gate
perceptron(AND_inputs , AND_expected_outputs)
```

```
0/4: w0=-0.16017917943126134, w1=-0.17979655559903737,  b=0.13037783061600466
1/4: w0=-0.16017917943126134, w1=-0.17979655559903737,  b=0.13037783061600466
2/4: w0=-0.16017917943126134, w1=-0.17979655559903737,  b=0.13037783061600466
3/4: w0=-0.06017917943126133, w1=-0.07979655559903737,  b=0.23037783061600467

0/5: w0=-0.06017917943126133, w1=-0.07979655559903737,  b=0.13037783061600466
1/5: w0=-0.06017917943126133, w1=-0.17979655559903737,  b=0.03037783061600466
2/5: w0=-0.06017917943126133, w1=-0.17979655559903737,  b=0.03037783061600466
3/5: w0=0.039820820568738674, w1=-0.07979655559903737,  b=0.13037783061600466

0/6: w0=0.039820820568738674, w1=-0.07979655559903737,  b=0.03037783061600466
1/6: w0=0.039820820568738674, w1=-0.07979655559903737,  b=0.03037783061600466
2/6: w0=-0.06017917943126133, w1=-0.07979655559903737,  b=-0.06962216938399535
3/6: w0=0.039820820568738674, w1=0.02020344440096264,  b=0.03037783061600466

0/7: w0=0.039820820568738674, w1=0.02020344440096264,  b=-0.06962216938399535
1/7: w0=0.039820820568738674, w1=0.02020344440096264,  b=-0.06962216938399535
2/7: w0=0.039820820568738674, w1=0.02020344440096264,  b=-0.06962216938399535
3/7: w0=0.13982082056873868, w1=0.12020344440096264,  b=0.03037783061600466

0/8: w0=0.13982082056873868, w1=0.12020344440096264,  b=-0.06962216938399535
1/8: w0=0.13982082056873868, w1=0.02020344440096264,  b=-0.16962216938399535
2/8: w0=0.13982082056873868, w1=0.02020344440096264,  b=-0.16962216938399535
3/8: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.06962216938399535

0/9: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.06962216938399535
1/9: w0=0.23982082056873869, w1=0.02020344440096264,  b=-0.16962216938399535
2/9: w0=0.13982082056873868, w1=0.02020344440096264,  b=-0.26962216938399536
3/9: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.16962216938399535

0/10: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.16962216938399535
1/10: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.16962216938399535
2/10: w0=0.13982082056873868, w1=0.12020344440096264,  b=-0.26962216938399536
3/10: w0=0.23982082056873869, w1=0.22020344440096265,  b=-0.16962216938399535

0/11: w0=0.23982082056873869, w1=0.22020344440096265,  b=-0.16962216938399535
1/11: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536
2/11: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536
3/11: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536

0/12: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536
1/12: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536
2/12: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536
3/12: w0=0.23982082056873869, w1=0.12020344440096264,  b=-0.26962216938399536

Training completed in 13 epochs.
After trainig the weights and bias are :  0.23982082056873869 0.12020344440096264 -0.26962216938399536
```

```
for input_data in AND_inputs:
    print(f"Input: {input_data}, Predicted Output: {testing(input_data)}")
```

```
Input: (0, 0), Predicted Output: 0
Input: (0, 1), Predicted Output: 0
Input: (1, 0), Predicted Output: 0
Input: (1, 1), Predicted Output: 1
```

Start coding or generate with AI.