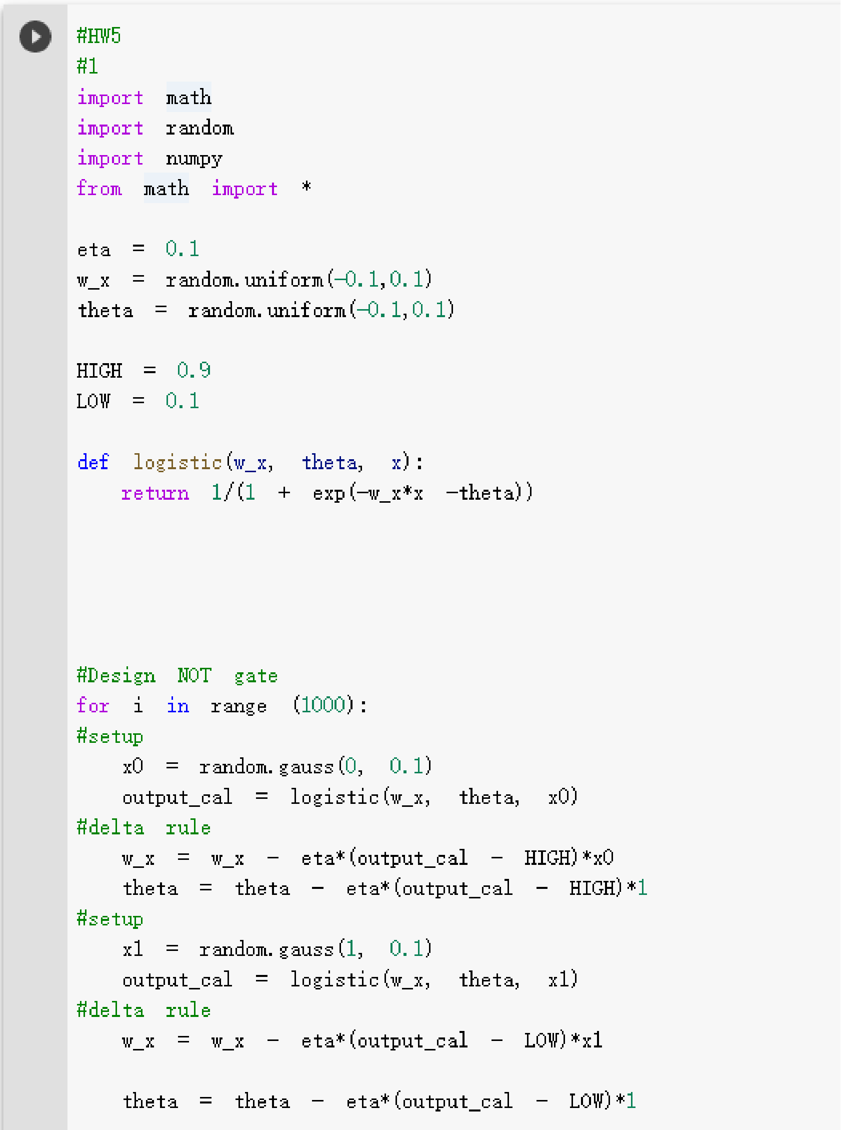
HW6

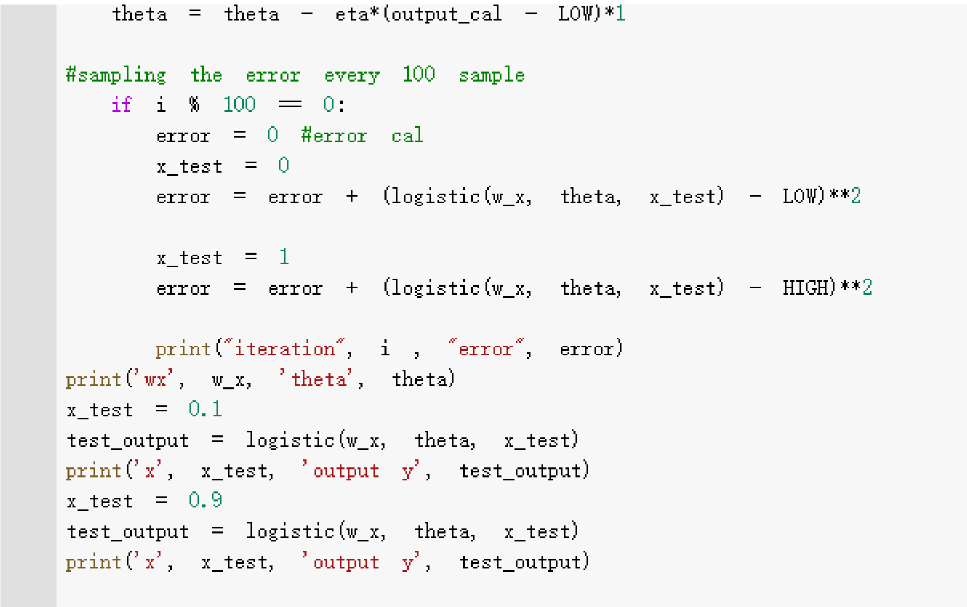
108034058

鍾岷翰

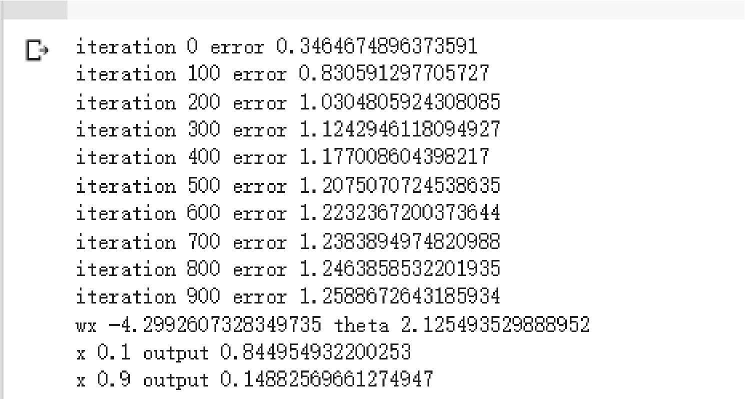
1.



```
#HW5
#1
import math
import random
import numpy
from math import *

eta = 0.1
w_x = random.uniform(-0.1,0.1)
theta = random.uniform(-0.1,0.1)

HIGH = 0.9
LOW = 0.1

def logistic(w_x, theta, x):
    return 1/(1 + exp(-w_x*x -theta))




#Design NOT gate
for i in range (1000):
#setup
    x0 = random.gauss(0, 0.1)
    output_cal = logistic(w_x, theta, x0)
#delta rule
    w_x = w_x - eta*(output_cal - HIGH)*x0
    theta = theta - eta*(output_cal - HIGH)*1
#setup
    x1 = random.gauss(1, 0.1)
    output_cal = logistic(w_x, theta, x1)
#delta rule
    w_x = w_x - eta*(output_cal - LOW)*x1

    theta = theta - eta*(output_cal - LOW)*1
```

