2048 報告

ICT SBA 專案

黃世隆
6B (6220)

顯理中學
作者: 黃世隆

目錄

1. 簡介

2048 是一款在網頁、電腦和手機上都很受歡迎的單人單機遊戲。 它看起來很簡單，但實際上有一定的挑戰難度，所以有很多人為這個遊戲著迷。

在這個專案中，我將使用 Python3 來創建一個名為“2048”的遊戲。 在我們開始編寫“2048”程式之前，我們應該了解什麼是 2048 以及如何玩。

在本章中，將說說遊戲 2048 的簡介。

1.1 什麼是 2048?

2048 是一款由義大利網路開發者 Gabriele Cirulli 創作的單人滑塊拼圖視頻遊戲，並在 GitHub 上發佈。 遊戲的目標是在一個網格上滑動帶有數位的方塊，將它們組合在一起，創建一個數位為 2048 的方塊; 然而，在達到目標后，玩家可以繼續玩遊戲，創建更大數位的方塊。 它最初是在一個週末用 JavaScript 和 CSS 編寫的，並於 2014 年 3 月 9 日作為免費開源軟體發佈，受到 MIT 許可證的約束。 iOS 和 Android 版本於 2014 年 5 月推出。

2048 的初衷是改進另外兩款遊戲，這兩款遊戲都是在一個月前發佈的 iOS 遊戲 Threes 的克隆版。 Cirulli 本人將 2048 描述為“在概念上類似”於 Threes。 2048 的發佈導致了許多類似遊戲的迅速出現，就像 2013 年 Flappy Bird 變種遊戲的泛濫一樣。 遊戲在評論家中獲得了普遍積極的評價，被描述為「病毒式」的和“令人上癮的”。

1.2 2048 玩法

2048 是在一個簡單的 4×4 網格上進行的，玩家通過使用四個箭頭鍵移動數位方塊來進行遊戲。 每回合，一個新的方塊以 2 或 4 的值隨機出現在空位上。 方塊在選擇的方向上盡可能遠地滑動，直到被另一個方塊或網格的邊緣阻止。 如果移動過程中兩個相同數位的方塊相撞，它們將合併成一個總值為兩個方塊之和的方塊。 合併後的方塊在同一次移動中無法再次合併。 得分較高的方塊會發出柔和的光芒;最高可能的方塊是 131,072。

如果一次移動導致三個連續相同數值的方塊在一起滑動，只有沿運動方向最遠的兩個方塊會合併。 如果一行或一列的四個空間都被相同數值的方塊填滿，與該行/列平行的移動將合併前兩個和后兩個方塊。 右上角的記分牌會記錄玩家的得分。 玩家的初始得分為零，每當兩個方塊合併時，得分會增加新方塊的值。

當一個值為 2048 的方塊出現在網格上時，遊戲勝利。 玩家可以繼續玩遊戲以達到更高的得分。 當玩家沒有合法的移動時（沒有空位和相鄰的方塊具有相同的值），遊戲結束。

2. 設計 2048 遊戲

在設計 2048 遊戲之初，我將先找出和解決一些問題並解決，然後在終端設計一個簡單的 2048 遊戲，最後進行測試和改進：

2.1 釐清和分析問題

在設計 2048 前，一大堆問題和選擇顯在眼前，例如：

- 語言選擇
- 應用選擇
- 包含功能
- 視窗設計
- …

我會就著這些問題進行解決

2.1.1 編程語言的選擇

市面上有非常多種編程語言的選擇，當中截止 2023 年最受歡迎的三種選擇是 Python3， C/C++和 Java。而我選擇使用 Python3 作為我的編程語言。

為什麼使用 Python3 而不是 C/C++或 Java 呢？

Python 是一種高階程式語言，以其簡潔和可讀性而聞名。它是一種直譯語言，這意味著用 Python 編寫的程式碼可以在不需要編譯的情況下執行，這使得開發和測試程序變得更加快速和簡便。此外，Python 擁有龐大且活躍的社群，提供了各種可以用於擴展其功能的庫和工具。

對於 2048 遊戲來說，Python 是一個不錯的選擇，因為該遊戲並不需要很高的性能水平，主要關注的是遊戲邏輯而不是圖形。Python 提供了簡單易懂的語法，使得編寫和維護代碼變得更加容易。此外，Python 擁有廣泛的庫，可用於處理文件的輸入/輸出，數據序列化以及其他遊戲所需的任務。

儘管 C/C++和 Java 也是流行的程式語言，但它們更適合需要高性能的應用，比如具有複雜圖形或模擬的遊戲。然而，對於像 2048 這樣的簡單遊戲來說，Python 在易用性和性能之間提供了一個良好的平衡。

2.1.2 在 GUI 或終端上與 2048 遊戲進行用戶交互？

GUI（圖形用戶界面）和終端（或稱命令行界面）是兩種用於與電腦系統進行交互的不同方式。它們在以下幾個方面有所不同：

1. 介面形式：GUI 使用視覺元素（如窗口、按鈕、菜單等）和圖形化的方式來呈現信息和接收用戶輸入。終端則主要通過文字命令和指令行界面來執行操作，用戶需要輸入特定的命令進行操作。

2. 互動方式：在 GUI 中，用戶通常可以通過點擊、拖放、拉動等方式直觀地與系統進行互動。而在終端中，用戶需要通過鍵盤輸入命令和參數來執行操作。

3. 可視化能力：由於 GUI 使用圖形化的方式呈現信息，它能夠提供更豐富和直觀的可視化效果，如圖片、圖表、視頻等。終端主要通過文字形式呈現信息，限制了可視化能力。

4. 學習曲線：GUI 通常被認為對於新用戶更容易上手，因為它提供了直觀的界面和操作方式。終端則通常需要較多的學習和熟悉，因為用戶需要熟悉特定的命令和語法。

5. 功能和靈活性：終端通常提供更大的功能和靈活性，用戶可以通過組合不同的命令和參數來實現更高級的操作和自動化。GUI 則更適合於日常的基本任務和常見操作，並且通常提供了直觀的操作方式。

總體而言，GUI 和終端在介面形式、互動方式、可視化能力、學習曲線以及功能和靈活性等方面存在差異。選擇使用哪種方式取決於任務的性質、用戶的需求和個人偏好。許多系統提供了 GUI 和終端兩種方式，以便用戶根據具體情況進行選擇和使用。

通過綜合設計成本、難易度、錯誤的出現機率、編碼時間等因素，最後選擇了以終端（或稱命令行界面）作為以 2048 遊戲的交互方法。

補充：
大部分由 AI 輸出的 2048 遊戲範本都是以 GUI 為底，設計終端交互的 2048 不但可以專注於功能和算法上的提升，更可以在不依賴於 AI 的情況下設計 2048。

2.1.3 包含的功能

在 2048 遊戲內，除了遊戲本身，還需要存在其他功能，這些功能與用戶的交互和遊戲特色有著不同程度的幫助。下面會列出 2048 中我會加入的功能：

1. 排名功能：各用戶可以輸入自己的名稱到程序內，然後程式會和分數紀錄在 txt 文檔內，這有助用戶與用戶之間的比較，增加遊戲的吸引力。

2. 遊戲存檔：用戶在遊戲的過程中會因事需要中途退出，無論是因為用戶手動退出，或其他原因導致遊戲崩潰（例如用戶在沒有儲存的情況下關機），都需要遊戲實時存檔功能，實時儲存數字的位置和分數，這樣無論如何都不會導致遊戲進度丟失。

3. 菜單循環：在程式被執行時，要讓用戶先選擇開新遊戲或者恢復上一次遊戲，然後在程式執行相對應的代碼。

4. 遊戲循環：在遊戲畫面當中會有各種動作讓用戶選擇，例如移動、查看排名、儲存遊戲、退出遊戲… 而當遊戲結束後會告知用戶並讓用戶選擇重新開始遊戲或者退出。

2.1.4 視窗設計

早起我們已經選擇以終端（或稱命令行界面）作為以 2048 遊戲的交互方法。然而，我們需要設計遊戲畫面，無論是菜單循環、遊戲循環和遊戲畫面。

對於菜單循環和遊戲循環都很容易，只需列出每個英文字母代表的動作並等待用戶輸入即可。

而對於遊戲畫面，我們需要製造一個 4x4 的網格，並在每個方格內填入空白或數字，執行以下偽代碼:

```
數列轉網格(int 數列[4][4])                    // 輸入一個 4x4 索引的數列
    str 打印[ ] ← [ ]
    int row
    在 數列[4][4] 中分別代入 int i 執行
        在 i 中分別代入 int n 執行
            如果 n ≠ 0 執行
                row ← 在每一個‘ | ‘中插入剛好 5 個字符的‘n’    // 例如‘ |1|2|8| ’
            否則
                row ← 在每一個‘ | ‘中插入‘    ’              // 例如‘ | | | | ’
        打印[ ] ← 插入元素 ‘|’ + row + ‘|’
        打印[ ] ← 插入元素 " ├──────┼──────┼──────┼──────┤ "
    刪除 打印[ ] 中的最後一個元素
    傳回 打印[ ]

輸出遊戲面板(str 列印內容[ ])
    輸出列印 " ┌──────┬──────┬──────┬──────┐ "
    在 列印內容[ ] 中分別代入 int row 中執行
        輸出列印 row
    輸出列印 " └──────┴──────┴──────┴──────┘ "
```

