**DL EXPERIMENT 1**

**AND GATE:**

**Python code:**def step(x):

return 1 if x >= 0 else 0

def check\_AND(w1, w2, b):

AND\_truth = {

(0, 0): 0,

(0, 1): 0,

(1, 0): 0,

(1, 1): 1

}

for (x1, x2), expected in AND\_truth.items():

output = step(x1 \* w1 + x2 \* w2 + b)

if output != expected:

return False

return True

w1 = float(input("Enter w1: "))

w2 = float(input("Enter w2: "))

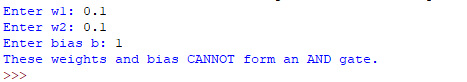
b = float(input("Enter bias b: "))

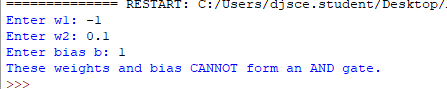
if check\_AND(w1, w2, b):

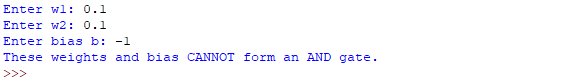
print("These weights and bias CAN form an AND gate.")

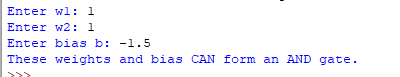
else:

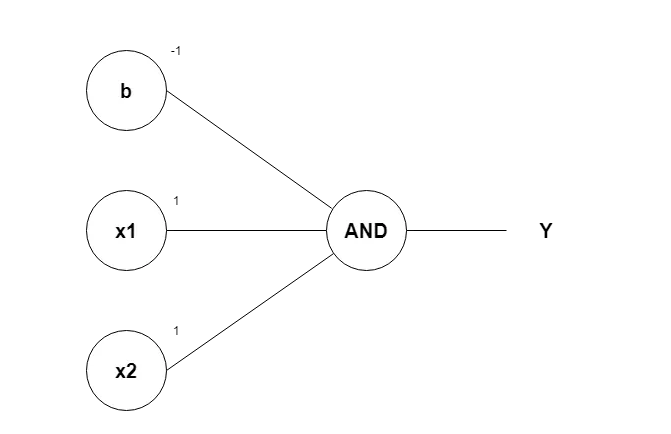
print("These weights and bias CANNOT form an AND gate.")

**Output:  
**

****

****

****

**ARCHITECTURE:  
**

**OR GATE:**

**CODE:**def step(x):

return 1 if x >= 0 else 0

def check\_OR(w1, w2, b):

AND\_truth = {

(0, 0): 0,

(0, 1): 1,

(1, 0): 1,

(1, 1): 1

}

for (x1, x2), expected in AND\_truth.items():

output = step(x1 \* w1 + x2 \* w2 + b)

if output != expected:

return False

return True

w1 = float(input("Enter w1: "))

w2 = float(input("Enter w2: "))

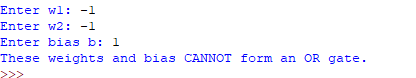
b = float(input("Enter bias b: "))

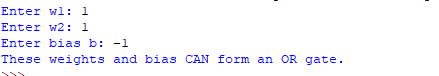
if check\_OR(w1, w2, b):

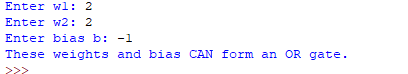
print("These weights and bias CAN form an OR gate.")

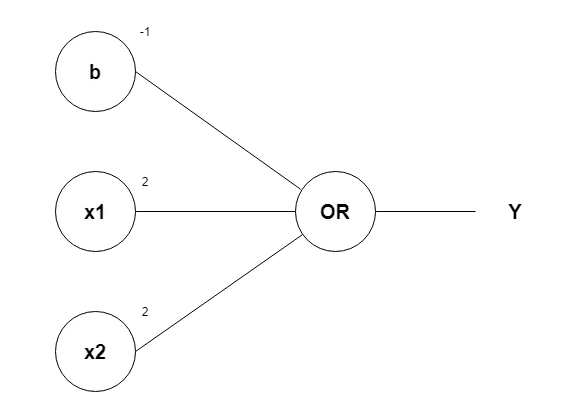
else:

print("These weights and bias CANNOT form an OR gate.")