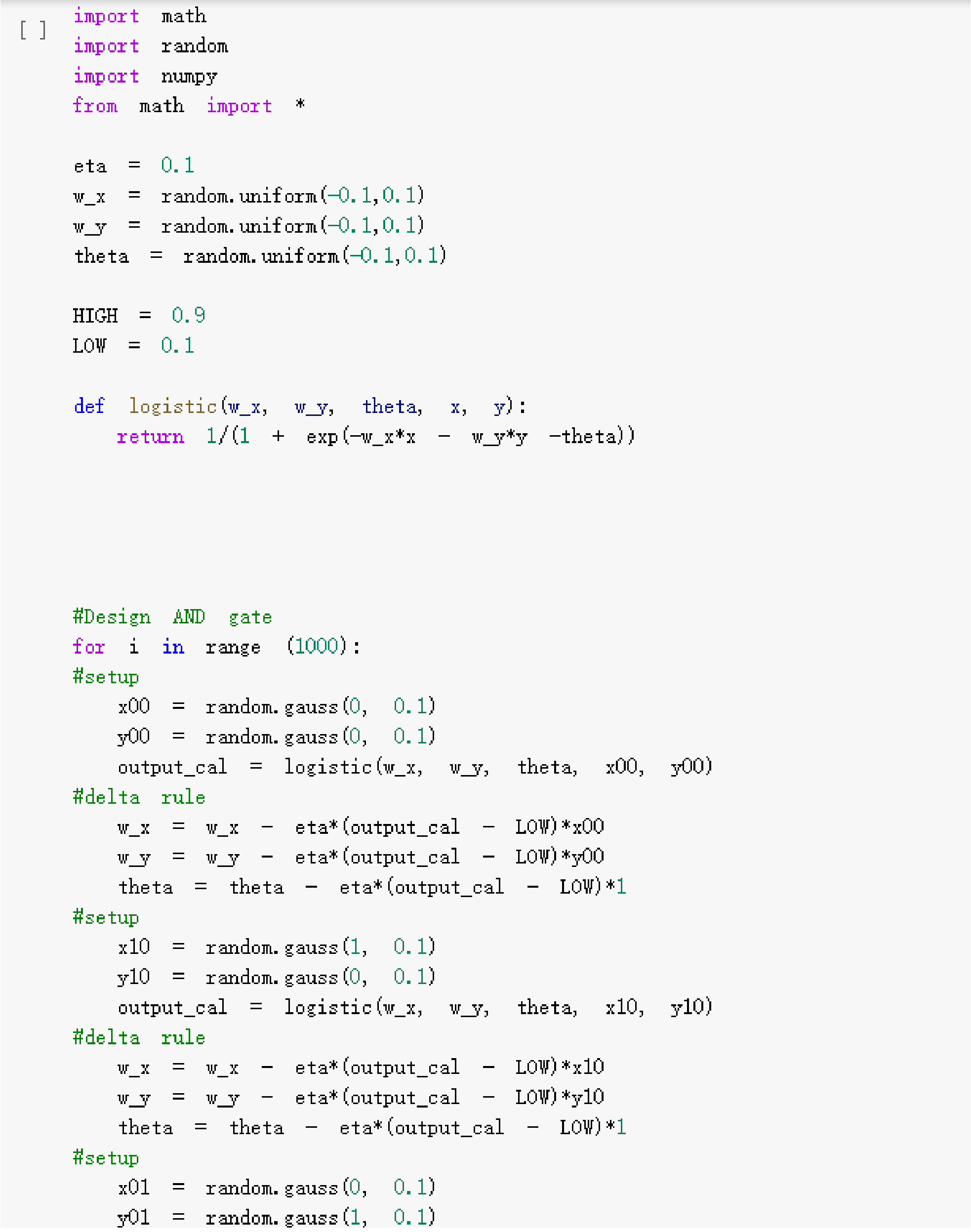


```
    theta = theta - eta*(output_cal - LOW)*1

#sampling the error every 100 sample
    if i % 100 == 0:
        error = 0 #error cal
        x_test = 0
        error = error + (logistic(w_x, theta, x_test) - LOW)**2

        x_test = 1
        error = error + (logistic(w_x, theta, x_test) - HIGH)**2

        print("iteration", i , "error", error)
print('wx', w_x, 'theta', theta)
x_test = 0.1
test_output = logistic(w_x, theta, x_test)
print('x', x_test, 'output y', test_output)
x_test = 0.9
test_output = logistic(w_x, theta, x_test)
print('x', x_test, 'output y', test_output)
```

