In [4]:

```
import numpy as np
```

1번

In [5]:

```
x = 2
y = x**3 + x
dfdx = 2*y
dydx = 3*x**2 + 1

dfdx * dydx
```

Out[5]:

260

In [6]:

```
x = 3; y= -4; z = 2; w = 1;
p = x*y
q = z+w
f = p * q
print(f'p : {p}, q : {q},f : {f}')

dfdp = q
dfdq = p

dpdx = y
dpdy = x
dqdz = 1
dqdw = 1

dfdx = dfdp * dpdx
dfdy = dfdp * dpdy
dfdz = dfdq * dqdz
dfdw = dfdq * dqdw
print(f'dfdx : {dfdx}, dfdy : {dfdy}')
print(f'dfdz : {dfdz}, dfdw : {dfdw}')
```

```
p : -12, q : 3,f : -36
dfdx : -12, dfdy : 9
dfdz : -12, dfdw : -12
```

In [7]:

```
def relu(x):
    return (x>0)*x
```

In [8]:

```
input_ = np.array([1.0,0.5])
input_w = np.array([[0.9,0.3],[0.2,0.8]])
output = np.dot(input_,w)
final = relu(output)
print(final)
```

```
[1.  0.5]
```

In [9]:

```
target=np.array([1.85,0.1]) # 목표값
error = abs(target-output) # error의 절대값
w = input_w.copy()
normalize = np.array([[w[0,0]/(w[0,0]+w[1,0]),w[0,1]/(w[0,1]+w[1,1])],
                      [w[1,0]/(w[1,0]+w[0,0]),w[1,1]/(w[0,1]+w[1,1])]]) # 정규화
hidden_error = np.dot(normalize,error)
hidden_error
```

Out[9]:

```
array([0.80454545, 0.44545455])
```

In [10]:

```
# tmp
learning_rate=0.1
act_func_div = 1 # activation function 미분
input_w -= learning_rate*np.dot(-error*act_func_div * np.dot(w,output),input_w.T)
input_w
```

Out[10]:

```
array([[0.987525, 0.33705 ],
       [0.287525, 0.83705 ]])
```

In [ ]:

```
# y=2x 미분하면 2가 됨, 속미분으로 input값 곱해주기
```