2024.04.14
WEEK2
REVIEW

MAJOR

1. PYTHON, JS基本語法熟悉。
2. FOR, IF 的交叉運用。

MINOR

1. 條件表達式(CONDITIONAL EXPRESSION)
2. 匿名函式(ANONYMOUS FUNCTION)

PRESENT BY YANBO彥伯

WEEK2

- 【第二週任務公告】如火如荼進入第二週！這週要請各位使用 JavaScript 和 Python 基礎語法完成一些邏輯演算。 詳情請參考以下說明及附件檔案：
- 任務目標：運用程式語言解決問題。
- 成果形式：請繼續使用第一週建立的 GitHub Repository，在其中另外建立一個名為 week2 的子資料夾，將所有本週任務相關的程式檔案都放進 week2 子資料夾中。
- 繳交期限：**4/14 週日，23 點 59 分（UTC+8）**。
- 繳交方式：請直接私訊彭彭老師，並在訊息中提供姓名、GitHub Repository 的 week2 資料夾原始碼所在的網址。 注意事項：若對題意有任何疑問，請務必儘早主動釐清確認。 本週任務中有標示 Optional 的題目，大家可以自由選擇是否要完成，並非一定要繳交的任務項目，請根據自己的情況決定是否完成 ~

Task 1:

We just received messages from 5 friends in JSON format, and we want to take the green line, including Xiaobitan station, of Taipei MRT to meet one of them. Write code to find out the nearest friend and print name, based on any given station currently located at and station count between two stations.

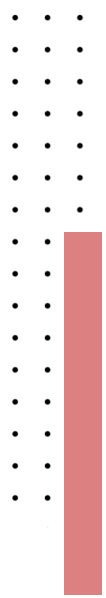
Note: Never change existing code.

Python

```
def find_and_print(messages, current_station):
        # your code here
messages={
        "Leslie":"I'm at home near Xiaobitan station.",
        "Bob":"I'm at Ximen MRT station.",
        "Mary":"I have a drink near Jingmei MRT station.",
        "Copper":"I just saw a concert at Taipei Arena.",
        "Vivian":"I'm at Xindian station waiting for you."
}
find_and_print(messages, "Wanlong") # print Mary
find_and_print(messages, "Songshan") # print Copper
find_and_print(messages, "Qizhang") # print Leslie
find_and_print(messages, "Ximen") # print Bob
find_and_print(messages, "Xindian City Hall") # print Vivian
```

解題概要: partly hinted from Teacher

1. 先撇開圖論
2. 主線、支線分開處理
3. 簡單思考：要找尋最短距離的朋友，
   所以怎麼計算差值？用 list 的 index，
   或者用 dictionary {key: value}
   - 似乎index 比較簡單→list
   - 但我需要有另外一個dictionary
     來記錄 {friend: distance}



解題概要: partly hinted from Teacher

1. 先撇開圖論
2. 主線、支線分開處理
3. 簡單思考：要找尋最短距離的朋友，
   所以怎麼計算差值？用 list 的 index，
   或者用 dictionary {key: value}
   - 似乎index 比較簡單→list
   - 但我需要有另外一個dictionary
     來記錄 {friend: distance}
4. 所以我可能需要創建至少：
   1. List (distance)
   2. Dictionary (friend_distance)
   3. 主線、支線的 List

流程

Def **find_and_print**(messages, current_station):):

Dictionary={} ? / List =[ ]?

1. 先檢驗名字跟訊息裡的站名、有符合的，就比對index

```
for name, msg in messages.items():
```

Dictionary={} ? / List =[ ]?

```
for sublist in [list2, list3] if current_station == "Xiaobitan" else [list1, list2, list3]:
  #按目前所在點是不是 xiaobitan檢視不同list, (不同路線) A:2,3, B:1,2,3
```

```
if current_station in sublist: #如果找到同樣在同樣的list (同樣路線)，按照list[index]來計算距離
    distances = [abs(sublist.index(item) - sublist.index(current_station)) for item in sublist if item in msg]
```

```
if current_station in sublist: #如果找到同樣在同樣的list (同樣路線)，按照list[index]來計算距離
    for item in sublist:
        if item in sublist and item in msg: # 判断 item 是否同时存在于 sublist 和 msg 中
            distances = [abs(sublist.index(item) - sublist.index(current_station)) ]
```

流程

2. 紀錄 the distance for each friend (用創立的那個 Dictionary)
   - 把一些東西加到空字典的方法：Dictionary[key] = volume

3. 取上面最小值(Value) 輸出 朋友的名字 (Key)

```
shortest_distance = min( Dictionary.values() )
Closest_friend=[]
For key, value in Dictionary.items():
        if value ==  _ _ _ _ _ _ _ :
                _ _ _ _ _ _ _ .append(key)
```

濃縮：列表生成式