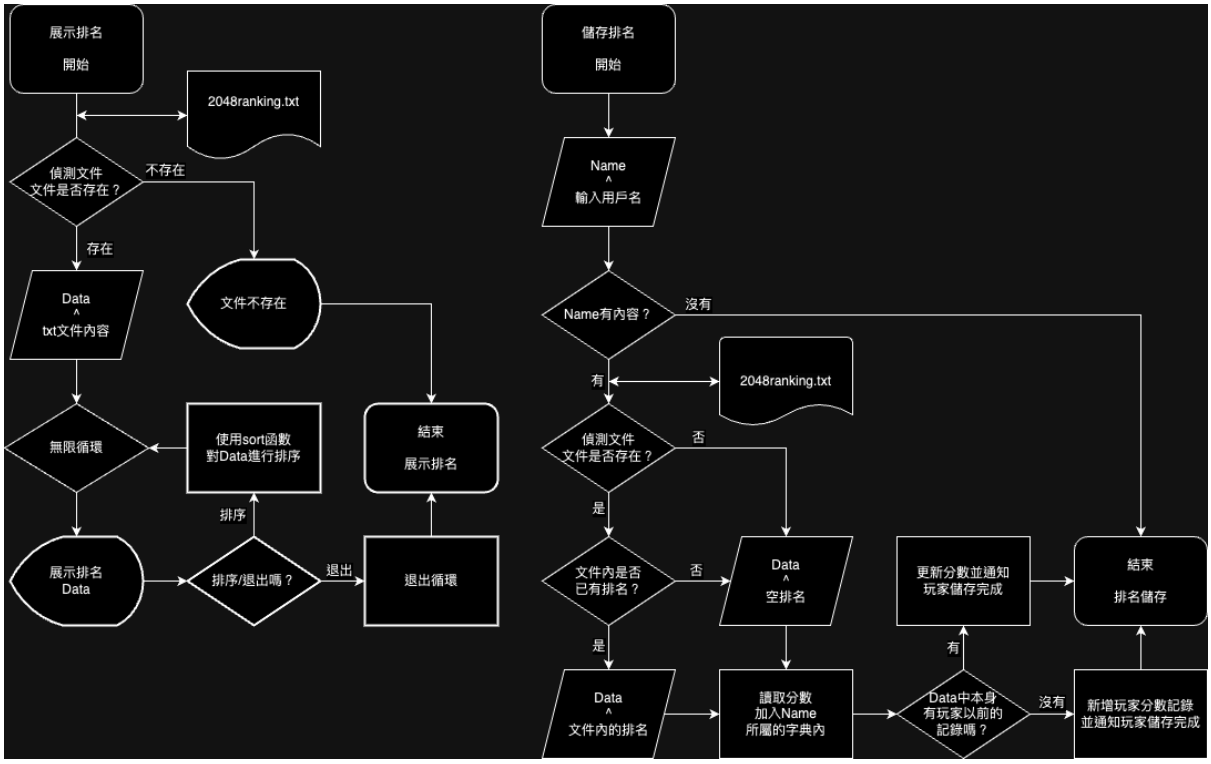
最後打印輸出的樣子大概是:

2.2 各功能的流程圖

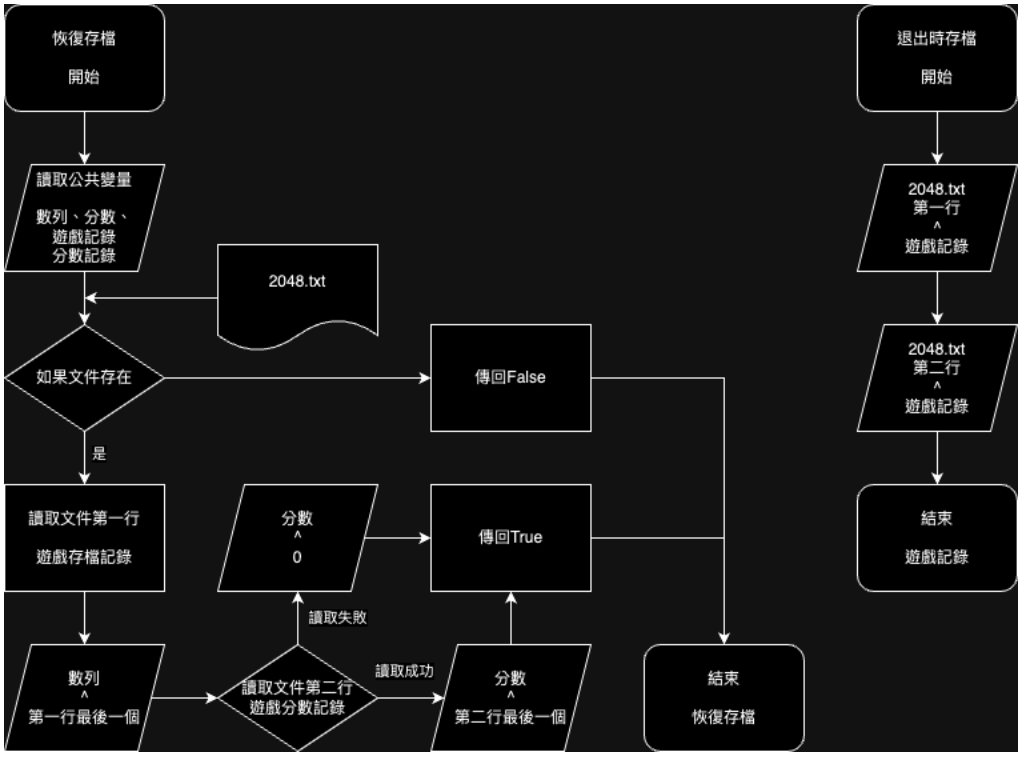
流程圖在理解、改進和管理流程方面具有重要的作用。它們能夠提供視覺化的方式來呈現流程，幫助人們更好地理解、溝通和改進流程，從而提高效率、品質和組織的整體效能。

我會列出 3 個比較重要和功能複雜的流程圖，分別是：

2.2.1 排名功能流程圖


展示排名
開始
2048ranking.txt
偵測文件
文件是否存在？
不存在
存在
Data
^
txt文件內容
文件不存在
無限循環
使用sort函數
對Data進行排序
結束
展示排名
排序
展示排名
Data
排序/退出嗎？
退出
退出循環
儲存排名
開始
Name
^
輸入用戶名
Name有內容？
沒有
有
2048ranking.txt
偵測文件
文件是否存在？
否
是
文件內是否
已有排名？
否
Data
^
空排名
更新分數並通知
玩家儲存完成
結束
排名儲存
是
有
Data
^
文件內的排名
讀取分數
加入Name
所屬的字典內
Data中本身
有玩家以前的
記錄嗎？
沒有
新增玩家分數記錄
並通知玩家儲存完成

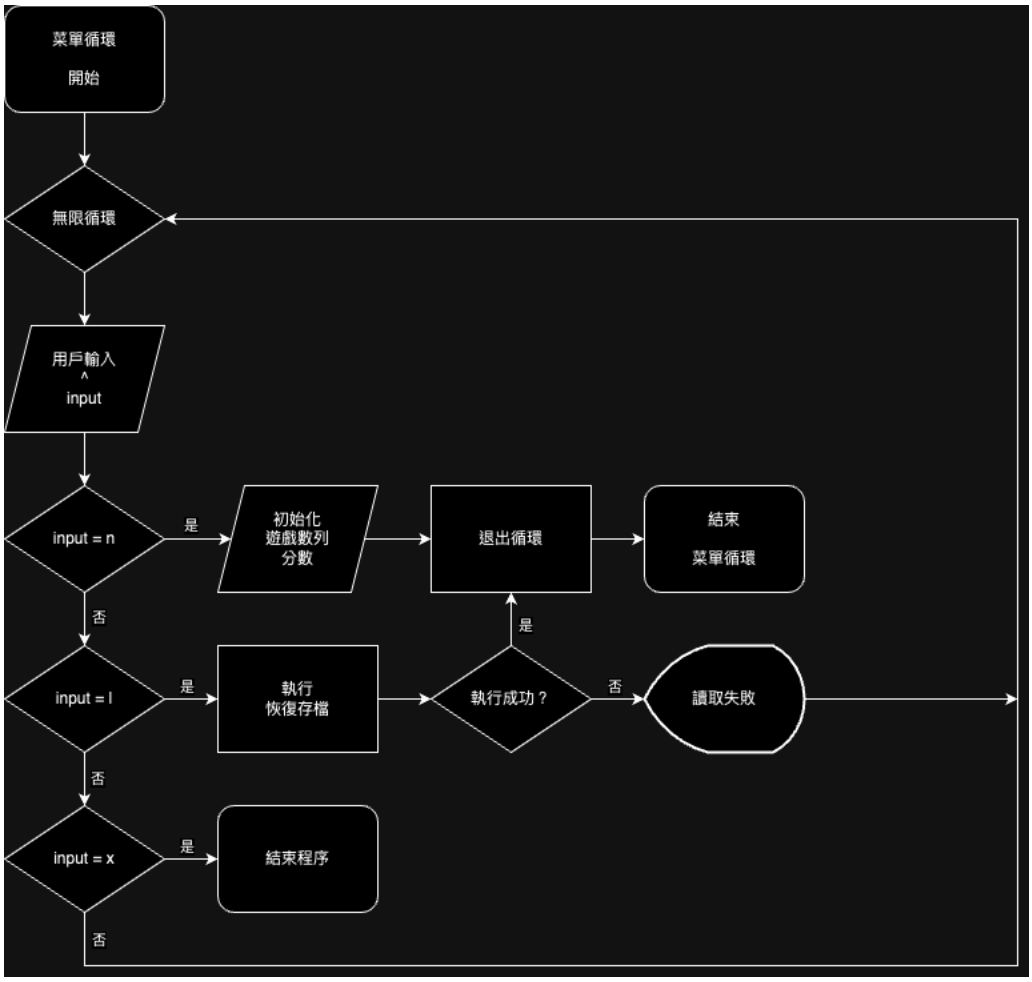
2.2.2 遊戲存檔流程圖


恢復存檔
開始
讀取公共變量
數列、分數、
遊戲記錄
分數記錄
2048.txt
如果文件存在
傳回False
是
讀取文件第一行
遊戲存檔記錄
分數
^
0
傳回True
讀取失敗
數列
^
第一行最後一個
讀取文件第二行
遊戲分數記錄
讀取成功
分數
^
第二行最後一個
結束
恢復存檔
退出時存檔
開始
2048.txt
第一行
^
遊戲記錄
2048.txt
第二行
^
遊戲記錄
結束
遊戲記錄

