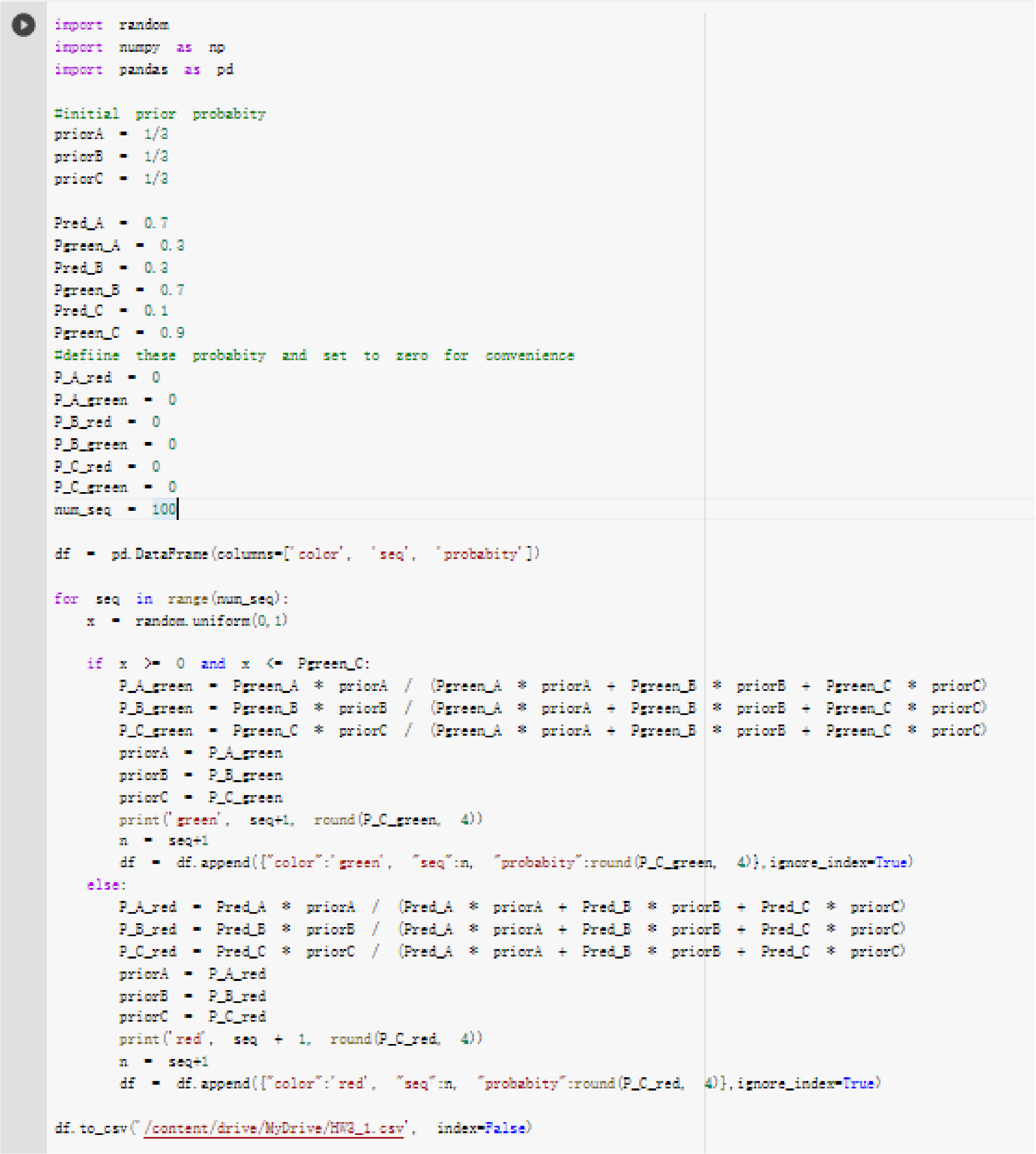
HW3

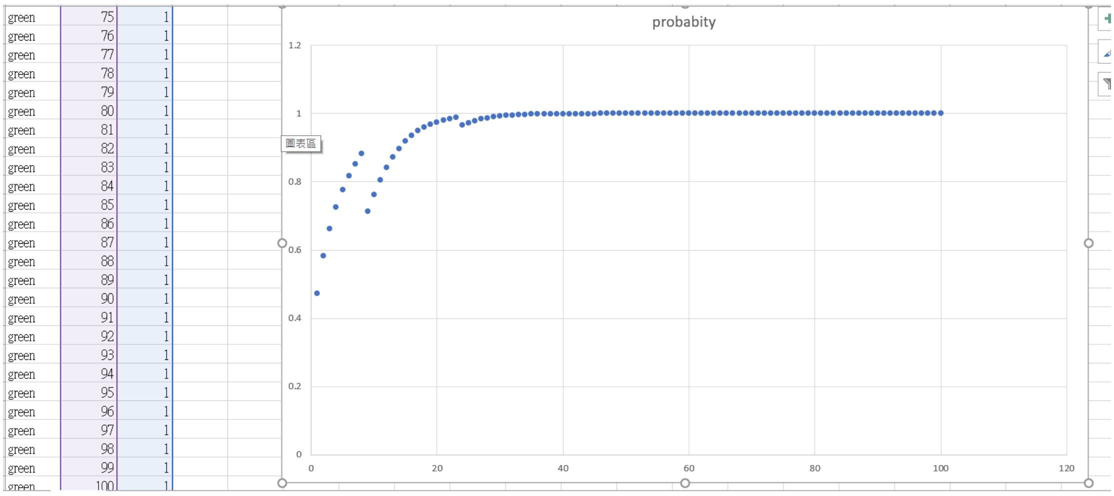
108034058

鍾岷翰

1.



