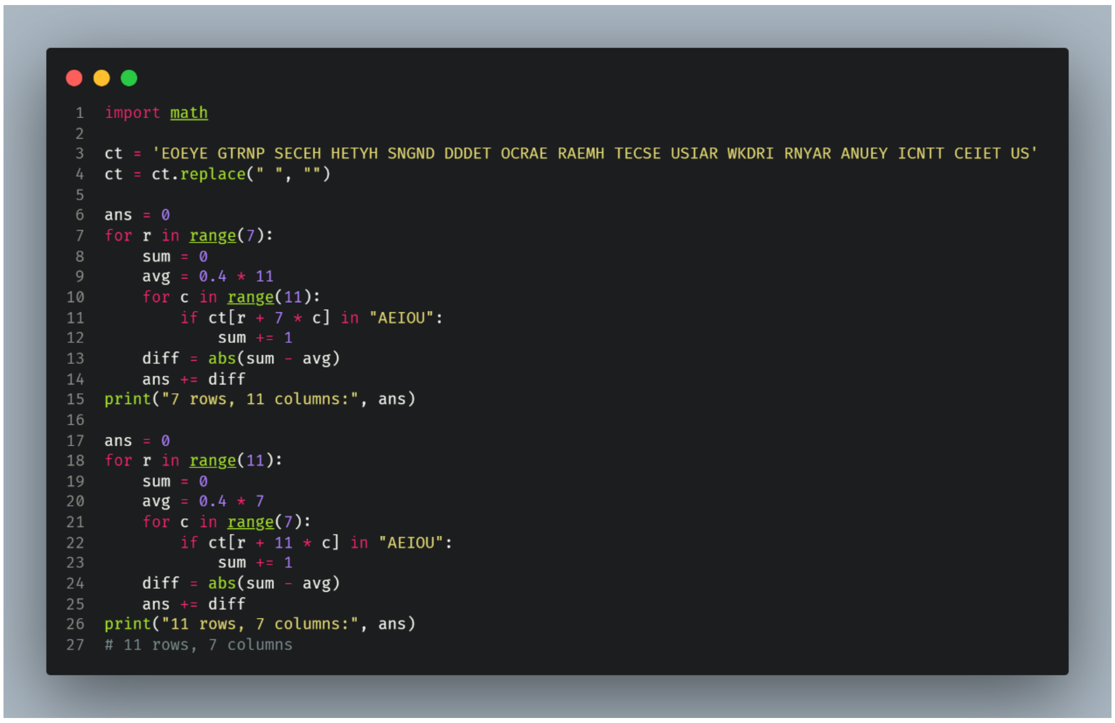
密碼工程 quiz4

109550087 單宇晟



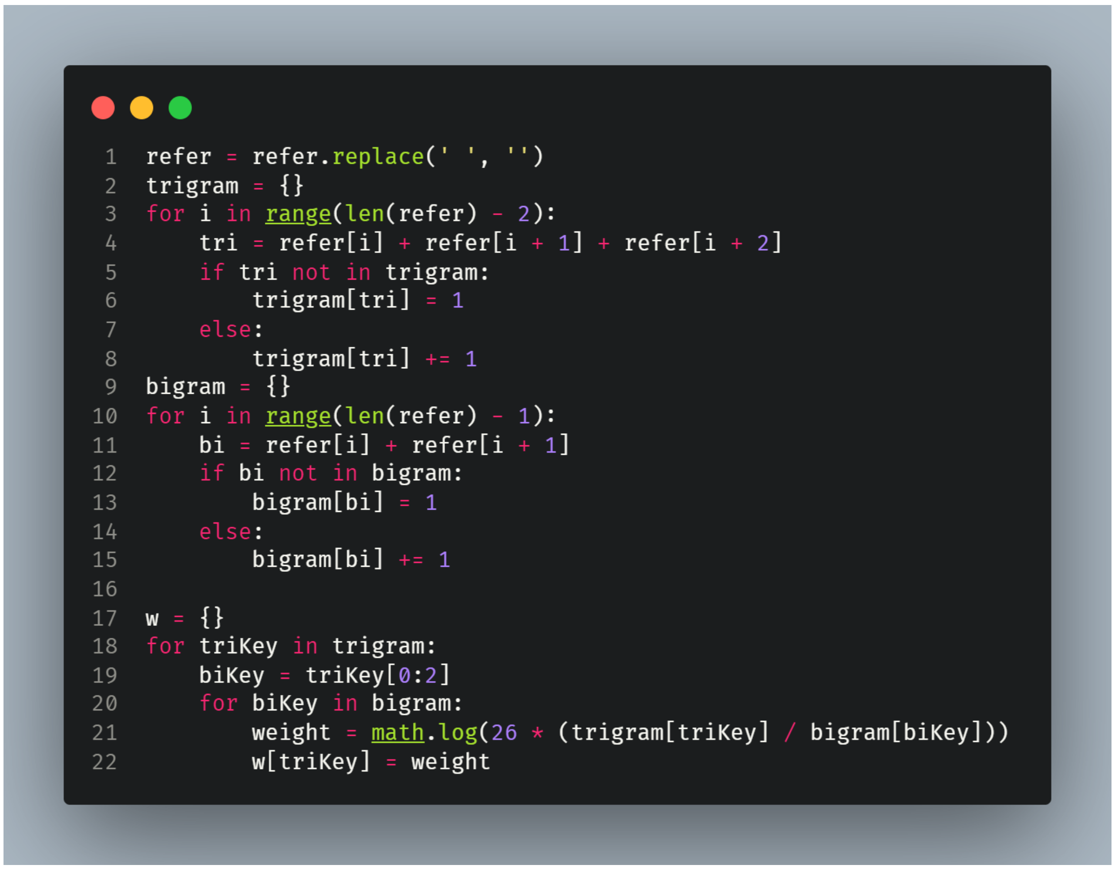
```python
import math

ct = 'EOEYE GTRNP SECEH HETYH SNGND DDDET OCRAE RAEMH TECSE USIAR WKDRI RNYAR ANUEY ICNTT CEIET US'
ct = ct.replace(" ", "")

ans = 0
for r in range(7):
    sum = 0
    avg = 0.4 * 11
    for c in range(11):
        if ct[r + 7 * c] in "AEIOU":
            sum += 1
    diff = abs(sum - avg)
    ans += diff
print("7 rows, 11 columns:", ans)

ans = 0
for r in range(11):
    sum = 0
    avg = 0.4 * 7
    for c in range(7):
        if ct[r + 11 * c] in "AEIOU":
            sum += 1
    diff = abs(sum - avg)
    ans += diff
print("11 rows, 7 columns:", ans)
# 11 rows, 7 columns
```

首先，我們要先決定原文是幾乘幾的矩陣，這部分和 quiz2 十分相似，就檢查每個 row 裡的母音數量，再減去平均值(0.4 x 字串長度)，並把每個 row 的 difference 加總，計算出結果後，發現原文以 11 個 row, 7 個 column 排列的機會比較大(difference 比較小)



```python
refer = refer.replace(' ', '')
trigram = {}
for i in range(len(refer) - 2):
    tri = refer[i] + refer[i + 1] + refer[i + 2]
    if tri not in trigram:
        trigram[tri] = 1
    else:
        trigram[tri] += 1
bigram = {}
for i in range(len(refer) - 1):
    bi = refer[i] + refer[i + 1]
    if bi not in bigram:
        bigram[bi] = 1
    else:
        bigram[bi] += 1

w = {}
for triKey in trigram:
    biKey = triKey[0:2]
    for biKey in bigram:
        weight = math.log(26 * (trigram[triKey] / bigram[biKey]))
        w[triKey] = weight
```

(refer 是教授給的 training reference)