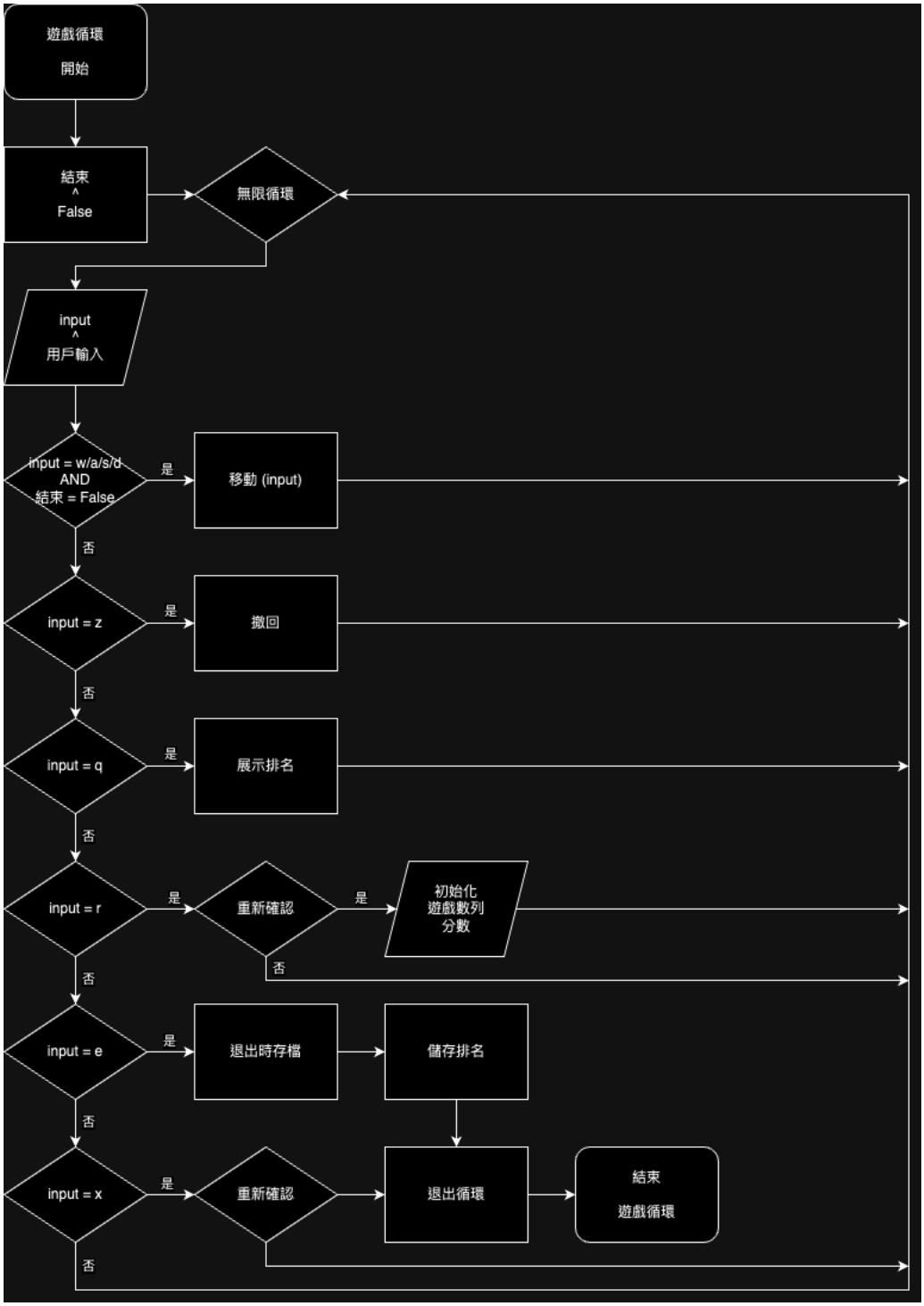
2.2.3 菜單/遊戲循環流程圖

菜單和遊戲循環是兩個不同的循環，當用戶在菜單循環選擇完畢後就會進入遊戲循環。

菜單循環:



遊戲循環


遊戲循環
開始
結束
^
False
無限循環
input
^
用戶輸入
input = w/a/s/d
AND
結束 = False
是
移動 (input)
否
input = z
是
撤回
否
input = q
是
展示排名
否
input = r
是
重新確認
是
初始化
遊戲數列
分數
否
否
input = e
是
退出時存檔
儲存排名
否
input = x
是
重新確認
退出循環
結束
遊戲循環
否

2.3 編寫一個簡單的程式

程式中有不同類型的函數:

這些功能結合成為一個完整的 2048 遊戲，接下來詳細講述每一個功能的代碼、實現方法和運行例子。

2.3.1 4x4 遊戲面板函數

首先，我們編寫一個程式來輸出一個 4x4 的棋盤：

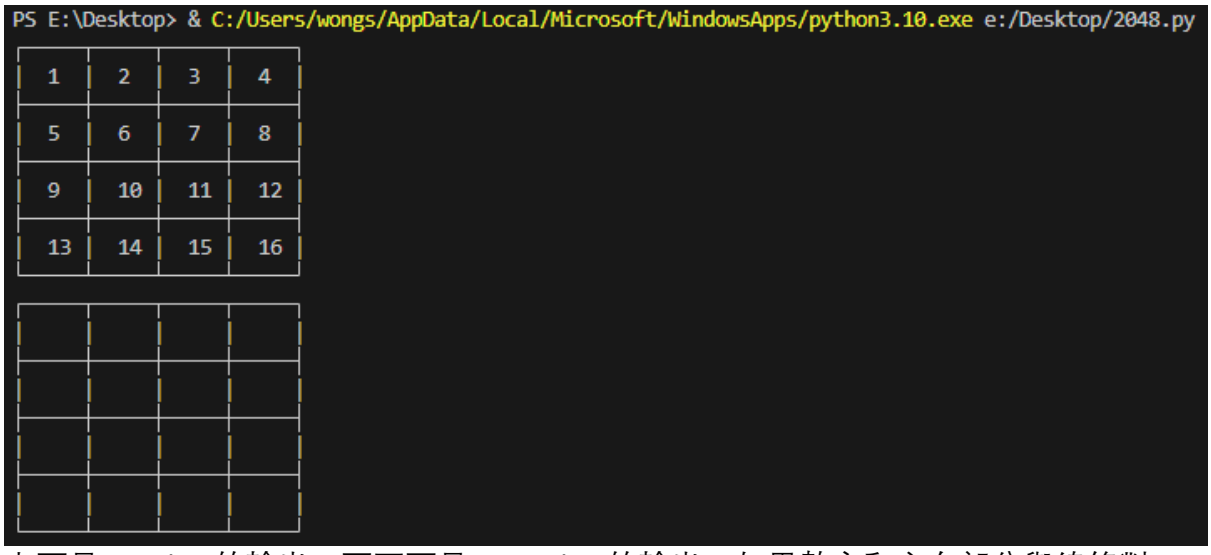
- 這段程式碼定義了三個函式：NumToPrtlist()、prtNum()和 printBoard()。
- NumToPrtlist() 函式接受一個數字列表作為輸入，將每個數字轉換為格式化的字符串，並添加前導空格以確保每個字符串長度為五個字符。然後，將結果字符串附加到一個新的列表中，該列表將由該函式返回。
- prtNum() 函式接受一個字符串列表作為輸入，以格式化的方式將它們打印出來，每個字符串之間以垂直線分隔。
- printBoard() 函式將 NumToPrtlist 的輸出作為輸入，使用 ASCII 字符打印出一個格式化的遊戲棋盤。棋盤由一個格子網絡組成，每個格子包含一個格式化的字符串。格子之間以水平線和垂直線分隔，棋盤的角落是圓角。
- 程式碼接下來創建一個包含 1 到 16 的數字列表，對該列表調用 NumToPrtlist() 以創建一個格式化的字符串列表，最後對該格式化列表調用 printBoard 來打印出一個遊戲棋盤。總體而言，該程式碼是用於打印出一個 4x4 網格的遊戲棋盤。

2048.py

```python
import numpy as np
import random as ran
import sys

numlist = [i for i in range(1, 17)]
emptylist = [""] * 16

def NumToPrtlist(numlist):
    j = 0
    printlist = []
    for i in numlist:
        if len(str(numlist[j])) == 0:
            obj = "    "
            printlist.append(obj)
        if len(str(numlist[j])) == 1:
            obj = f"  {str(numlist[j])} "
            printlist.append(obj)
        if len(str(numlist[j])) == 2:
            obj = f"  {str(numlist[j])} "
            printlist.append(obj)
        if len(str(numlist[j])) == 3:
            obj = f" {str(numlist[j])} "
            printlist.append(obj)
        if len(str(numlist[j])) == 4:
            obj = f" {str(numlist[j])}"
            printlist.append(obj)
        if len(str(numlist[j])) == 5:
            obj = f"{str(numlist[j])}"
            printlist.append(obj)
        j+=1
    return printlist

def prtNum(a):
    row = "|".join(a)
    return f"|{row}|"

def printBoard(prtlist):
    prtSrt = "┌────┬────┬────┬────┐"
    prtMid = "├────┼────┼────┼────┤"
    prtEnd = "└────┴────┴────┴────┘"
    print(prtSrt)
    print(prtNum(prtlist[0:4]))
    print(prtMid)
    print(prtNum(prtlist[4:8]))
    print(prtMid)
    print(prtNum(prtlist[8:12]))
    print(prtMid)
    print(prtNum(prtlist[12:]))
    print(prtEnd)

printBoard(NumToPrtlist(numlist))
printBoard(NumToPrtlist(emptylist))
```

這段程式碼將會按照 4x4 的網格排列，打印出數字 1 到 16 的遊戲棋盤，每個數字都被包裹在一個格子中。這些格子之間以水平和垂直線分隔，而棋盤的角落則是圓角的。

以下是終端的輸出：



上面是 numlist 的輸出，而下面是 emptylist 的輸出，如果數字和空白部分與線條對齊，代表成功!

