前言:

本次的主題響應前陣子學校舉行的繞圈賽，藉由邏輯思考、寫程式、和同學討論，求出最短的時間。我們這組要解出3人10圈以及4人16圈的最佳解，我們分別利用了Excel、Python、Scratch等寫出三組程式，再比較最後的結果，以及其中的方法。

目錄:

數學建構模型

規則:
1. 假設第一個人要跑n圈，他可以一趟跑完，也可以分幾趟跑完。
2. 第一個人如果要分兩趟跑，就要等所有人跑完第1趟，她才可以接著跑第2趟。
3. 因為他會越跑越累，假設趟疲比1.2、圈疲比1.1

題目1
若A跑一圈需要 40 秒，B跑一圈需要 44 秒，
C跑一圈需要 48 秒，3 人要合力跑完 10 圈，
這 3 人如何分配每人的跑步的圈數，讓完成 10 圈所需的時間最少?

Step 1 整體分析

討論整體最佳方式：
設 A 總圈數為 n_A 圈，B 總圈數為 n_B 圈，C 總圈數為 n_C 圈

顯然條件 $n_A \geq n_B \geq n_C \geq 1$（$\because$ 跑步速率快慢：$A > B > C$）

可能組合

n_A	4		5		6		7, 8
n_B	3	4	3	4	2	3	…
n_C	3	2	2	1	2	1	…

題目3

若A跑一圈需要 40 秒，B跑一圈需要 44 秒，C跑一圈需要 48 秒，
D跑一圈需要 52 秒 4 人要合力跑完 16 圈，這 4 人如何分配每人的跑步的圈數，讓完成 16 圈所需的時間最少?

Step 2 簡化討論

討論個人最佳方式（以 A 分兩次跑為例）：

設 A 第 1 次跑 n_1 圈、第 2 次跑 n_2 圈，所需總時間 $A(t) = A(n_1, n_2)$ (s)

顯然條件 $n_1 \geq n_2$（∵ 趟疲比 > 圈疲比）

可能組合

1. 當 $n_A = 4$ 時

   (1) $A(t) = A(2,2) = [\sum_{k=1}^{2}(1.1)^{k-1} + \sum_{k=1}^{2}(1.2)(1.1)^{k-1}] \times 40 = \frac{40[(1.1)^2-1]}{1.1-1} + \frac{48[(1.1)^2-1]}{1.1-1}$ (s)

   (2) $A(t) = A(3,1) = [\sum_{k=1}^{3}(1.1)^{k-1} + 1.2] \times 40 = \frac{40[(1.1)^3-1]}{1.1-1} + 48$ (s)

2. 當 $n_A = 5$ 時

   (1) $A(t) = A(3,2) = [\sum_{k=1}^{3}(1.1)^{k-1} + \sum_{k=1}^{2}(1.2)(1.1)^{k-1}] \times 40 = \frac{40[(1.1)^3-1]}{1.1-1} + \frac{48[(1.1)^2-1]}{1.1-1}$ (s)

   (2) $A(t) = A(4,1) = [\sum_{k=1}^{4}(1.1)^{k-1} + 1.2] \times 40 = \frac{40[(1.1)^4-1]}{1.1-1} + 48$ (s)

3. 當 $n_A = 6$ 時

   (1) $A(t) = A(3,3) = [\sum_{k=1}^{3}(1.1)^{k-1} + \sum_{k=1}^{3}(1.2)(1.1)^{k-1}] \times 40 = \frac{40[(1.1)^3-1]}{1.1-1} + \frac{48[(1.1)^3-1]}{1.1-1}$ (s)

   (2) $A(t) = A(4,2) = [\sum_{k=1}^{4}(1.1)^{k-1} + \sum_{k=1}^{2}(1.2)(1.1)^{k-1}] \times 40 = \frac{40[(1.1)^4-1]}{1.1-1} + \frac{48[(1.1)^2-1]}{1.1-1}$ (s)

   (3) $A(t) = A(5,1) = [\sum_{k=1}^{5}(1.1)^{k-1} + 1.2] \times 40 = \frac{40[(1.1)^5-1]}{1.1-1} + 48$ (s)

數學建構模型—小結

對接力繞圈賽這類數據資料龐大的最佳解問題，以我們這組的經驗而言，建立數學模型的目的並非直指如何解決，而是提供可能的方向。雖然列式過程繁瑣，但可以讓思考的脈絡更有層次；並且先過濾掉顯然不恰當的情形，提高後續討論的效率。

以Excel解題

題目1
若A跑一圈需要 40 秒，B跑一圈需要 44 秒，
C跑一圈需要 48 秒，3 人要合力跑完 10 圈，
這 3 人如何分配每人的跑步的圈數，
讓完成 10 圈所需的時間最少?
圈疲比1.1
趟疲比1.2