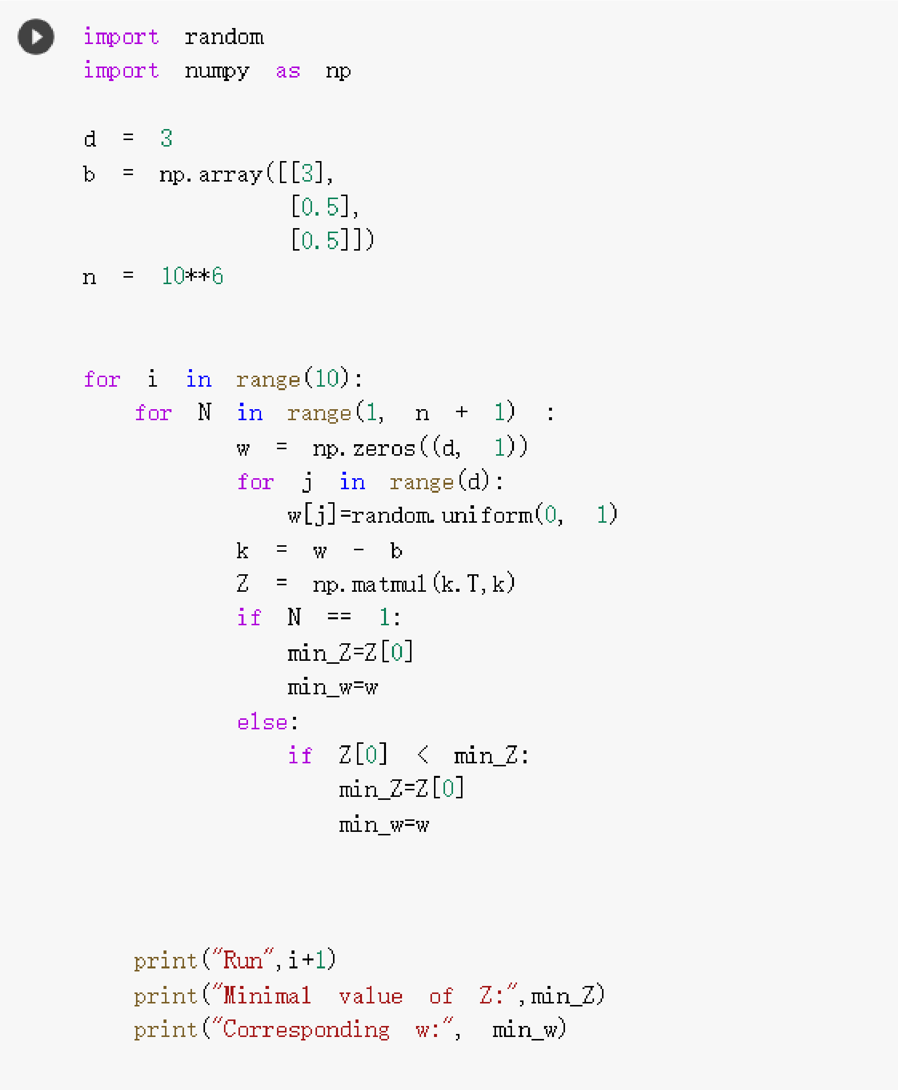
```python
import random
import numpy as np
import pandas as pd

#initial prior probabity
priorA = 1/3
priorB = 1/3
priorC = 1/3

Pred_A = 0.7
Pgreen_A = 0.3
Pred_B = 0.3
Pgreen_B = 0.7
Pred_C = 0.1
Pgreen_C = 0.9
#defiine these probabity and set to zero for convenience
P_A_red = 0
P_A_green = 0
P_B_red = 0
P_B_green = 0
P_C_red = 0
P_C_green = 0
num_seq = 100

df = pd.DataFrame(columns=['color', 'seq', 'probabity'])

for seq in range(num_seq):
    x = random.uniform(0,1)

    if x >= 0 and x <= Pgreen_C:
        P_A_green = Pgreen_A * priorA / (Pgreen_A * priorA + Pgreen_B * priorB + Pgreen_C * priorC)
        P_B_green = Pgreen_B * priorB / (Pgreen_A * priorA + Pgreen_B * priorB + Pgreen_C * priorC)
        P_C_green = Pgreen_C * priorC / (Pgreen_A * priorA + Pgreen_B * priorB + Pgreen_C * priorC)
        priorA = P_A_green
        priorB = P_B_green
        priorC = P_C_green
        print('green', seq+1, round(P_C_green, 4))
        n = seq+1
        df = df.append({"color":'green', "seq":n, "probabity":round(P_C_green, 4)},ignore_index=True)
    else:
        P_A_red = Pred_A * priorA / (Pred_A * priorA + Pred_B * priorB + Pred_C * priorC)
        P_B_red = Pred_B * priorB / (Pred_A * priorA + Pred_B * priorB + Pred_C * priorC)
        P_C_red = Pred_C * priorC / (Pred_A * priorA + Pred_B * priorB + Pred_C * priorC)
        priorA = P_A_red
        priorB = P_B_red
        priorC = P_C_red
        print('red', seq + 1, round(P_C_red, 4))
        n = seq+1
        df = df.append({"color":'red', "seq":n, "probabity":round(P_C_red, 4)},ignore_index=True)

df.to_csv('/content/drive/MyDrive/HW3_1.csv', index=False)
```



green	75	1
green	76	1
green	77	1
green	78	1
green	79	1
green	80	1
green	81	1
green	82	1
green	83	1
green	84	1
green	85	1
green	86	1
green	87	1
green	88	1
green	89	1
green	90	1
green	91	1
green	92	1
green	93	1
green	94	1
green	95	1
green	96	1
green	97	1
green	98	1
green	99	1
green	100	1

2.