2.3.2 startNum() 函數

然後，我們製作一個 2048 的開始頁面：
這段程式碼用於打印出一個 4x4 網格的 2048 遊戲棋盤，其初始值由 startNum()函式隨機分配。

2048.py

```python
import numpy as np
import random as ran
import sys

def startNum():
    numlist = [0] * 16
    randlist = ran.sample(range(1, 16), 2)
    numlist[randlist[0]] = 2
    numlist[randlist[1]] = ran.sample((2, 4), 1)[0]
    return numlist

def NumToPrtlist(numlist):
    j = 0
    printlist = []
    for i in numlist:
        if numlist[j] == 0:
            obj = "      "
            printlist.append(obj)
        if len(str(numlist[j])) == 1 and not numlist[j] == 0:
            obj = f"  {str(numlist[j])}  "
            printlist.append(obj)
        if len(str(numlist[j])) == 2:
            obj = f"  {str(numlist[j])}  "
            printlist.append(obj)
        if len(str(numlist[j])) == 3:
            obj = f" {str(numlist[j])}  "
            printlist.append(obj)
        if len(str(numlist[j])) == 4:
            obj = f" {str(numlist[j])}"
            printlist.append(obj)
        if len(str(numlist[j])) == 5:
            obj = f"{str(numlist[j])}"
            printlist.append(obj)
        j+=1
    return printlist

def prtNum(a):
    row = "|".join(a)
    return f"|{row}|"

def printBoard(prtlist):
    prtSrt = "┌──────┬──────┬──────┬──────┐"
    prtMid = "├──────┼──────┼──────┼──────┤"
    prtEnd = "└──────┴──────┴──────┴──────┘"
    print(prtSrt)
    print(prtNum(prtlist[0:4]))
    print(prtMid)
    print(prtNum(prtlist[4:8]))
    print(prtMid)
    print(prtNum(prtlist[8:12]))
    print(prtMid)
    print(prtNum(prtlist[12:]))
    print(prtEnd)

printBoard(NumToPrtlist(numlist = startNum()))
```

這段程式碼將會打印出一個 2048 遊戲的遊戲棋盤，其中有兩個隨機分配的初始值，一個是 2，另一個是 2 或 4 之間的數字，且不重複。

遊戲棋盤將以 4x4 的格子網格顯示，每個格子包含一個數字或空白區域，數字格式化為帶有前導空格的字符串，以確保每個字符串長度為五個字符。格子之間以水平和垂直線分隔，而棋盤的角落則是圓角的。

以下是終端輸出：

（圖片例子內的數字與實際輸出可能不相同）
如果見到兩個數字在面板內代表成功了

2.3.3 升級至 2D 矩陣

接下來將 startNum() 和 NumToPrintlist() 函數升級至 2D 矩陣

```python
def startNum():
    numlist = [[0 for _ in range(4)] for _ in range(4)]
    randlist = ran.sample(range(16), 2)
    numlist[randlist[0]//4][randlist[0]%4] = 2
    numlist[randlist[1]//4][randlist[1]%4] = ran.choice([2, 4])
    print(numlist)
    return numlist
def NumToPrtlist(numlist):
    j = 0
    printlist = []
    for i in numlist:
        l = 0
        for k in i:
            if numlist[j][l] == 0:
                obj = "      "
                printlist.append(obj)
            if len(str(numlist[j][l])) == 1 and not numlist[j][l] == 0:
                obj = f"  {str(numlist[j][l])}  "
                printlist.append(obj)
            if len(str(numlist[j][l])) == 2:
                obj = f"  {str(numlist[j][l])} "
                printlist.append(obj)
            if len(str(numlist[j][l])) == 3:
                obj = f" {str(numlist[j][l])} "
                printlist.append(obj)
            if len(str(numlist[j][l])) == 4:
                obj = f" {str(numlist[j][l])}"
                printlist.append(obj)
            if len(str(numlist[j][l])) == 5:
                obj = f"{str(numlist[j][l])}"
                printlist.append(obj)
            l+=1
        j+=1
    return printlist
```

升級至 2D 矩陣可以更方便處理方格內的數字，特別是之後的移動和數字合成。

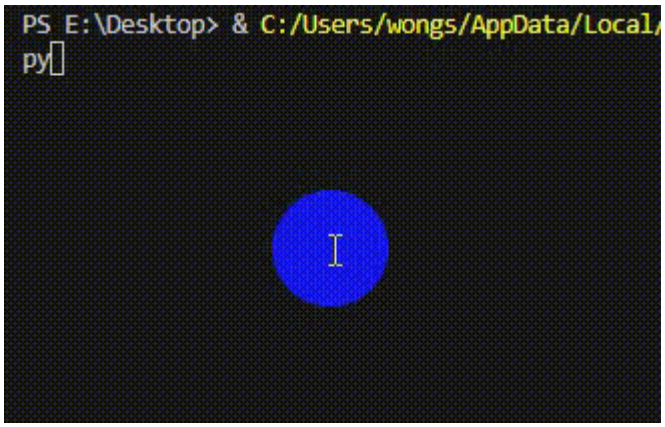
2.3.4 遊戲循環

接下來，我們添加一個遊戲循環，讓用戶輸入。
while 循環是主要的遊戲循環，它會重複清除控制台，打印當前的遊戲板，並等待玩家輸入。玩家可以通過按下'w'、'a'、's'或'd'鍵來將數字在遊戲板上移動，分別將數字向上、向左、向下或向右移動。按下'r'鍵可以重新開始遊戲，按下'e'鍵可以退出遊戲。如果玩家輸入無效的按鍵，循環將繼續而不進行任何操作。

```
while True:
    os.system('cls' if os.name == 'nt' else 'clear')
    printBoard(NumToPrtlist(numlist))
    print("input WASD to move, R to retry, E to exit.")
    act = input("What next? :")
    x = 0
    y = 0
    if act == "w":
        for _ in range(4):
            for x in range(4):
                for y in range(3):
                    if numlist[y+1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y+1][x]
                        numlist[y+1][x] = 0
    elif act == "a":
        for _ in range(4):
            for y in range(4):
                for x in range(3):
                    if numlist[y][x+1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x+1]
                        numlist[y][x+1] = 0
    elif act == "s":
        for _ in range(4):
            for x in range(4):
                for y in (3, 2, 1):
                    if numlist[y-1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y-1][x]
                        numlist[y-1][x] = 0
    elif act == "d":
        for _ in range(4):
            for y in range(4):
                for x in (3, 2, 1):
                    if numlist[y][x-1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x-1]
                        numlist[y][x-1] = 0
    elif act == "r":
        numlist = startNum()
        continue
    elif act == "e":
        break
    else:
        continue
```

遊戲開始時，startNum() 函數會在遊戲板上隨機選擇兩個位置，並在每個位置放置一個 2 或 4。然後，使用 printBoard() 函數將生成的遊戲板打印到控制台。

之後，遊戲循環會重複打印當前的遊戲板並等待玩家輸入。根據玩家的輸入，遊戲板將被更新並再次打印。如果玩家輸入'r'鍵，遊戲將重新開始並使用 startNum 函數生成新的遊戲板。如果玩家輸入'e'鍵，遊戲將退出並終止程序。



為了令程式更清晰，我將移動的代碼部分造成函數 moving()：

```
59  def moving(act, numlist):
60      if act == "w":
61          for _ in range(4):
62              for x in range(4):
63                  for y in range(3):
64                      if numlist[y+1][x] !=0 and numlist[y][x] == 0:
65                          numlist[y][x] = numlist[y+1][x]
66                          numlist[y+1][x] = 0
67      elif act == "a":
68          for _ in range(4):
69              for y in range(4):
70                  for x in range(3):
71                      if numlist[y][x+1] !=0 and numlist[y][x] == 0:
72                          numlist[y][x] = numlist[y][x+1]
73                          numlist[y][x+1] = 0
74      elif act == "s":
75          for _ in range(4):
76              for x in range(4):
77                  for y in (3, 2, 1):
78                      if numlist[y-1][x] !=0 and numlist[y][x] == 0:
79                          numlist[y][x] = numlist[y-1][x]
80                          numlist[y-1][x] = 0
81      elif act == "d":
82          for _ in range(4):
83              for y in range(4):
84                  for x in (3, 2, 1):
85                      if numlist[y][x-1] !=0 and numlist[y][x] == 0:
86                          numlist[y][x] = numlist[y][x-1]
87                          numlist[y][x-1] = 0
88      return numlist
```

```
92  while True:
93      os.system('cls' if os.name == 'nt' else 'clear')
94      printBoard(NumToPrtlist(numlist))
95      print("input WASD to move, R to retry, E to exit.")
96      act = input("What next? :")
97      x = 0
98      y = 0
99      if act in ("w", "a", "s", "d"):
100         numlist = moving(act, numlist)
101     elif act == "r":
102         numlist = startNum()
103         continue
104     elif act == "e":
105         break
106     else:
107         continue
108
```

無論如何，保持代碼清晰可以提高程式碼的透明度，並且降低維護成本

2.3.5 check()函數

接下來，我們添加一個 addNum() 函數來在移動後增加一個數字，並添加一個 check() 函數來檢查遊戲板是否已滿：

```python
def addNum(numlist):
    while True:
        randNum = ran.randint(0, 15)
        if numlist[randNum//4][randNum%4] == 0:
            numlist[randNum//4][randNum%4] = ran.choice([2, 4])
            break
    return numlist

def check(numlist):
    over = 1
    for i in numlist:
        for j in i:
            if j == 0:
                over = 0
                break
    return over
```

```python
while True:
    os.system('cls' if os.name == 'nt' else 'clear')
    printBoard(NumToPrtlist(numlist))
    print("input WASD to move, R to retry, E to exit." if not over else "GAMEOVER! R to retry, E to exit.")
    act = input("What next? :")
    x = 0
    y = 0
    if act in ("w", "a", "s", "d") and not over:
        numlist = moving(act, numlist)
        numlist = addNum(numlist)


    elif act == "r":
        numlist = startNum()
        continue
    elif act == "e":
        break
    else:
        continue
    over = check(numlist)
```