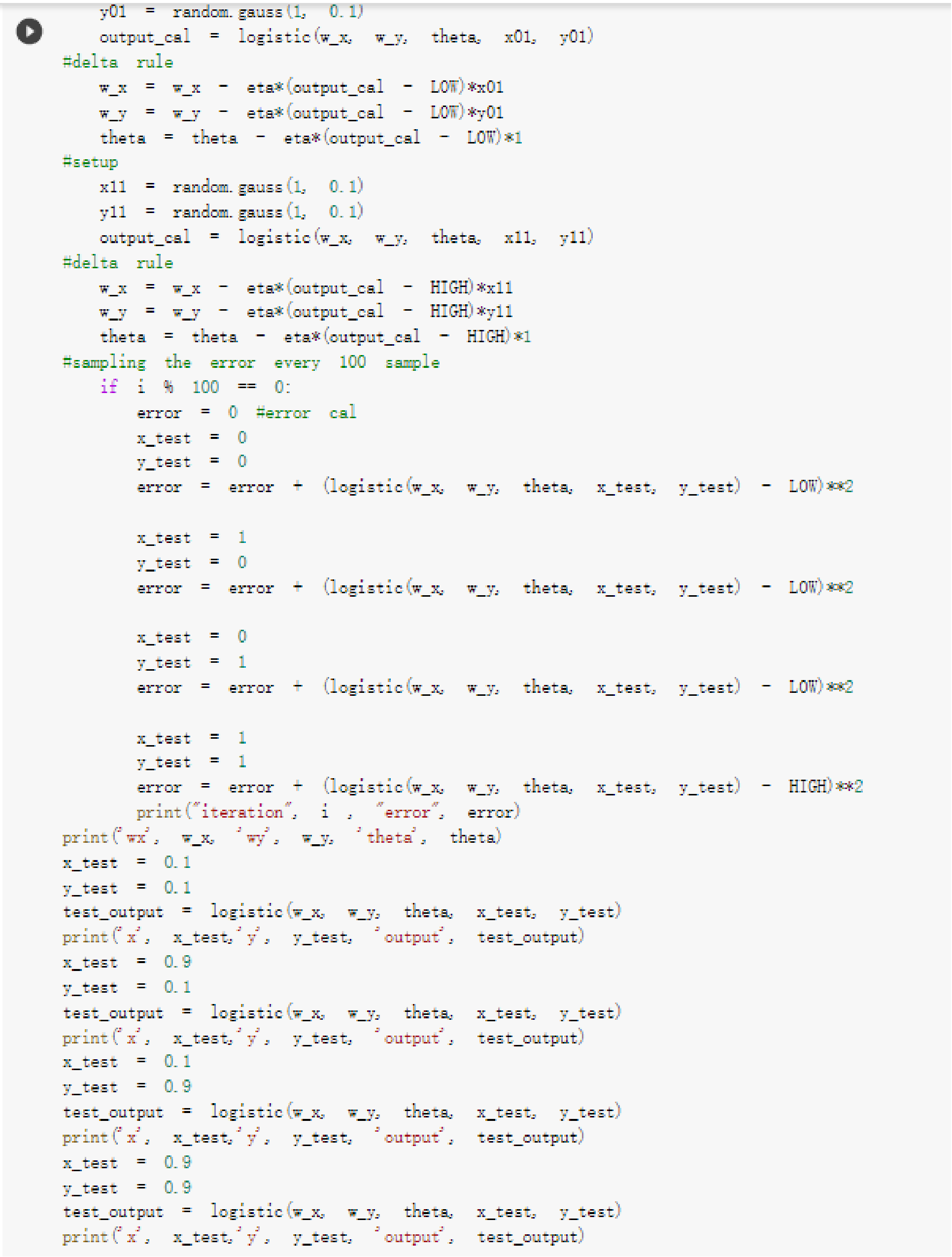


```
iteration 0 error 0.3464674896373591
iteration 100 error 0.830591297705727
iteration 200 error 1.0304805924308085
iteration 300 error 1.1242946118094927
iteration 400 error 1.177008604398217
iteration 500 error 1.2075070724538635
iteration 600 error 1.2232367200373644
iteration 700 error 1.2383894974820988
iteration 800 error 1.2463858532201935
iteration 900 error 1.2588672643185934
wx -4.2992607328349735 theta 2.125493529888952
x 0.1 output 0.844954932200253
x 0.9 output 0.14882569661274947
```