(a)



```
import random
import numpy as np

d = 3
b = np.array([[3],
              [0.5],
              [0.5]])
n = 10**6


for i in range(10):
    for N in range(1, n + 1) :
        w = np.zeros((d, 1))
        for j in range(d):
            w[j]=random.uniform(0, 1)
        k = w - b
        Z = np.matmul(k.T,k)
        if N == 1:
            min_Z=Z[0]
            min_w=w
        else:
            if Z[0] < min_Z:
                min_Z=Z[0]
                min_w=w



    print("Run",i+1)
    print("Minimal value of Z:",min_Z)
    print("Corresponding w:", min_w)
```

```
Run 1

Minimal value of Z: [4.00138847]
```

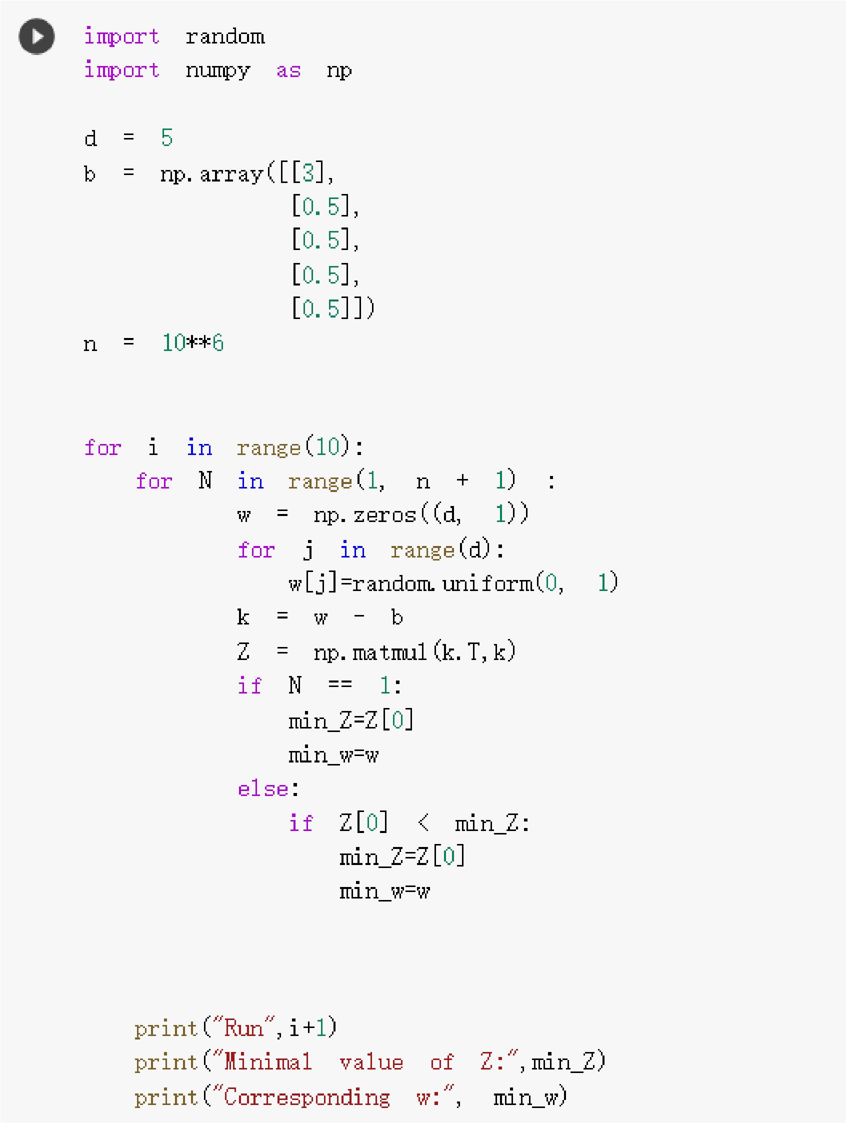
```
Corresponding w: [[0.99980643]
 [0.47580479]
 [0.49463876]]
Run 2
Minimal value of Z: [4.00052196]
Corresponding w: [[0.99987835]
 [0.49444905]
 [0.4978691 ]]
Run 3
Minimal value of Z: [4.00036959]
Corresponding w: [[0.99999511]
 [0.48148326]
 [0.50267734]]
Run 4
Minimal value of Z: [4.00241557]
Corresponding w: [[0.99946059]
 [0.51042159]
 [0.51220688]]
Run 5
Minimal value of Z: [4.00108955]
Corresponding w: [[0.99988299]
 [0.52004773]
 [0.48518112]]
Run 6
Minimal value of Z: [4.000575]
Corresponding w: [[0.99987269]
 [0.49491573]
 [0.50631735]]
Run 7
Minimal value of Z: [4.0011204]
Corresponding w: [[0.99972941]
 [0.49584742]
 [0.49544755]]
Run 8
Minimal value of Z: [4.00037229]
Corresponding w: [[0.99998685]
 [0.49195584]
 [0.51596794]]
```

```
Run 9
Minimal value of Z: [4.00176451]
Corresponding w: [[0.99984266]
 [0.51343011]
 [0.53089912]]
Run 10
Minimal value of Z: [4.00043057]
Corresponding w: [[0.99994454]
 [0.48557321]
 [0.49923499]]
```

