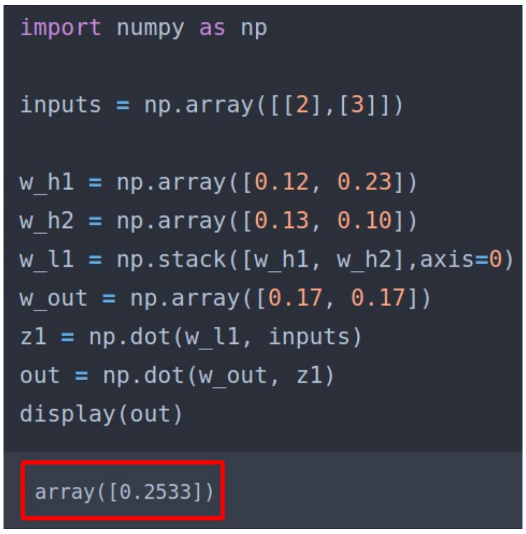
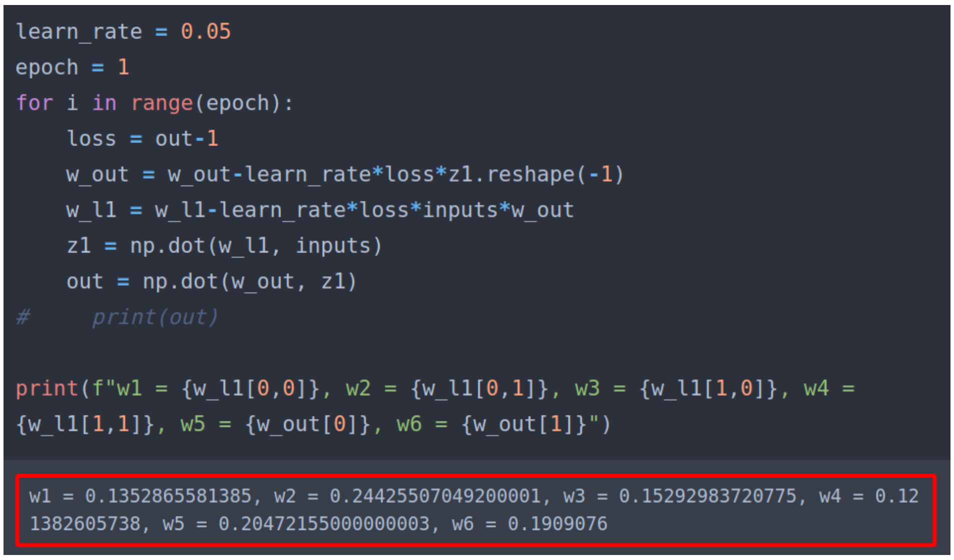
1. 把 w_1、w_2、w_3、w_4、w_5、w_6 分別設定為 0.12、0.23、0.13、0.10、0.17 和 0.17 模型針對(2,3) 的輸出是:
   - 

```python
import numpy as np

inputs = np.array([[2],[3]])

w_h1 = np.array([0.12, 0.23])
w_h2 = np.array([0.13, 0.10])
w_l1 = np.stack([w_h1, w_h2],axis=0)
w_out = np.array([0.17, 0.17])
z1 = np.dot(w_l1, inputs)
out = np.dot(w_out, z1)
display(out)
```

```
array([0.2533])
```

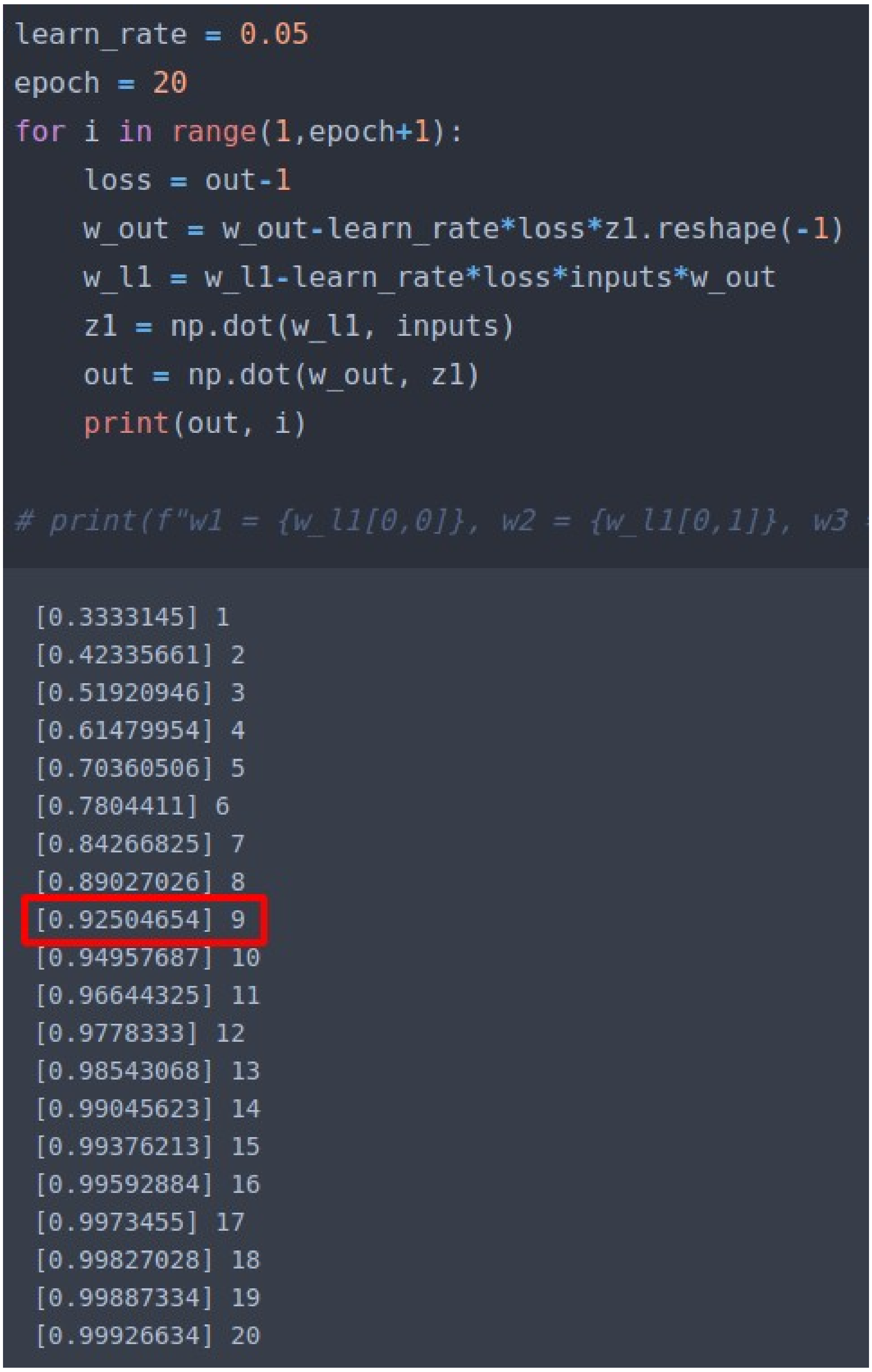
2. 第三版的權重對於 (2,3) 這一組輸入所產生輸出值是多少?
   - 

```python
learn_rate = 0.05
epoch = 1
for i in range(epoch):
    loss = out-1
    w_out = w_out-learn_rate*loss*z1.reshape(-1)
    w_l1 = w_l1-learn_rate*loss*inputs*w_out
    z1 = np.dot(w_l1, inputs)
    out = np.dot(w_out, z1)
#     print(out)

print(f"w1 = {w_l1[0,0]}, w2 = {w_l1[0,1]}, w3 = {w_l1[1,0]}, w4 =