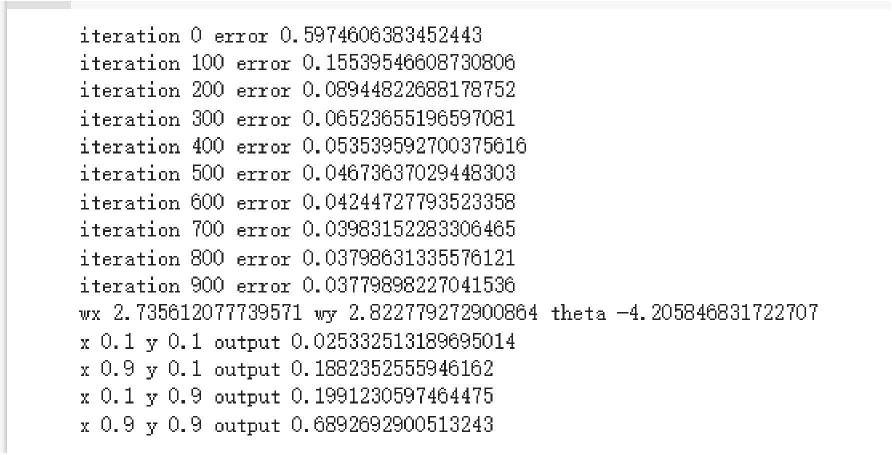
2.