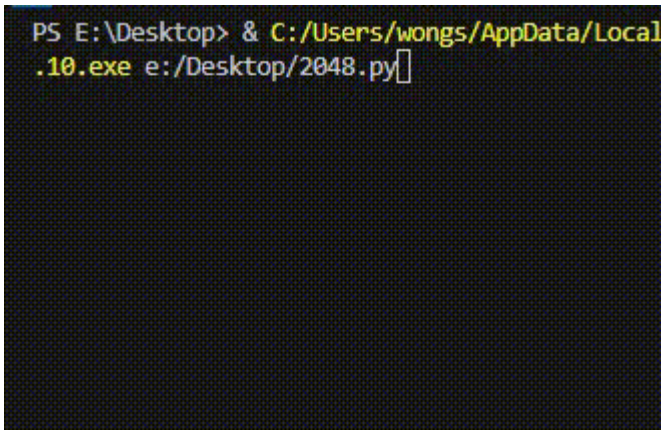
如果遊戲面板已滿，這會顯示 GAME OVER!

```
┌─────┬─────┬─────┬─────┐
│  2  │  2  │  4  │  4  │
├─────┼─────┼─────┼─────┤
│  2  │  2  │  4  │  4  │
├─────┼─────┼─────┼─────┤
│  2  │  4  │  2  │  2  │
├─────┼─────┼─────┼─────┤
│  4  │  2  │  4  │  2  │
└─────┴─────┴─────┴─────┘
GAMEOVER! R to retry, E to exit.
What next? :
```

接下來，我新增一個偵測代碼，檢查相鄰的兩個數字能否合成:

```python
def moving(act, numlist):
    if act == "w":
        for _ in range(4):
            for x in range(4):
                for y in range(3):
                    if numlist[y+1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y+1][x]
                        numlist[y+1][x] = 0
                    elif numlist[y+1][x] !=0 and numlist[y][x] != 0 and numlist[y+1][x] == numlist[y][x]:
                        numlist[y][x] = numlist[y+1][x] + numlist[y][x]
                        numlist[y+1][x] = 0
    elif act == "a":
        for _ in range(4):
            for y in range(4):
                for x in range(3):
                    if numlist[y][x+1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x+1]
                        numlist[y][x+1] = 0
                    elif numlist[y][x+1] !=0 and numlist[y][x] != 0 and numlist[y][x+1] == numlist[y][x]:
                        numlist[y][x] = numlist[y][x+1] + numlist[y][x]
                        numlist[y][x+1] = 0
    elif act == "s":
        for _ in range(4):
            for x in range(4):
                for y in (3, 2, 1):
                    if numlist[y-1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y-1][x]
                        numlist[y-1][x] = 0
                    elif numlist[y-1][x] !=0 and numlist[y][x] != 0 and numlist[y-1][x] == numlist[y][x]:
                        numlist[y][x] = numlist[y-1][x] + numlist[y][x]
                        numlist[y-1][x] = 0
    elif act == "d":
        for _ in range(4):
            for y in range(4):
                for x in (3, 2, 1):
                    if numlist[y][x-1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x-1]
                        numlist[y][x-1] = 0
                    elif numlist[y][x-1] !=0 and numlist[y][x] != 0 and numlist[y][x-1] == numlist[y][x]:
                        numlist[y][x] = numlist[y][x-1] + numlist[y][x]
                        numlist[y][x-1] = 0
    return numlist
```

現在兩個相鄰相同的數字進行移動（moving()函數）後可以合成:



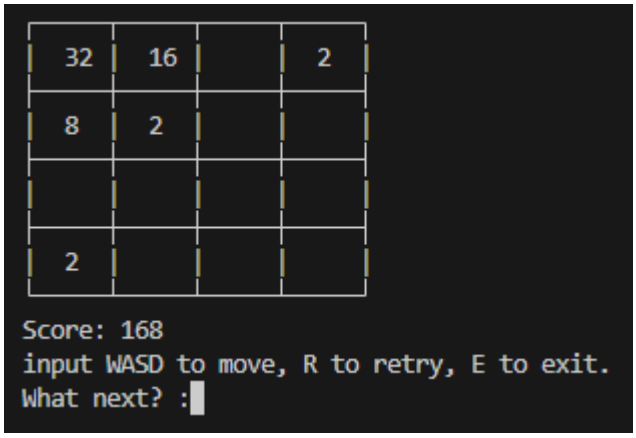
2.3.6 計分板功能

接下來，在程序內新增一個計分板功能：

```
def moving(act, numlist):
    global score
    if act == "w":
        for _ in range(4):
            for x in range(4):
                for y in range(3):
                    if numlist[y+1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y+1][x]
                        numlist[y+1][x] = 0
                    elif numlist[y+1][x] !=0 and numlist[y][x] != 0 and numlist[y+1][x] == numlist[y][x]:
                        numlist[y][x] = numlist[y+1][x] + numlist[y][x]
                        numlist[y+1][x] = 0
                        score = score + numlist[y][x]

    elif act == "a":
        for _ in range(4):
            for y in range(4):
                for x in range(3):
                    if numlist[y][x+1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x+1]
                        numlist[y][x+1] = 0
                    elif numlist[y][x+1] !=0 and numlist[y][x] != 0 and numlist[y][x+1] == numlist[y][x]:
                        numlist[y][x] = numlist[y][x+1] + numlist[y][x]
                        numlist[y][x+1] = 0
                        score = score + numlist[y][x]
    elif act == "s":
        for _ in range(4):
            for x in range(4):
                for y in (3, 2, 1):
                    if numlist[y-1][x] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y-1][x]
                        numlist[y-1][x] = 0
                    elif numlist[y-1][x] !=0 and numlist[y][x] != 0 and numlist[y-1][x] == numlist[y][x]:
                        numlist[y][x] = numlist[y-1][x] + numlist[y][x]
                        numlist[y-1][x] = 0
                        score = score + numlist[y][x]
    elif act == "d":
        for _ in range(4):
            for y in range(4):
                for x in (3, 2, 1):
                    if numlist[y][x-1] !=0 and numlist[y][x] == 0:
                        numlist[y][x] = numlist[y][x-1]
                        numlist[y][x-1] = 0
                    elif numlist[y][x-1] !=0 and numlist[y][x] != 0 and numlist[y][x-1] == numlist[y][x]:
                        numlist[y][x] = numlist[y][x-1] + numlist[y][x]
                        numlist[y][x-1] = 0
                        score = score + numlist[y][x]
    return numlist
score = 0

while True:
    os.system('cls' if os.name == 'nt' else 'clear')
    printBoard(NumToPrtlist(numlist))
    print(f"Score: {score}")
    print("input WASD to move, R to retry, E to exit." if not over else "GAMEOVER! R to retry, E to exit.")
    act = input("What next? :")
    x = 0
    y = 0
    if act in ("w", "a", "s", "d") and not over:
        numlist = moving(act, numlist)
        numlist = addNum(numlist)


    elif act == "r":
        numlist = startNum()
        continue
    elif act == "e":
        break
    else:
        continue
    over = check(numlist)
```

在 moving()函數中，如果有數字合成，會加分，而在遊戲循環中加入了分數顯示，並在每一次循環更新分數。

現在，遊戲開始時會計分：



計分方法：
當兩個數字合成後，新的數字會出現，該數字會加入分數內

例子：
原始計分：0
‘2’ 和 ‘2’ 合成後會變成 ‘4’，記 4 分
現在計分：4

原始計分：5000
‘128‘ 和 ’128‘ 合成後會變成 ’256‘，記 256 分
現在計分：5256

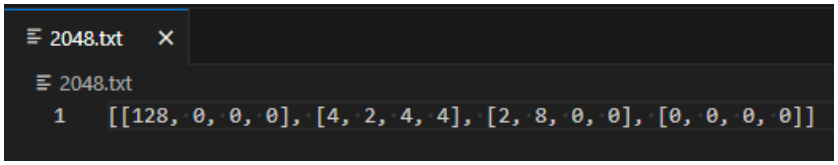
2.3.7 儲存功能

接下來，我們新增一個儲存功能，讓玩家可以在開始遊戲時選擇新遊戲或者上一次遊戲

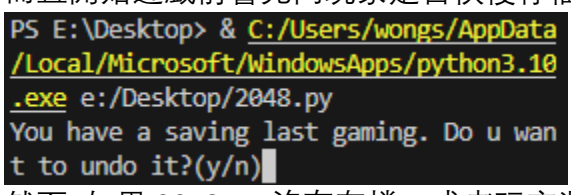
```
f = open('2048.txt', 'r')
s = f.read()
if s:
    if input("You have a saving last gaming. Do u want to undo it?(y/n)") in ("Y", "y"):
        numlist = ast.literal_eval(s)
    else:
        numlist = startNum()
else:
    numlist = startNum()

while True:
    os.system('cls' if os.name == 'nt' else 'clear')
    printBoard(NumToPrtlist(numlist))
    print(f"Score: {score}")
    print("input WASD to move, R to retry, E to exit." if not over else "GAMEOVER! R to retry, E to exit.")
    act = input("What next? :")
    x = 0
    y = 0
    if act in ("w", "a", "s", "d") and not over:
        numlist = moving(act, numlist)
        if moved:
            numlist = addNum(numlist)
            moved = 0


    elif act == "r":
        numlist = startNum()
        continue
    elif act == "e":
        f = open('2048.txt', 'w')
        f.write(str(numlist))
        f.close()
        f = open()
        break
    else:
        continue
    over = check(numlist)
```

程式碼中的 ast.literal_eval(s)引用了 ast 庫，可以直接讀寫 txt 文件

結束遊戲後，遊戲紀錄會以 2D 矩陣列表的形式儲存在 2048.txt



而且開始遊戲前會先問玩家是否恢復存檔:



然而, 如果 2048.txt 沒有存檔，或者玩家選擇輸入 n，會開始新遊戲

2.3.8 排名功能

接下來，我們在遊戲內新增排名功能：

```
121     try:
122         score = f.readlines()[1]
123     except IndexError:
124         score = 0
```

讀取 2048.txt 第二行的分數，如果沒有分數紀錄，分數設為 0

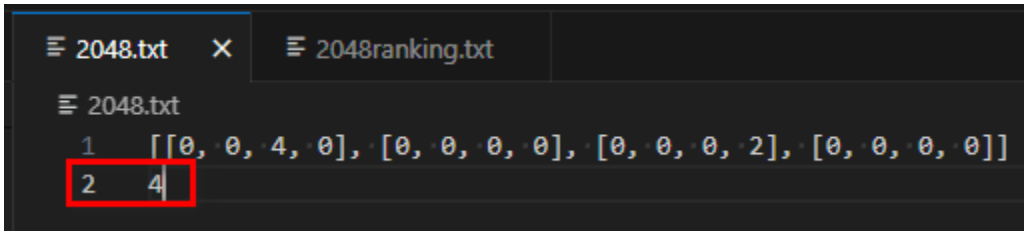
```
147     elif act == "q":
148         os.system('cls' if os.name == 'nt' else 'clear')
149         while True:
150             with open('2048ranking.txt', 'r') as f:
151                 a = f.read()
152                 data = ast.literal_eval(a)
153                 for i in data:
154                     print(f"Name: {i}, Score: {data[i]}")
155             input("Enter to continue!")
156             break
```

在玩家選擇查看分數後，程式會讀取 2048ranking.txt，並列出
使用循環以停留程式查看分數

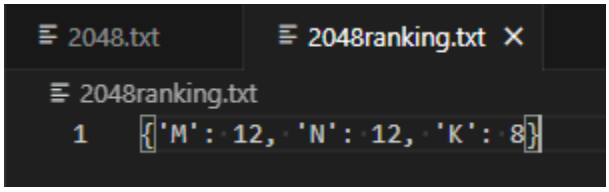
```
160     elif act == "e":
161         name = input("Enter your name or just enter if you don't want to save the score:")
162         with open('2048.txt', 'w') as f:
163             f.write(str(numlist))
164             f.write(f"\n{score}")
165         if name != "":
166             with open('2048ranking.txt', 'r') as f:
167                 s = f.read()
168                 if s:
169                     data = ast.literal_eval(s)
170                 else:
171                     data = {}
172             with open('2048ranking.txt', 'w') as f:
173                 try:
174                     t = data[name]
175                 except KeyError:
176                     data[name] = score
177                     print("You are new player! You have been added in to ranking.")
178                 else:
179                     if data[name] < score:
180                         data[name] = score
181                         print("You have played before and you get a heighter scores!")
182                     else:
183                         print("You have played before and you get more scores last time.")
184                 f.write(str(data))
185         break
```

當玩家選擇離開，程式會先詢問玩家是否儲存分數和排名，然後將玩家暱稱和分數儲存在 2048ranking.txt

程式會將計分儲存在 2048.txt 的第二行:



同時，將排名儲存在 2048ranking.txt:



2.3.9 排序功能

接下來，我們在排名系統新增一個排序功能：

```
148     elif act == "q":
149         with open('2048ranking.txt', 'r') as f:
150             a = f.read()
151             data = ast.literal_eval(a)
152             while True:
153                 os.system('cls' if os.name == 'nt' else 'clear')
154                 for i in data:
155                     print(f"Name: {i}, Score: {data[i]}")
156                 if input("input s to sort or just enter to exit.") == "s":
157                     keys = list(data.keys())
158                     values = list(data.values())
159                     sorted_value_index = np.argsort(values)[::-1]
160                     data = {keys[i]: values[i] for i in sorted_value_index}
161                 else:
162                     break
```

程式碼中引入了 numpy 庫的 argsort()函數，在排序後對齊暱稱

現在，排名系統可以進行排名：

排序前：

```
Name: M, Score: 12
Name: N, Score: 20
Name: K, Score: 8
input s to sort or just enter to exit.
```

排序後：

```
Name: N, Score: 20
Name: M, Score: 12
Name: K, Score: 8
input s to sort or just enter to exit.
```

2.3.10 遊戲介面

遊戲介面是必須的，可以令玩家直接選擇不同功能和玩法，使用歡迎字眼讓玩家產生興趣

接下來，我們新增一個遊戲介面：

```
while True:
    os.system("cls" if os.name == "nt" else "clear")
    print("Welcome to 2048!")
    print("==================")
    print("Press 'n' to start a new game.")
    print("Press 'l' to load a saved game.")
    print("Press 'x' to exit.")
    
    choice = input("Your choice: ").lower()
    if choice == "n":
        numlist = startNum()
        score = 0
        moved = 0
        break
    elif choice == "l":
        if not loadGame():
            print("No saved game found.")
            input("Press enter to continue...")
            continue
        moved = 0
        break
    elif choice == "x":
        sys.exit()
    else:
        continue
```

現在我們可以見到遊戲介面

```
Welcome to 2048!
==================
Press 'n' to start a new game.
Press 'l' to load a saved game.
Press 'x' to exit.
Your choice:
```

2.3.11 注釋和函數解釋

最後，我們為程式新增注釋讓其他開發者明白代碼的意思

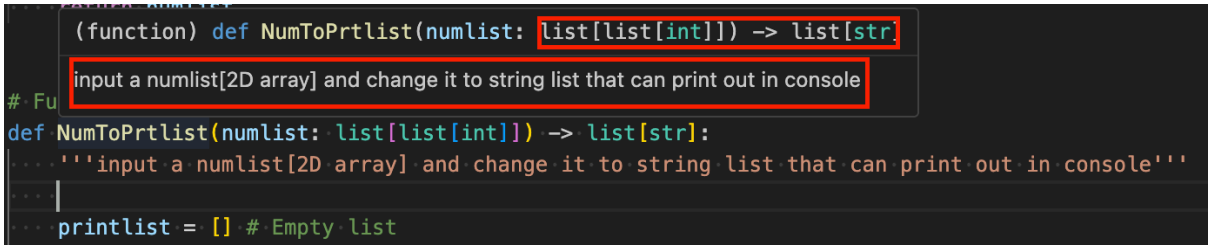
一個簡單的注釋例子:

```
# Function to convert the number list into a printable list
def NumToPrtlist(numlist):
    printlist = [] # Empty list
    for i in numlist:
        '''

        (1)
        Join each of "|" bewteen boxes.
        Example:
        {  2  |  2  |  2  |  2  }
        If the number = 0 it will show empty (5 space).
        Example:
        {     }

        (2)
        Add "|" in front and last of box
        Then it will be:
        {|  2  |  2  |  2  |  2  |}

        (3)
        Add middle line in each below box

        (4)
        Remove the last middle line because it will be close line

        '''
        row = "|".join(f"{n:^5}" if n != 0 else "     " for n in i) # (1)
        printlist.append(f"|{row}|") # (2)
        printlist.append("├─────┼─────┼─────┼─────┤") # (3)
    printlist.pop() # (4)
    return printlist
```

然後，為每個函數新增解釋，並列明每個變量的類別及返回類別，這樣可以增加代碼的可讀性下面是一個例子：

新增前：

新增後：



到此為止，遊戲設計暫時完成並順利運行

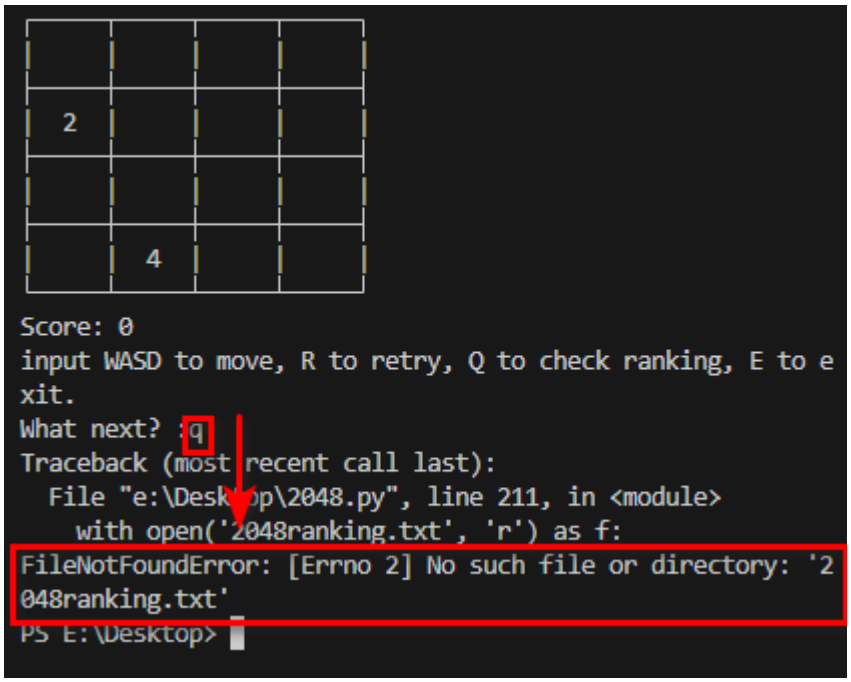
2.2 遊戲運行測試

遊戲運行測試是這個 2048 項目任務的非常重要的一部分。完成編寫軟體程式後需要運行測試的原因主要是確保軟體的功能正常、品質優良以及符合預期需求。運行測試可以幫助發現軟體中的錯誤、缺陷和漏洞。通過模擬不同的使用情境和輸入數據，測試可以驗證軟體的各個功能是否正常運作。此外，運行測試可以驗證軟體是否按照需求規格正確實現了各個功能。這有助於確保軟體能夠如預期般工作，並滿足項目的要求。