{w_l1[1,1]}, w5 = {w_out[0]}, w6 = {w_out[1]}")
```

```
w1 = 0.1352865581385, w2 = 0.24425507049200001, w3 = 0.15292983720775, w4 = 0.12
1382605738, w5 = 0.20472155000000003, w6 = 0.1909076
```

3. 持續這樣的遞迴(疊代)，總共要更新幾次之後，所得的模型才會"足夠"接近 1？
   - 如果誤差要求在 0.1 以內的話，9 次迭代即可



```python
learn_rate = 0.05
epoch = 20
for i in range(1,epoch+1):
    loss = out-1
    w_out = w_out-learn_rate*loss*z1.reshape(-1)
    w_l1 = w_l1-learn_rate*loss*inputs*w_out
    z1 = np.dot(w_l1, inputs)
    out = np.dot(w_out, z1)
    print(out, i)

# print(f"w1 = {w_l1[0,0]}, w2 = {w_l1[0,1]}, w3
```

```
[0.3333145] 1
[0.42335661] 2
[0.51920946] 3
[0.61479954] 4
[0.70360506] 5
[0.7804411] 6
[0.84266825] 7
[0.89027026] 8
[0.92504654] 9
[0.94957687] 10
[0.96644325] 11
[0.9778333] 12
[0.98543068] 13
[0.99045623] 14
[0.99376213] 15
[0.99592884] 16
[0.9973455] 17
[0.99827028] 18
[0.99887334] 19
[0.99926634] 20
```