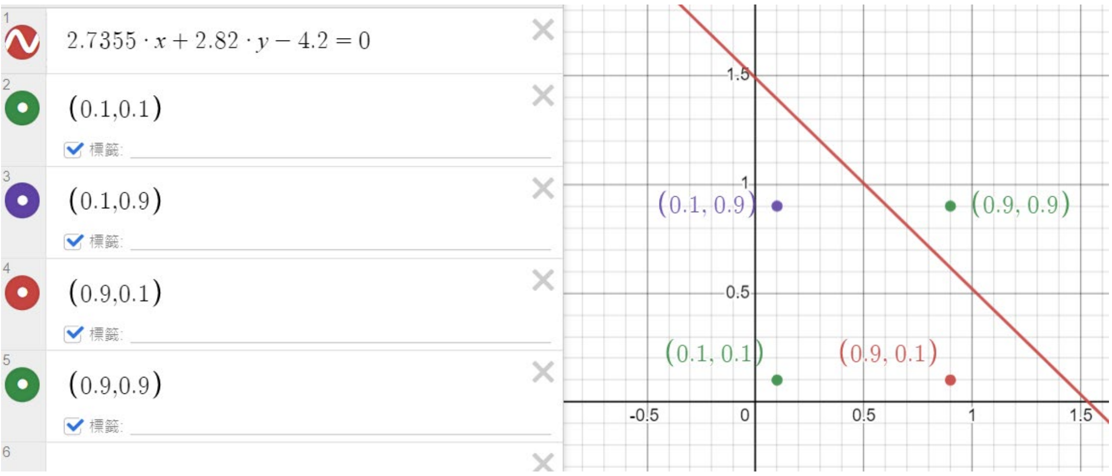
```
import math
import random
import numpy
from math import *

eta = 0.1
w_x = random.uniform(-0.1,0.1)
w_y = random.uniform(-0.1,0.1)
theta = random.uniform(-0.1,0.1)

HIGH = 0.9
LOW = 0.1

def logistic(w_x, w_y, theta, x, y):
    return 1/(1 + exp(-w_x*x - w_y*y -theta))




#Design AND gate
for i in range (1000):
#setup
    x00 = random.gauss(0, 0.1)
    y00 = random.gauss(0, 0.1)
    output_cal = logistic(w_x, w_y, theta, x00, y00)
#delta rule
    w_x = w_x - eta*(output_cal - LOW)*x00
    w_y = w_y - eta*(output_cal - LOW)*y00
    theta = theta - eta*(output_cal - LOW)*1
#setup
    x10 = random.gauss(1, 0.1)
    y10 = random.gauss(0, 0.1)
    output_cal = logistic(w_x, w_y, theta, x10, y10)
#delta rule
    w_x = w_x - eta*(output_cal - LOW)*x10
    w_y = w_y - eta*(output_cal - LOW)*y10
    theta = theta - eta*(output_cal - LOW)*1
#setup
    x01 = random.gauss(0, 0.1)
    y01 = random.gauss(1, 0.1)
```



```
    y01 = random.gauss(1, 0.1)
    output_cal = logistic(w_x, w_y, theta, x01, y01)
#delta rule
    w_x = w_x - eta*(output_cal - LOW)*x01
    w_y = w_y - eta*(output_cal - LOW)*y01
    theta = theta - eta*(output_cal - LOW)*1
#setup
    x11 = random.gauss(1, 0.1)
    y11 = random.gauss(1, 0.1)
    output_cal = logistic(w_x, w_y, theta, x11, y11)
#delta rule
    w_x = w_x - eta*(output_cal - HIGH)*x11
    w_y = w_y - eta*(output_cal - HIGH)*y11
    theta = theta - eta*(output_cal - HIGH)*1
#sampling the error every 100 sample
    if i % 100 == 0:
        error = 0 #error cal
        x_test = 0
        y_test = 0
        error = error + (logistic(w_x, w_y, theta, x_test, y_test) - LOW)**2

        x_test = 1
        y_test = 0
        error = error + (logistic(w_x, w_y, theta, x_test, y_test) - LOW)**2

        x_test = 0
        y_test = 1
        error = error + (logistic(w_x, w_y, theta, x_test, y_test) - LOW)**2

        x_test = 1
        y_test = 1
        error = error + (logistic(w_x, w_y, theta, x_test, y_test) - HIGH)**2
        print("iteration", i , "error", error)
print('wx', w_x, 'wy', w_y, 'theta', theta)
x_test = 0.1
y_test = 0.1
test_output = logistic(w_x, w_y, theta, x_test, y_test)
print('x', x_test,'y', y_test, 'output', test_output)
x_test = 0.9
y_test = 0.1
test_output = logistic(w_x, w_y, theta, x_test, y_test)
print('x', x_test,'y', y_test, 'output', test_output)
x_test = 0.1
y_test = 0.9
test_output = logistic(w_x, w_y, theta, x_test, y_test)
print('x', x_test,'y', y_test, 'output', test_output)
x_test = 0.9
y_test = 0.9
test_output = logistic(w_x, w_y, theta, x_test, y_test)
print('x', x_test,'y', y_test, 'output', test_output)
```



```
iteration 0 error 0.5974606383452443
iteration 100 error 0.15539546608730806
iteration 200 error 0.08944822688178752
iteration 300 error 0.06523655196597081
iteration 400 error 0.053539592700375616
iteration 500 error 0.04673637029448303
iteration 600 error 0.04244727793523358
iteration 700 error 0.03983152283306465
iteration 800 error 0.03798631335576121
iteration 900 error 0.03779898227041536
wx 2.735612077739571 wy 2.822779272900864 theta -4.205846831722707
x 0.1 y 0.1 output 0.025332513189695014
x 0.9 y 0.1 output 0.1882352555946162
x 0.1 y 0.9 output 0.1991230597464475
x 0.9 y 0.9 output 0.6892692900513243
```



1 $2.7355 \cdot x + 2.82 \cdot y - 4.2 = 0$

2 $(0.1, 0.1)$

3 $(0.1, 0.9)$

4 $(0.9, 0.1)$

5 $(0.9, 0.9)$

6