我將展示我發現的一個錯誤的例子，並且修復它。

2.3.1 FileNotFoundError 錯誤

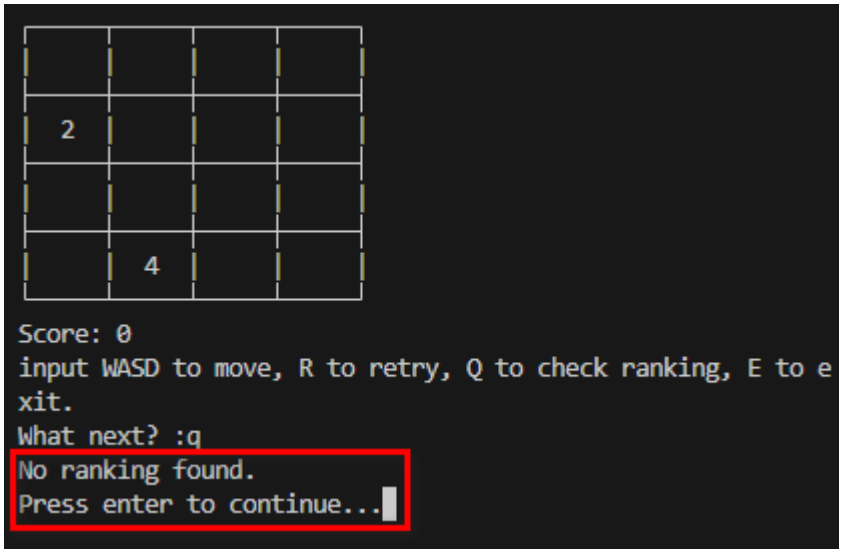
這個錯誤是當 2048ranking.txt 檔案被刪除或者消失時出現:



解決方法: 在代碼中新增 exists()函數

```
209     elif act == "q":
210         if os.path.exists("2048ranking.txt"):
211             with open('2048ranking.txt', 'r') as f:
212                 a = f.read()
213                 data = ast.literal_eval(a)
214                 while True:
215                     os.system('cls' if os.name == 'nt' else 'clear')
216                     for i in data:
217                         print(f"Name: {i}, Score: {data[i]}")
218                     if input("input s to sort or just enter to exit.").lower() == "s":
219                         keys = list(data.keys())
220                         values = list(data.values())
221                         sorted_value_index = np.argsort(values)[::-1]
222                         data = {keys[i]: values[i] for i in sorted_value_index}
223                     else:
224                         break
225         else:
226             print("No ranking found.")
227             input("Press enter to continue...")
228     elif act == "r":
229         numlist = startNum()
230         continue
231     elif act == "e":
232         name = input("Enter your name or just enter if you don't want to save the score:")
233         with open('2048.txt', 'w') as f:
234             f.write(str(numlist))
235             f.write(f"\n{score}")
236         if name != "":
237             if os.path.exists("2048ranking.txt"):
238                 with open('2048ranking.txt', 'r') as f:
239                     s = f.read()
240                     if s:
241                         data = ast.literal_eval(s)
242                     else:
243                         data = {}
244             else:
245                 data = {}
```

現在如果檔案不存在會直接在終端顯示並不會出現錯誤:



文件的 exists()函數和 try: … except: … 是程式中常用的除錯函數，在程序中經常出現，以避免程序出現錯誤時直接結束程式:

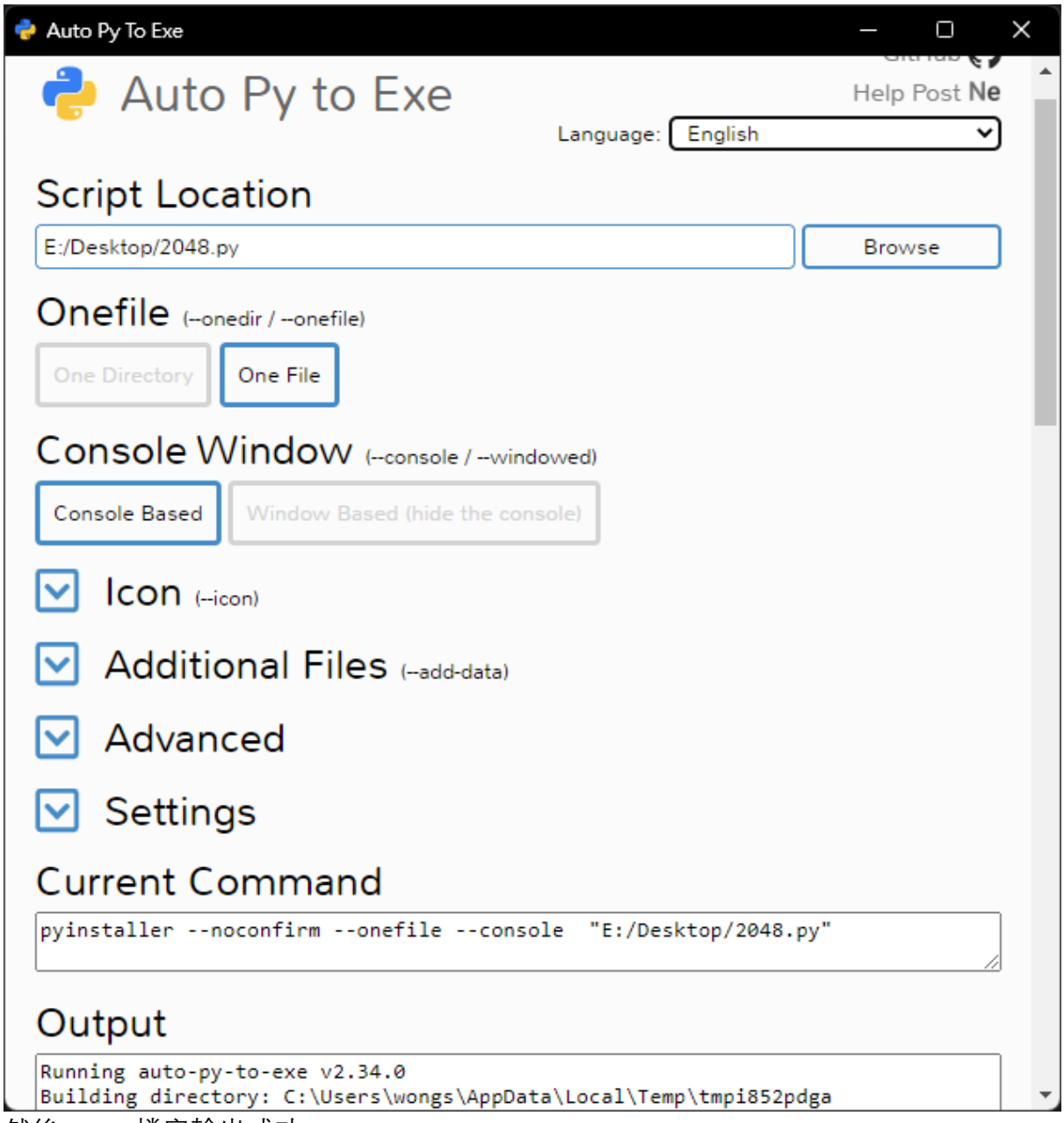
```
174  # Function to load the saved game state
175  def loadGame():
176      global numlist, score, gameRec, scoreRec
177      if os.path.exists("2048.txt"): # Check is the file is available or not
178          with open("2048.txt", "r") as f:
179              # Read the game last played
180              s = f.readlines()[0]
181              gameRec = ast.literal_eval(s)
182              print(gameRec)
183              numlist = gameRec[-1]
184
185          with open("2048.txt", "r") as f:
186              # Read the score
187              try:
188                  s = f.readlines()[1]
189                  scoreRec = ast.literal_eval(s)
190                  score = scoreRec[-1]
191              except IndexError:
192                  score = 0
193              return True
194      else:
195          return False
```

2.3 遊戲用戶測試

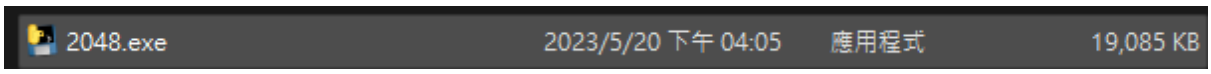
固然我們已經完成遊戲的測試並確保沒有運行錯誤的存在，我們依然需要把程序給於用戶進行測試，這樣不但可以更容易發現不可估算的錯誤，而且用戶的建議可以令到程式變得更好。

2.3.1 可執行檔案輸出

在將遊戲主文件發送給用戶之前，我們需要將該文件從.py 格式更改為.exe 格式。我們可以使用 auto-py-to-exe 模塊將.py 文件轉換為.exe 文件。我們這樣做是為了保護我們的源代碼，不讓其他人複製它。

