class MP\_Neuron:

threshold = 0

w1 = 0

w2 = 0

w1\_vals= [-1, 1]

w2\_vals= [-1, 1]

thresh\_vals = [-2, -1, 0, 1, 2]

def \_\_init\_\_(self, input\_matrix):

self.input\_matrix = input\_matrix

def iterate\_all\_values(self):

for w1 in self.possible\_w1\_vals:

self.w1 = w1

for w2 in self.possible\_w2\_vals:

self.w2 = w2

for threshold in self. thresh\_vals:

self.threshold = threshold

if self.check\_combination():

return True

return False

def check\_combination(self):

valid = True

for (x1, x2, y) in self.input\_matrix:

if not self.compare\_target(x1, x2, y):

valid = False

return valid

def compare\_target(self, x1, x2, target):

if self.neuron\_activate(x1, x2) == target:

return True

else:

return False

def neuron\_activate(self, x1, x2):

output = self.w1\*x1 + self.w2\*x2

if output >= self.threshold:

return 1

else:

return 0

if \_\_name\_\_=="\_\_main\_\_":

AND\_Matrix = [

[-1, -1, 0],

[-1, 1, 0],

[ 1, -1, 0],

[ 1, 1, 1],

]

OR\_Matrix = [

[-1, -1, 0],

[-1, 1, 1],

[ 1, -1, 1],

[ 1, 1, 1],

]

NAND\_Matrix = [

[-1, -1, 1],

[-1, 1, 1],

[ 1, -1, 1],

[ 1, 1, 0],

]

XOR\_Matrix = [

[-1, -1, 0],

[-1, 1, 1],

[ 1, -1, 1],

[ 1, 1, 0],

]

def neuron\_calculate(mp):

if mp.iterate\_all\_values():

print("Weights are : {}, {}".format(mp.w1, mp.w2))

print("Threshold is {}".format(mp.threshold))

else:

print("Not linearly separable.")

print()

print("++ AND Gate ++")

mp\_AND = MP\_Neuron(AND\_Matrix)

neuron\_calculate(mp\_AND)

print("++ OR Gate ++")

mp\_OR = MP\_Neuron(OR\_Matrix)

neuron\_calculate(mp\_OR)

print("++ NAND Gate ++")

mp\_NAND = MP\_Neuron(NAND\_Matrix)

neuron\_calculate(mp\_NAND)

print("++ XOR Gate ++")

mp\_XOR = MP\_Neuron(XOR\_Matrix)

neuron\_calculate(mp\_XOR)