**OUTPUT:  
**

****

****

**ARCHITECTURE:  
**

**NOT GATE:**

def step(x):

return 1 if x >= 0 else 0

def check\_NOT(w1, b):

NOT\_truth = {

0: 1,

1: 0

}

for x, expected in NOT\_truth.items():

output = step(x \* w1 + b)

if output != expected:

return False

return True

w1 = float(input("Enter w1: "))

b = float(input("Enter bias b: "))

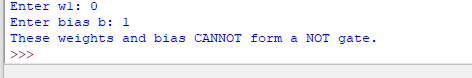
if check\_NOT(w1, b):

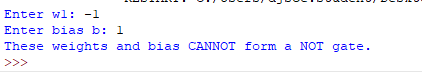
print("These weights and bias CAN form a NOT gate.")

else:

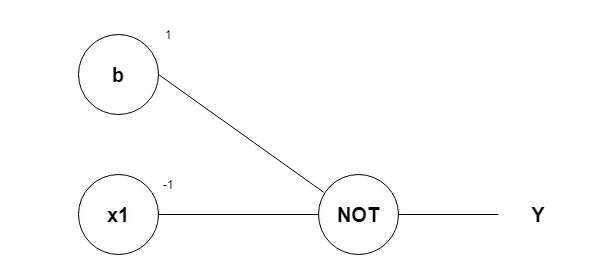
print("These weights and bias CANNOT form a NOT gate.")

**OUTPUT:  
**

****

****

****

****

**XOR GATE:  
CODE:**def step(x):

return 1 if x >= 0 else 0

def perceptron(x1, x2, w1, w2, b):

return step(x1\*w1 + x2\*w2 + b)

def perceptron\_not(x, w, b):

return step(x\*w + b)

def check\_final\_layer\_XOR(w1\_final, w2\_final, b\_final):

w1\_and, w2\_and, b\_and = 0.5, 0.5, -0.7

w1\_or, w2\_or, b\_or = 0.5, 0.5, -0.2

w\_not, b\_not = -1, 0.5

XOR\_truth = {

(0, 0): 0,

(0, 1): 1,

(1, 0): 1,

(1, 1): 0

}

for (x1, x2), expected in XOR\_truth.items():

and\_out = perceptron(x1, x2, w1\_and, w2\_and, b\_and)

or\_out = perceptron(x1, x2, w1\_or, w2\_or, b\_or)

not\_and\_out = perceptron\_not(and\_out, w\_not, b\_not)

xor\_out = perceptron(or\_out, not\_and\_out, w1\_final, w2\_final, b\_final)

if xor\_out != expected:

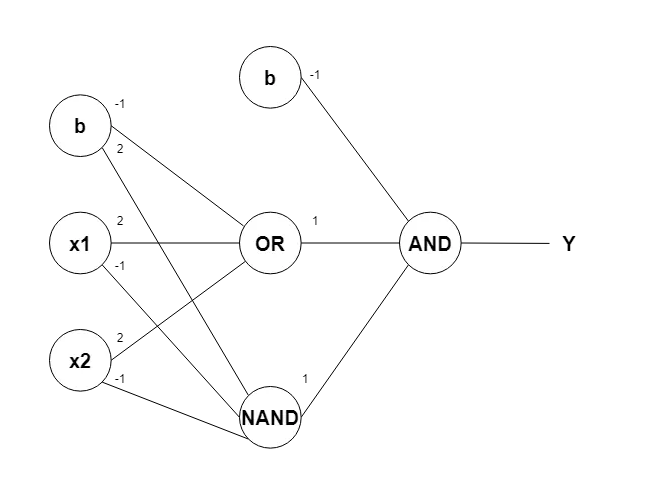
return False

return True

print(check\_final\_layer\_XOR(1, 1, -1.5))

**OUTPUT:  
**

**GIVING DIFFERENT WEIGTHS LETS SAY : -1, 1 AND BIAS AS 1 WE GET:  
**

**ARCHITECTURE:  
**