D16 {=SUM(SMALL(C4:E8,ROW(INDIRECT("1:10"))))}

	A	B	C	D	E	F	G	H	I	J
1										
2	Q1.		圈數							
3	3人10圈		1	2	3	4	5	6		
4	趟數	A1	40	44	48.4	53.24	58.564	64.4204		
5		B1	44	48.4	53.24	58.564	64.4204	70.86244		
6		C1	48	52.8	58.08	63.888	70.2768	77.30448		
7		A2	48	52.8	58.08	63.888	70.2768	77.30448		
8		B2	52.8	58.08	63.888	70.2768	77.30448	85.03493		
9		C2	57.6	63.36	69.696	76.6656	84.33216	92.76538		
10		A3	57.6	63.36	69.696	76.6656	84.33216	92.76538		
11		B3	63.36	69.696	76.6656	84.33216	92.76538	102.0419		
12		C3	69.12	76.032	83.6352	91.99872	101.1986	111.3185		
13		A4	69.12	76.032	83.6352	91.99872	101.1986	111.3185		
14		B4	76.032	83.6352	91.99872	101.1986	111.3185	122.4503		
15		C4	82.944	91.2384	100.3622	110.3985	121.4383	133.5821		
16	A(3,2)+B(2,1)+C(2)=			479.2						
17										

1. 利用公比跑出行和列的資料，再設定終止值避免無限延伸
2. 選取前十小的數值
3. 因為表格中無法直接加總，訂出另個公式
   =SUM(SMALL(始位:終位,ROW(INDIRECT(“1:圈”))))
4. 輸入公式後按下ctrl+shift+enter即可求出答案479.2

題目3
若A跑一圈需要 40 秒，B跑一圈需要 44 秒，C跑一圈需要 48 秒，
D跑一圈需要 52 秒 4 人要合力跑完 16 圈，這 4 人如何分配每人的跑步的圈數，讓完成 16 圈所需的時間最少?
圈疲比1.1
趟疲比1.2

F37 {=SUM(SMALL(C21:F27,ROW(INDIRECT("1:16"))))}

	A	B	C	D	E	F	G	H	I	J
23		C1	48	52.8	58.08	63.888	70.2768	77.30448		
24		D1	52	57.2	62.92	69.212	76.1332	83.74652		
25		A2	48	52.8	58.08	63.888	70.2768	77.30448		
26		B2	52.8	58.08	63.888	70.2768	77.30448	85.03493		
27		C2	57.6	63.36	69.696	76.6656	84.33216	92.76538		
28	趟數	D2	62.4	68.64	75.504	83.0544	91.35984	100.4958		
29		A3	57.6	63.36	69.696	76.6656	84.33216	92.76538		
30		B3	63.36	69.696	76.6656	84.33216	92.76538	102.0419		
31		C3	69.12	76.032	83.6352	91.99872	101.1986	111.3185		
32		D3	74.88	82.368	90.6048	99.66528	109.6318	120.595		
33		A4	69.12	76.032	83.6352	91.99872	101.1986	111.3185		
34		B4	76.032	83.6352	91.99872	101.1986	111.3185	122.4503		
35		C4	82.944	91.2384	100.3622	110.3985	121.4383	133.5821		
36		D4	89.856	98.8416	108.7258	119.5983	131.5582	144.714		
37	A(4,2)+B(3,1)+C(3,1)+D(2)					810.56				
38	A(4,3)+B(3,1)+C(2,1)+D(2)									
39	A(4,2)+B(3,2)+C(2,1)+D(2)									
40	A3不符規則,因為D2沒有跑									

1. 按照程式選出前十六小的值
2. 最小值不符合規則，進而找到次小值(綠色方格)
3. 次小值皆符合規則，求出三解810.56

程式邏輯分析
Step 1 利用表格求出各圈、趟的值
Step 2 代入公式，找出題目所需的最小值
Step 3 檢查其是否符合規則

以 Python 解題

Test 1

程式碼

```python
a,b,c,d = map(float, input("四個人的時間:").split())
x = float(input("圈疲:"))
y = float(input("趟疲："))
n = int(input("圈數:"))

p_list = []
for k in range(1, n):
    for m in range(1, n):
        p = a * (y)**(k-1) * (x)**(m-1)
        q = b * (y)**(k-1) * (x)**(m-1)
        r = c * (y)**(k-1) * (x)**(m-1)
        s = d * (y) ** (k - 1) * (x) ** (m - 1)
        p_list += [p, q, r, s]

p_list_sorted = sorted(p_list)[:int(n)]
result = sum(p_list_sorted)
print(result)
```

輸入

```
40 44 48 52
1.1
1.2
16
```

輸出

```
810.08(WA，答案為810.56)
```

與用 Excel 解出的答案不符

遭遇困難：

當有選手的單圈時間與其他人相距太大，使其秒數不被加入串列時
=> 違反規則一（每人都要跑，所有人跑完第一趟才能有人跑第二趟）
=> 造成誤差（輸出結果小於真正的最佳解）
∴ Test 1 有明顯的侷限性，無法推廣至任意情形