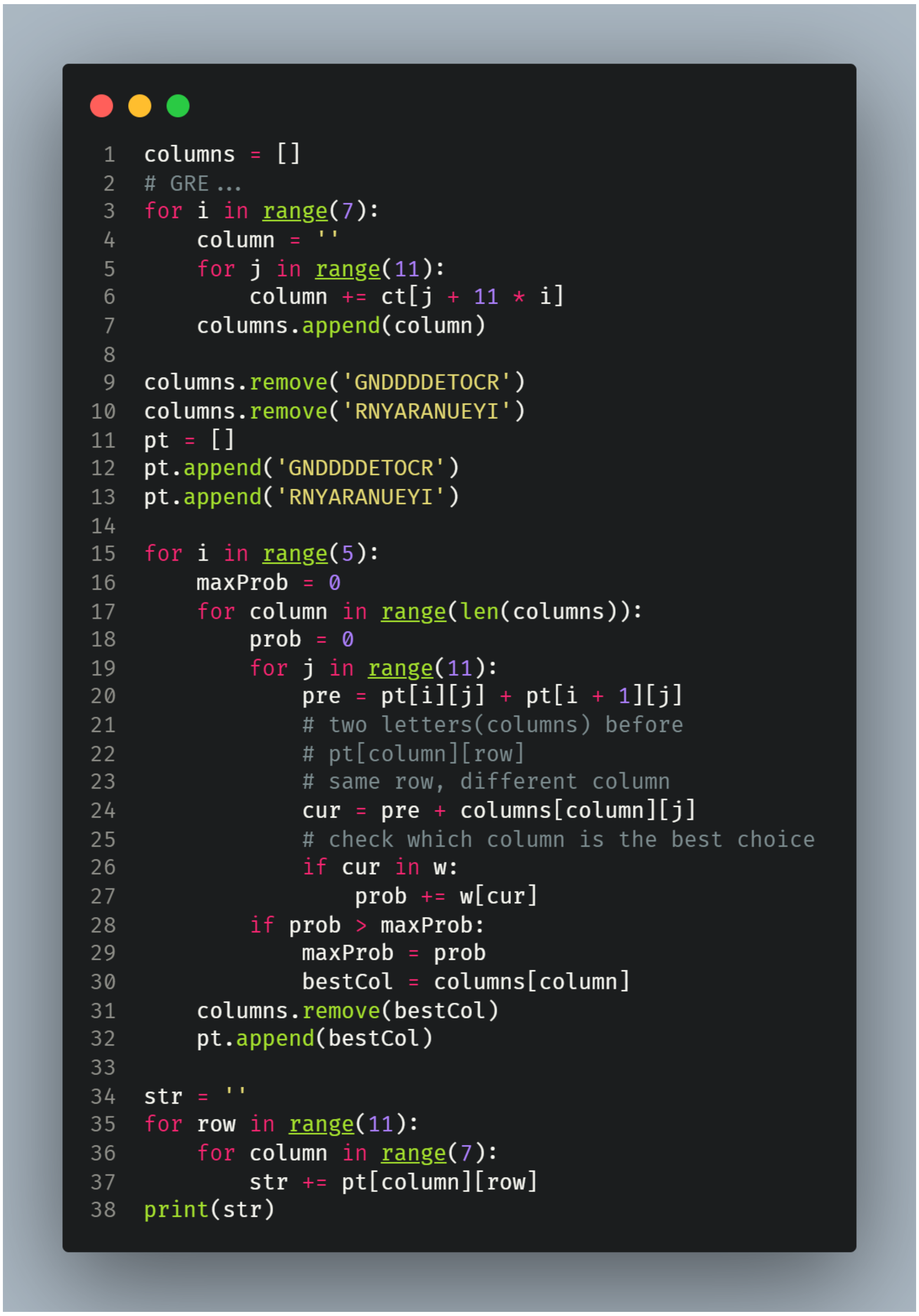
接著，我開始計算各個 trigram、bigram 出現的次數，並分別用一個 dictionary 存起來。完成之後，再根據教授給的公式--

W(THE)=

Log Pc(THE/TH) / Random

=log (A / A+B)/(1/26)

算出各個 trigram 的 weight，並且一樣用一個 dictionary 存起來


```python
columns = []
# GRE...
for i in range(7):
    column = ''
    for j in range(11):
        column += ct[j + 11 * i]
    columns.append(column)

columns.remove('GNDDDDETOCR')
columns.remove('RNYARANUEYI')
pt = []
pt.append('GNDDDDETOCR')
pt.append('RNYARANUEYI')

for i in range(5):
    maxProb = 0
    for column in range(len(columns)):
        prob = 0
        for j in range(11):
            pre = pt[i][j] + pt[i + 1][j]
            # two letters(columns) before
            # pt[column][row]
            # same row, different column
            cur = pre + columns[column][j]
            # check which column is the best choice
            if cur in w:
                prob += w[cur]
        if prob > maxProb:
            maxProb = prob
            bestCol = columns[column]
    columns.remove(bestCol)
    pt.append(bestCol)

str = ''
for row in range(11):
    for column in range(7):
        str += pt[column][row]
print(str)
```

最後，就到解密的環節了，首先把每個 column 用一個 list 存起來，並根據教授給的提示，明文是 GRE...，把 G 和 R 開頭的 column 移除，放到另一個 list(pt, plaintext)，而沒有把 E 開頭的也一起移除是因為有兩個 column 的開頭都是 E，所以這部份就需要用計算來決定哪個才是下一個 column。

接下來，我一個三層迴圈來解密，pre 代表的是前兩個字母，cur 則是現在要計算的三個字母，透過加總各個 column 的 weight，決定出哪個 column 最有可能是下一個，並把該 column 移出之後要繼續比較的 list(避免重複)，放入 pt 這個 list 裡，跑完之後，就可以得到明文了!

Plaintext:
GREECEANNOUNCEDYESTERDAYITHADREACHEDAGREEMENTWITHTURKEYTOENDTHECYPRUSCRISISNS