```
import numpy as np
```

```
## AND 연산
inputs = [(0,0),(0,1),(1,0),(1,1)]  # 튜플로 x 설정
w1, w2, b = 0.5,0.5,-0.7   # 가중치와 y절편(bias)
for x1, x2 in inputs :
  y = w1*x1 + w2*x2 + b
  if y <= 0 :
    y = 0
  else :
    y = 1

  print('({x1},{x2}) => {y}'.format(x1=x1, x2=x2, y=y))
```

```
(0,0) => 0
(0,1) => 0
(1,0) => 0
(1,1) => 1
```

```
## NAND 연산 : 가중치 -0.5, bias = -0.7
## OR 연산 : 가중치 0.5, bias = -0.2
```

```
def AND(x1, x2) :
  x = np.array([x1,x2])
  w = np.array([0.5,0.5])
  b = -0.7
  y = np.sum(x*w) + b
  if y <= 0 :
    return 0
  else :
    return 1

def NAND(x1, x2) :
  x = np.array([x1,x2])
  w = np.array([-0.5,-0.5])
  b = 0.7
  y = np.sum(x*w)+b
  if y <= 0 :
    return 0
  else :
    return 1

def OR(x1, x2) :
  x = np.array([x1,x2])
  w = np.array([0.5,0.5])
  b = -0.2
  y = np.sum(x*w)+b
  if y<= 0 :
    return 0
  else :
    return 1
```

```python
## multi perceptron ##

def XOR(x1,x2) :
  s1 = NAND(x1,x2)
  s2 = OR(x1,x2)
  y = AND(s1,s2)
  return y

inputs =([0,0],[1,0],[0,1],[1,1])

print('AND :')
for x1,x2 in inputs :
  y1= AND(x1,x2)
  print('({x1},{x2}) -> {y}'.format(x1=x1, x2=x2, y=y1))

print('NAND :')
for x1,x2 in inputs :
  y = NAND(x1,x2)
  print('({x1},{x2}) -> {y}'.format(x1=x1, x2=x2, y=y))

print('OR :')
for x1,x2 in inputs :
  y = OR(x1,x2)
  print('({x1},{x2}) -> {y}'.format(x1=x1,x2=x2,y=y))

print('XOR :')
for x1, x2 in inputs :
  y = XOR(x1,x2)
  print('({x1},{x2}) -> {y}'.format(x1=x1, x2=x2, y=y))
```

```
AND :
(0,0) -> 0
(1,0) -> 0
(0,1) -> 0
(1,1) -> 1
NAND :
(0,0) -> 1
(1,0) -> 1
(0,1) -> 1
(1,1) -> 0
OR :
(0,0) -> 0
(1,0) -> 1
(0,1) -> 1
(1,1) -> 1
XOR :
(0,0) -> 0
(1,0) -> 1
(0,1) -> 1
(1,1) -> 0
```

```python
# CS231N
# https://tykimos.github.io/lecture/
# 역전파
# optimizer : 가중치를 찾는 과정
# 새 가중치 = 기존 가중치 - n(learning rate) * 편미분 E/w(gradient)

# gradient 어떤 방향으로 학습?
# learning rate 얼마나 학습시킬지
```

```
0
```