解決方案：

使用陣列，考慮以迴圈求出最佳解

Test 2

定義

```python
times = [int(x) for x in input("秒數: ").split()]
rounds = int(input("圈數:"))
cp = float(input("圈疲:"))
tp = float(input("趟疲:"))
```

```python
def min_time(n, t):
    row = len(times)
    col = rounds - 2
    arr = [[float('inf')] * col for _ in range(row)]
    for i in range(row):
        for j in range(col):
            arr[i][j] = times[i] * (cp ** (j))
    total_time = []
```

1. 建立二維串列

```python
    for i in range(row):
        total_time.append(arr[i][0])
        arr[i][0] *= tp
    max_search_col = 1
```

2. 計算選手第 i 趟所需時間

運算

```python
    k = [1] * row
    while len(total_time) < rounds:
        min_time, min_time_row, min_time_col = float('inf'), 0, 0
        for i in range(row):
            for j in range(max_search_col + 1):
                if arr[i][j] <= min_time:
                    if j == 0 and k[i] - k[-1] >= 1:
                        continue
                    min_time = arr[i][j]
                    min_time_row, min_time_col = i, j
        if min_time_col == 0:
            k[min_time_row] += 1
```

3. 計算完成繞圈賽最短時間

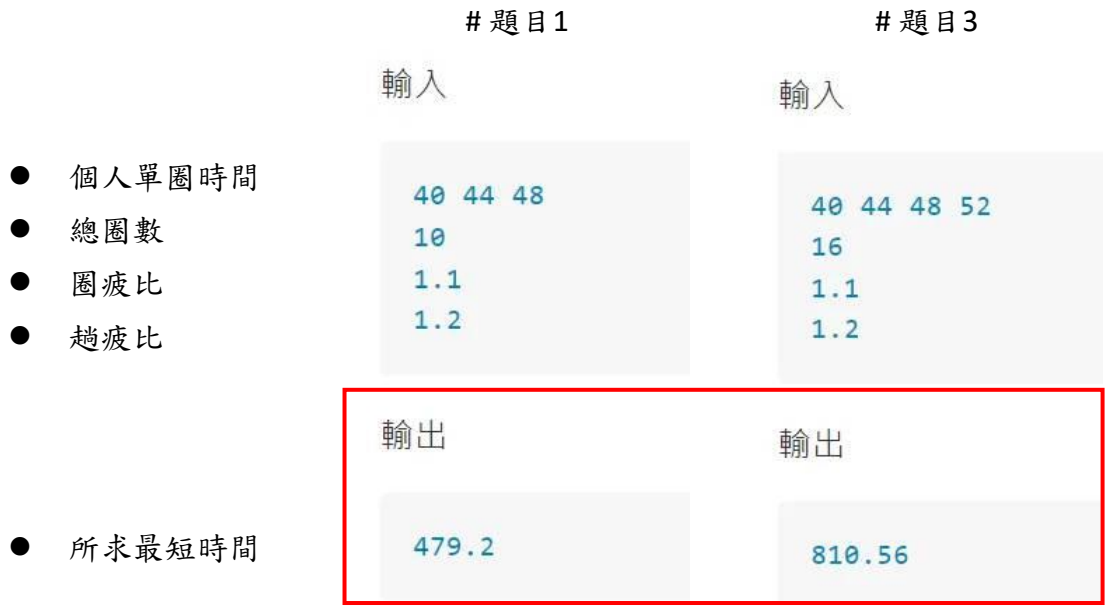
```python
        total_time.append(arr[min_time_row][min_time_col])
        arr[min_time_row][min_time_col] *= tp
        max_search_col = max(max_search_col, min_time_col + 1)

    return sum(total_time)
```

4. 回傳結果

輸出

```python
print(min_time(rounds, times))
```



程式邏輯分析：

Step 1 輸入各項變數

Step 2 透過迴圈運算列出符合的數值加入串列

Step 3 輸出串列內前 k 個最小的數值，k = 總圈數

結語

經過此次繞圈賽題目的研究，依結果而言，我們並非如起初所想像順遂，但也有其他收穫，以程式的建構和算式的列舉為例:組員在建構Python由於無法實行dp且迫於時間壓力，最後改以迴圈+串列的方式進行，並以EXCEL為備案， 不過也因此多了個可討論的專題，並且理解一個數學情境並非必定得用特定的工具和方式。在scratch方面，則面臨比較變數違反規則的問題，不過亦能藉此 探討泡沫演算法的可行性，在列式的方面，面對接力繞圈賽這類數據資料龐大 的最佳解問題，以我們這組的經驗而言，建立數學模型的目的並非直指如何解 決，而是提供可能的方向。雖然列式過程繁瑣，但可以讓思考的脈絡更有層 次；並且先過濾掉顯然不恰當的情形，提高後續討論的效率。