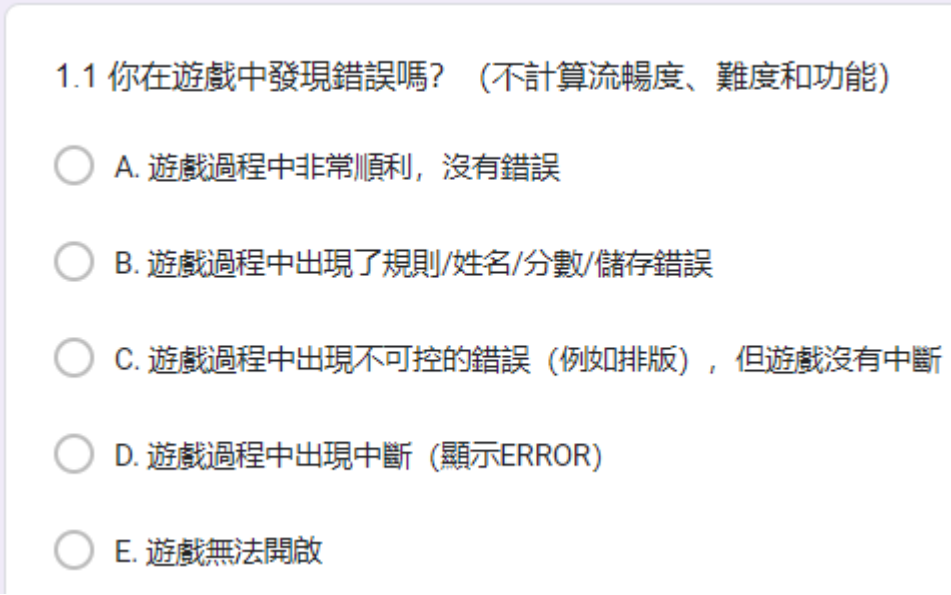


然後，exe 檔案輸出成功：：



2.3.3 製造問卷調查

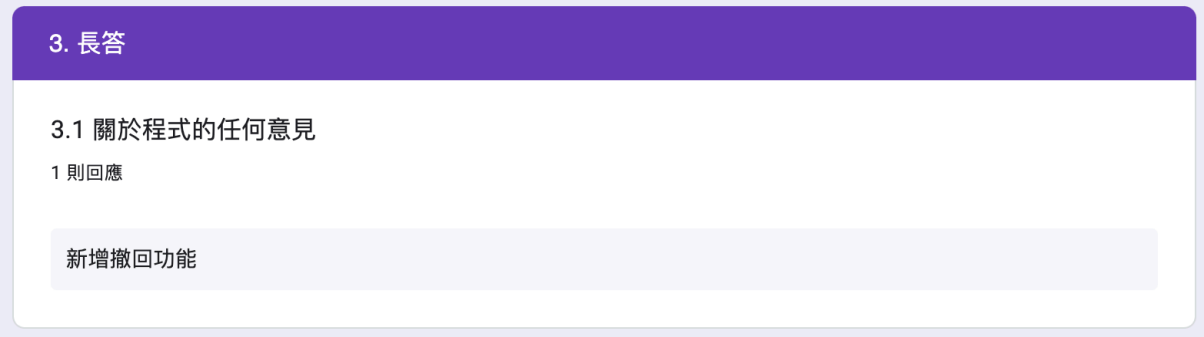
在玩家完成測試後，我們需要玩家給予我們一些意見，我們需要製造一份 Google 表單，讓測試玩家給予我們評語:



第一題有特定後續，方便玩家提供錯誤原因

第二三題為評分和選擇題

2.3.4 調查結果



看到有用戶希望增加撤回功能，撤回功能不但可以降低用戶的遊戲難度，更可避免用戶因操作失誤而導致遊戲失敗，大大提升遊戲體驗。

有關撤回功能的程序在下一節中介紹。

2.3.5 撤回功能

在程式內增加一個撤回功能：

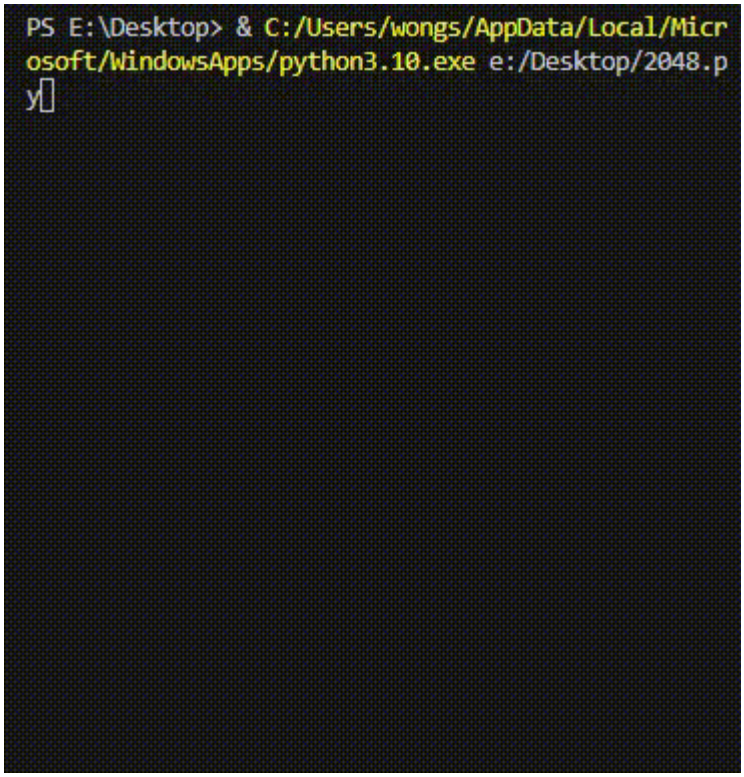
```
260     # Reverse
261     if act == "z":
262         # Check can it reverse or not
263         try:
264             # If yes, reverse it
265             numlist = gameRec[-2]
266             score = scoreRec[-2]
267             gameRec.pop()
268             gameRec = copy.deepcopy(gameRec)
269             scoreRec.pop()
270             scoreRec = copy.deepcopy(scoreRec)
271 
272         except IndexError:
273             # If no, tell user
274             print(gameRec)
275             print("It is already the first one, can't reverse.")
276             input("Press enter to continue...")
277         print(numlist)
```

同時，更新遊戲介面的控制面板

```
241     # Get user input for moving, checking ranking, retrying, or exiting
242     print("=\n|W/A/S/D: move\n|R: retry\n|Q: check ranking\n|Z: reverse\n|E: save and exit\n|X: exit without save\n="\
243         if not over else "=\n|GAME OVER!\n|R: retry\n|Q: check ranking\n|Z: reverse\n|E: save and exit\n|X: exit without save\n=")
```

撤回功能利用數據結構「棧」實現，有著先入後出，後入先出的特點。

更新後，遊戲面板有撤回功能，玩家在遊戲過程可以撤回



3 重新審視程式

雖然我們已經完成了 2048 遊戲的程式，並且它的使用非常完美，但我將解釋為什麼我們在程序中使用這些方法或函數，並談談函數的優點。同時，我還將討論一些我在程序中可以改進的方法。

3.1 為何使用 Numpy 庫？

為什麼在這個程式中使用 NumPy 呢?

NumPy 是一個流行的用於科學計算的 Python 庫，它提供了對大型多維數組和矩陣的支持，以及一系列操作這些數組的數學函數。它被設計成能夠高效處理數值計算，因此在數值分析和數據處理任務中是一個很好的選擇。

對於 2048 遊戲來說，NumPy 可以用於將遊戲板表示為二維數組，這樣可以更輕鬆地執行操作，比如移動方塊和合併相鄰方塊。它還提供了方便的函數來索引和切片數組，這可以用於訪問和操作遊戲板的特定部分。

此外，NumPy 還提供了高效的數學操作的實現，如加法、乘法和指數運算，這些在遊戲邏輯中常常用到。這些函數針對性能進行了優化，相比純 Python 實現，它們可以提供顯著的加速。

總的來說，在 2048 遊戲中使用 NumPy 可以簡化實現並改善遊戲邏輯的性能。

3.2 使用面向對象編程會更好？

是的，在 2048 遊戲的程式中，我可以使用類別（classes）。類別是 Python 中面向對象編程的基本組件，可以將數據和行為封裝在一個單元中。在遊戲的上下文中，類別可以用來表示遊戲板、方塊以及遊戲邏輯。

例如，我可以定義一個 Tile 類別，表示遊戲板上的單個方塊。Tile 類別可以具有屬性，如方塊的值和在遊戲板上的位置，以及用於合併其他方塊和更新其值的方法。

同樣地，我可以定義一個 Board 類別，代表整個遊戲板，其中包含了移動方塊和合併相鄰方塊的方法。Board 類別還可以具有檢查遊戲是否結束（即無法再進行更多移動）以及在遊戲板上生成新方塊的方法。

以這種方式使用類別可以幫助組織程式碼，使其更易於理解和維護。它還提供了一種自然的方式將遊戲邏輯和數據封裝到單獨的單元中，使得更容易理解和修改遊戲的行為。

3.3 其他提升

我可以通過幾種方式來改進 2048 遊戲程式：

添加圖形化用戶界面（GUI）：目前遊戲在終端中進行，但添加圖形化用戶界面可以使遊戲對用戶更具互動性和吸引力。

添加難度級別：我可以添加不同的難度級別，如簡單、中等和困難，為玩家提供更大的挑戰。

添加動畫效果：在遊戲中添加動畫效果，例如在方塊合併時平滑過渡，可以增強用戶體驗，使遊戲更具視覺吸引力。

優化性能：儘管目前的遊戲實現相對簡單高效，但仍然可以進行優化以提高性能。例如，我可以使用專為滑動方塊拼圖設計的算法，以提高遊戲邏輯的效率。

重構程式碼：重構程式碼可以使其更易於閱讀、理解和維護。我可以將程式碼分解為更小、更模塊化的函數或類，並刪除冗餘的程式碼以簡化實現。

這只是我可以改進 2048 遊戲程式的一些想法。最終，我所做的改進將取決於我對遊戲的目標和我想實現的複雜程度。

4 時間線 + 資料來源 + 鳴謝

以下是設計和製造 2048 遊戲時間線，各種資料來源，軟件的使用和鳴謝。這些為我的遊戲設計和報告提供幫助。

4.1 時間線

日期	進度
6 月	收到題目《2048 遊戲》 釐清和分析問題
7 月	找出問題的解決方法
8 月	編寫程序
9 月	編寫程序和用戶測試
10 月	代碼除錯和解釋
11 月	反思和總結
12 月 6 日	繳交 SBA 報告和代碼

4.2 資料來源

1. 2048 簡介
   Wiki
   https://en.wikipedia.org/wiki/2048_(video_game)

2. Python3 的使用方法
   Runoob
   https://www.runoob.com/python3/python3-tutorial.html

3. Python3 的使用方法
   W3Schools
   https://www.w3schools.com/python/default.asp

4.3 軟件使用

1. VSCode
   編程 IDE

2. Word
   報告

3. Python3
   編程語言

4. Chrome
   使用 Google Search 尋找代碼錯誤的解決方案和資料收集

4.4 鳴謝

1. Mr. Tam
   我的 ICT 老師，幫助我解決問題和測試程式

2. Nathan To
   我的同學，幫助我測試程式，提供改善建議