(b)

由 3,0.5,0.5 這個點以圓心出發，並不會接觸到此範圍的端點。並且由題(a)可得知最短距離並非在端點上。

(c)



```python
import random
import numpy as np

d = 5
b = np.array([[3],
              [0.5],
              [0.5],
              [0.5],
              [0.5]])
n = 10**6


for i in range(10):
    for N in range(1, n + 1) :
        w = np.zeros((d, 1))
        for j in range(d):
            w[j]=random.uniform(0, 1)
        k = w - b
        Z = np.matmul(k.T,k)
        if N == 1:
            min_Z=Z[0]
            min_w=w
        else:
            if Z[0] < min_Z:
                min_Z=Z[0]
                min_w=w



    print("Run",i+1)
    print("Minimal value of Z:",min_Z)
    print("Corresponding w:", min_w)
```

```
Run 1
Minimal value of Z: [4.01018901]
Corresponding w: [[0.99965185]
 [0.56641203]
 [0.49197593]
 [0.44198602]
 [0.46908503]]
Run 2
Minimal value of Z: [4.02361917]
Corresponding w: [[0.99936062]
 [0.49557217]
 [0.61628046]
 [0.58519612]
 [0.51618928]]
Run 3
Minimal value of Z: [4.00626693]
Corresponding w: [[0.99942141]
 [0.46856787]
 [0.54233232]
 [0.48617531]
 [0.5313226 ]]
Run 4
Minimal value of Z: [4.01746627]
Corresponding w: [[0.99899952]
 [0.48234813]
 [0.41742523]
 [0.42584333]
 [0.52887836]]
Run 5
Minimal value of Z: [4.01321197]
Corresponding w: [[0.99703192]
 [0.51505635]
 [0.51295715]
 [0.52472619]
 [0.51802467]]
Run 6
Minimal value of Z: [4.00782272]
Corresponding w: [[0.99909344]
```

```
 [0.51206274]
 [0.50566598]
 [0.47862839]
 [0.44032324]]
Run 7
Minimal value of Z: [4.00339092]
Corresponding w: [[0.99938546]
 [0.48087851]
 [0.50503177]
 [0.50669911]
 [0.52228324]]
Run 8
Minimal value of Z: [4.01641293]
Corresponding w: [[0.99705143]
 [0.43756189]
 [0.49328362]
 [0.52315809]
 [0.51140316]]
Run 9
Minimal value of Z: [4.01033131]
Corresponding w: [[0.99983572]
 [0.57091717]
 [0.51941806]
 [0.54972146]
 [0.45762514]]
Run 10
Minimal value of Z: [4.01642261]
Corresponding w: [[0.99802915]
 [0.54564286]
 [0.54452834]
 [0.4457374 ]
 [0.53904932]]
```