# **AIM:** Implement ANDNOT function using McCulloch-Pits neuron (use binary data representation

print("#######\_\_\_\_\_\_Methos\_1\_\_\_\_\_\_########")

class McCullochPittsNeuron:

    def \_\_init\_\_(self, weights, threshold):

        self.weights = weights

        self.threshold = threshold

    def activate(self, inputs):

        if sum([x \* w for x, w in zip(inputs, self.weights)]) >= self.threshold:

            return 1

        else:

            return 0

def ANDNOT(a, b):

    weights = [1, -1]

    threshold = 1

    neuron = McCullochPittsNeuron(weights, threshold)

    return neuron.activate([a, b])

print("ANDNOT(0, 0) =", ANDNOT(0, 0))

print("ANDNOT(0, 1) =", ANDNOT(0, 1))

print("ANDNOT(1, 0) =", ANDNOT(1, 0))

print("ANDNOT(1, 1) =", ANDNOT(1, 1))

print("#########\_\_\_\_\_\_Methos\_2\_\_\_\_\_\_###########")

def mccullock\_pitts\_andnot(A,B):

    w1=1

    w2=-1

    threshold=0

    weighted\_sum=w1\*A+w2\*B

    output=1 if weighted\_sum>threshold else 0

    return output

input\_A=int(input("Enter the value of A (0 or 1): "))

input\_B=int(input("Enter the value of B (0 or 1): "))

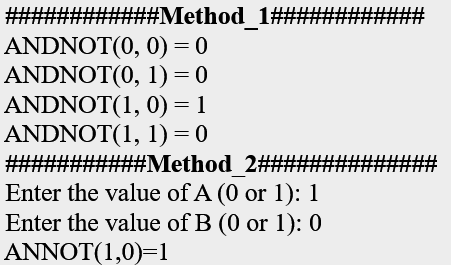
if input\_A in (0,1) and input\_B in (0,1):

    result=mccullock\_pitts\_andnot(input\_A,input\_B)

    print(f"ANNOT({input\_A},{input\_B})={result}")

else:

    print("Invalid input.Please enter 0 or 1 for A and B.")

**Output:**

## **AIM:** Generate XOR function using McCulloch-Pitts neural network.

import numpy as np

print("\*\*\*\*\*\* XOR CODE \*\*\*\*\*\*\*\*\*")

def sigmoid(x):

  return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(x):

  return x \* (1 - x)

try:

  A = int(input("Enter 1st Binary Input(0,1): "))

  B = int(input("Enter 2nd Binary Input (0,1): "))

  if A in [0, 1] and B in [0, 1]:

    # Create the XOR truth table

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

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

    # Initialize the network parameters

    input\_size = 2

    hidden\_size = 2

    output\_size = 1

    hidden\_weight = np.random.uniform(size=(input\_size, hidden\_size))

    hidden\_bias = np.random.uniform(size=(1, hidden\_size))

    output\_weights = np.random.uniform(size=(hidden\_size, output\_size))

    output\_bias = np.random.uniform(size=(1, output\_size))

    # Forward pass

    hidden\_layer\_input = np.dot(x, hidden\_weight) + hidden\_bias

    hidden\_layer\_output = sigmoid(hidden\_layer\_input)

    output\_layer\_input = np.dot(hidden\_layer\_output, output\_weights)

+ output\_bias

    output\_layer\_output = sigmoid(output\_layer\_input)

    # Print the output

    print(f"XOR({A}, {B}) = {(output\_layer\_output[0][0])}")

  else:

    print("Invalid Input. Please enter 0 or 1 for binary input.")

except:

  print("Invalid input.")

**Output:**

\*\*\*\*\*\* XOR CODE \*\*\*\*\*\*\*\*\*

Enter 1st Binary Input(0,1): 1

Enter 2nd Binary Input (0,1): 0

XOR(1, 0) = 0.7789986774398672