```python
import numpy as np

# weights
w = np.array([1, 1]) # Weighst for AND and OR gate
wNOT = -1

# biases
bNOT = 0.5
bAND = -1.5
bOR = -0.5

def unitStep_activation(v):
  if v >= 0:
    return 1
  else:
    return 0

def dot_activation(x, w, b):
  v = np.dot(w, x) + b
  y = unitStep_activation(v)
  return y

def NOT_logic(x):
  return dot_activation(x, wNOT, bNOT)

def AND_logic(x):
  return dot_activation(x, w, bAND)

def OR_logic(x):
  return dot_activation(x, w, bOR)

# XOR Logic Function
def XOR_logic(x):
  y1 = AND_logic(x)
  y2 = OR_logic(x)
  y3 = NOT_logic(y1)
  final_x = np.array([y2, y3])
  finalOutput = AND_logic(final_x)
  return finalOutput

# testing the Perceptron Model
test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
for i in test:
  print("XOR({}, {}) = {}".format(i[0], i[1], XOR_logic(i)))

```

```
XOR(0, 0) = 0
XOR(0, 1) = 1
XOR(1, 0) = 1
XOR(1, 1) = 0
```