

```
# importing Python library
import numpy as np

# define Unit Step Function
def unitStep(v):
    if v >= 0:
        return 1
    else:
        return 0

# design Perceptron Model
def perceptronModel(x, w, b):
    v = np.dot(w, x) + b
    y = unitStep(v)
    return y

# NOT Logic Function
# wNOT = -1, bNOT = 0.5
def NOT_logicFunction(x):
    wNOT = -1
    bNOT = 0.5
    return perceptronModel(x, wNOT, bNOT)

# OR Logic Function
# w1 = 1, w2 = 1, bOR = -0.5
def OR_logicFunction(x):
    w = np.array([1, 1])
    bOR = -0.5
    return perceptronModel(x, w, bOR)

# NOR Logic Function
# with OR and NOT
# function calls in sequence
def NOR_logicFunction(x):
    output_OR = OR_logicFunction(x)
    output_NOT = NOT_logicFunction(output_OR)
    return output_NOT

# testing the Perceptron Model
test1 = np.array([0, 1])
test2 = np.array([1, 1])
test3 = np.array([0, 0])
test4 = np.array([1, 0])

print("NOR({}, {}) = {}".format(0, 1, NOR_logicFunction(test1)))
print("NOR({}, {}) = {}".format(1, 1, NOR_logicFunction(test2)))
print("NOR({}, {}) = {}".format(0, 0, NOR_logicFunction(test3)))
print("NOR({}, {}) = {}".format(1, 0, NOR_logicFunction(test4)))

NOR(0, 1) = 0
NOR(1, 1) = 0
NOR(0, 0) = 1
NOR(1, 0) = 0
```