***AI EXPERIMENT 07***

1. ***And Gate with 2 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result<=1.5:  
        return 0  
    else:  
        return 1  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def andfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 2 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    eg1=np.array([i1,i2])  
    inps.append(eg1)  
      
for ins in inps:  
    print("And (",ins[0]," , ",ins[1],") : ",andfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 4**

**Enter 2 digits 0 or 1**

**0**

**0**

**1**

**0**

**0**

**1**

**1**

**1**

**And ( 0 , 0 ) : 0**

**And ( 1 , 0 ) : 0**

**And ( 0 , 1 ) : 0**

**And ( 1 , 1 ) : 1**

***-----------------------------------------------------------------------------------------------------------------***

1. ***OR Gate with 2 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result<=0.5:  
        return 0  
    else:  
        return 1  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def orfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 2 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    eg1=np.array([i1,i2])  
    inps.append(eg1)  
      
for ins in inps:  
    print("OR (",ins[0]," , ",ins[1],") : ",orfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 4**

**Enter 2 digits 0 or 1**

**1**

**0**

**0**

**1**

**1**

**1**

**0**

**0**

**OR ( 1 , 0 ) : 1**

**OR ( 0 , 1 ) : 1**

**OR ( 1 , 1 ) : 1**

**OR ( 0 , 0 ) : 0**

***-----------------------------------------------------------------------------------------------------------------***

1. ***Nand Gate with 2 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result<=1.5:  
        return 1  
    else:  
        return 0  
         
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)      
     
def nandfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 2 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    eg1=np.array([i1,i2])  
    inps.append(eg1)  
     
for ins in inps:  
    print("Nand (",ins[0]," , ",ins[1],") : ",nandfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 4**

**Enter 2 digits 0 or 1**

**1**

**0**

**0**

**1**

**1**

**1**

**0**

**0**

**Nand ( 1 , 0 ) : 1**

**Nand ( 0 , 1 ) : 1**

**Nand ( 1 , 1 ) : 0**

***-----------------------------------------------------------------------------------------------------------------***

1. ***Nor Gate with 2 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result<=0.5:  
        return 1  
    else:  
        return 0  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def norfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 2 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    eg1=np.array([i1,i2])  
    inps.append(eg1)  
      
for ins in inps:  
    print("NOR (",ins[0]," , ",ins[1],") : ",norfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 4**

**Enter 2 digits 0 or 1**

**1**

**0**

**0**

**1**

**1**

**1**

**0**

**0**

**NOR ( 1 , 0 ) : 0**

**NOR ( 0 , 1 ) : 0**

**NOR ( 1 , 1 ) : 0**

**NOR ( 0 , 0 ) : 1**

***-----------------------------------------------------------------------------------------------------------------***

1. ***XOR Gate with 2 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result>=0:  
        return 1  
    else:  
        return 0  
  
         
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)      
     
def xorfn(x):  
    #Assign wts for input of each of 3 neurons  
    wts\_gt1=np.array([-1,-1])  
    wts\_gt2=np.array([-1,-1])  
    wts\_gt3=np.array([1,-1])  
     
    #assign bias value for each og gates  
    bias1=1.5  
    bias2=0.5  
    bias3=-0.5  
     
    #Obtain reslt for gate1  
    gate1\_op=neuron(x,wts\_gt1,bias1)  
     
    #Obtain reslt for gate2  
    gate2\_op=neuron(x,wts\_gt2,bias2)  
     
    #pass gate1 and gate2 op as input to gate3  
    gate\_3\_input=np.array([gate1\_op,gate2\_op])  
    return neuron(gate\_3\_input,wts\_gt3,bias3)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 2 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    eg1=np.array([i1,i2])  
    inps.append(eg1)  
     
for ins in inps:  
    print("XOR (",ins[0]," , ",ins[1],") : ",xorfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 4**

**Enter 2 digits 0 or 1**

**1**

**1**

**0**

**1**

**1**

**0**

**0**

**0**

**XOR ( 1 , 1 ) : 0**

**XOR ( 0 , 1 ) : 1**

**XOR ( 1 , 0 ) : 1**

**XOR ( 0 , 0 ) : 0**

***-----------------------------------------------------------------------------------------------------------------***

1. ***Not Gate***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result==0.5:  
        return 1  
    else:  
        return 0  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def notfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 1 digit 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    eg1=np.array([i1])  
    inps.append(eg1)  
      
for ins in inps:  
    print("NOT (",ins[0],") : ",notfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 2**

**Enter 1 digit 0 or 1**

**1**

**0**

**NOT ( 1 ) : 0**

**NOT ( 0 ) : 1**

***-----------------------------------------------------------------------------------------------------------------***

1. ***And Gate with 3 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result>=3.5:  
        return 1  
    else:  
        return 0  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def andfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 3 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    i3=int(input(""))    
    eg1=np.array([i1,i2,i3])  
    inps.append(eg1)  
      
for ins in inps:  
    print("And (",ins[0]," , ",ins[1]," , ",ins[2],") : ",andfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 3**

**Enter 3 digits 0 or 1**

**1**

**0**

**0**

**1**

**1**

**1**

**0**

**1**

**1**

**And ( 1 , 0 , 0 ) : 0**

**And ( 1 , 1 , 1 ) : 1**

**And ( 0 , 1 , 1 ) : 0**

***-----------------------------------------------------------------------------------------------------------------***

1. ***Or Gate with 3 inputs***

***Program:***

import numpy as np  
def binarythreshhold(result):  
    if result==0.5:  
        return 0  
    else:  
        return 1  
          
def neuron(x\_inputs,wt\_values,bias):  
    sum1=np.dot(x\_inputs,wt\_values)+bias  
    return binarythreshhold(sum1)       
      
def orfn(x):  
    #Assign wts for input of neuron  
    wts=np.array([1,1,1])  
    #assign bias value  
    bias=0.5  
    #calculate the op of ading two numbers in array x  
    return neuron(x,wts,bias)  
  
inps=[]  
n=int(input("Enter no of test cases: "))    
print("Enter 3 digits 0 or 1")  
for i in range(n):  
    i1=int(input(""))    
    i2=int(input(""))    
    i3=int(input(""))    
    eg1=np.array([i1,i2,i3])  
    inps.append(eg1)  
      
for ins in inps:  
    print("Or (",ins[0]," , ",ins[1]," , ",ins[2],") : ",orfn(ins))

***Output:***

***-----------------------------------------------------------------------------------------------------------------***

**Enter no of test cases: 3**

**Enter 3 digits 0 or 1**

**1**

**0**

**0**

**0**

**0**

**0**

**1**

**1**

**0**

**Or ( 1 , 0 , 0 ) : 1**

**Or ( 0 , 0 , 0 ) : 0**

**Or ( 1 , 1 , 0 ) : 1**

***-----------------------------------------------------------------------------------------------------------------***