方格程式－數獨

程式碼展示(僅部分程式碼)

```python
import tkinter as tk
import random
import copy
import time
from tkinter import messagebox

def is_valid(board, row, col, num):
    for i in range(9):
        if board[row][i] == num or board[i][col] == num: #檢查同行同列是否有相同數字
            return False
        if board[3*(row//3)+i//3][3*(col//3)+i%3] == num: #檢查同一個 3x3 宮格是否有相同數字
            return False
    return True

def solve(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0: #空白格子
                for num in range(1, 10):
                    if is_valid(board, row, col, num): #檢查數字是否符合數獨規則
                        board[row][col] = num
                        if solve(board):
                            return True
                        board[row][col] = 0
                return False
    return True

def generate_puzzle():
    board = [[0]*9 for _ in range(9)]
    solve(board)  # 先生成一組完整的解
    puzzle = copy.deepcopy(board)

    attempts = 40  # 再挖掉格子數量（控制難度）
    while attempts > 0:
        row, col = random.randint(0, 8), random.randint(0, 8)
        if puzzle[row][col] != 0:
            puzzle[row][col] = 0
            attempts -= 1
    return puzzle

class SudokuGame:
    def __init__(self, master):   #方法 (建構子)
        self.master = master      #儲存 Tkinter 的主視窗物件
        master.title("數獨遊戲")  #視窗的標題

        self.puzzle = generate_puzzle()            #生成一個新的數獨謎題並儲存
        self.solution = copy.deepcopy(self.puzzle) #複製謎題並解決它，得到答案
        solve(self.solution)

        self.entries = [[None for _ in range(9)] for _ in range(9)] #9x9 的二維列表
        self.error_count = 0  #錯誤計數器
        self.start_time = time.time() #記錄遊戲開始的時間
        self.timer_running = True
```

構建基本數獨盤面

加入：
- 錯誤顯示
- 提示&答案顯示按鈕
- 計時器
- 隨機產生數獨題目

美化介面：
- 錯誤數字顯示紅色
- 提示數字顯示藍色
- 九宮格外框加粗

Python專題

應用到的技巧：
- GUI處理技術
- 錯誤檢查
- 計時器
- 提示&答案顯示

11301025A楊寓涵
11301052A孫玉妍
指導老師：程凱

	5	6		7	
4			3	5	
9			1		5
	8			6	
2			4		
3		1			

貢獻度說明

11301025A楊寓涵 50%
負責程式碼

11301052A孫玉妍 50%
負責海報、簡報製作

專題-Python作業自動批改系統

報告人：11303105A李瑋鈺、11303069A賴偉儒

報告日期：6/3(二)

目錄

HackMD連結:

HackMD
https://hackmd.io/VNIr06mmQhehrk_xJHBHtA?view=

題目說明

Python 作業自動批改系統

需要的功能有

1. 輸入測資產生出測資的txt檔和對應測資答案的txt檔
2. 自動產生出需要的資料夾並取名成對應名字
3. 執行被批改的程式並比對執行結果和輸出學生成績的txt檔
   (如果學生程式有錯誤輸出錯誤訊息到txt檔)
4. 將每位同學的成績輸出成csv檔

autoinputans.py

```python
import subprocess
import os

expy = {
    "1": "ex1.py",
    "2": "ex2.py",
    "3": "ex3.py"
}

os.makedirs("input", exist_ok = True)
os.makedirs("ans", exist_ok = True)


countdata = {}

while True:
    qid = input("請輸入題號，輸入a則結束：").strip()
    if qid.lower() == 'a':
        break

    if qid not in expy:
        print("找不到題號 " + qid + " 的參考解答，檢查expy。")
        continue

    solutionfile = expy[qid]

    if qid not in countdata:
        countdata[qid] = 0


    while True:
        data = input("請輸入測資，直接按Enter回到輸入題號：").strip()
        if data == "":
            break

        countdata[qid] += 1
        idindex = countdata[qid]

        inputfile = "input/" + qid + "_" + str(idindex) + ".txt"
        ansfile = "ans/" + qid + "_" + str(idindex) + ".txt"

        with open(inputfile, "w") as file:
            file.write(data + "\n")

        result = subprocess.run(
            ['python', solutionfile],
            input = data,
            stdout = subprocess.PIPE,
            text = True
        )

        with open(ansfile, "w") as file:
            file.write(result.stdout.strip() + "\n")

        print("已產生：" + inputfile + " 與 " + ansfile)

print("\n測資及答案已產生完")

```

correctsup.py

```python
import subprocess
import os
import compare

ansscore = {
    "1": 100,
    "2": 50,
    "3": 10
}

def testcasecorrectsup(hwpath, qid, inputfile = "input", ansfile = "ans", timeout = 5):
    txt = []
    passcount = 0
    totalcount = 0

    fullscore = ansscore[qid]

    inputfiles = []
    for name in os.listdir(inputfile):
        if name.startswith(qid + "_") and name.endswith(".txt"):
            inputfiles.append(name)
    inputfiles.sort()

    if not inputfiles:
        txt.append("找不到測資檔案" + qid + "_*.txt")
        return txt, 0.0, fullscore

    for input_file in inputfiles:
        inputpath = os.path.join(inputfile, input_file)
        anspath = os.path.join(ansfile, input_file)

        if not os.path.exists(anspath):
            txt.append("缺少答案檔案 " + anspath)
            continue

        with open(inputpath, 'r') as file:
            testcase = file.read()
        with open(anspath, 'r') as file:
            answer = file.read().strip()

        try:
            result = subprocess.run(
                ['python', hwpath],
                input = testcase,
                stdout = subprocess.PIPE,
                stderr = subprocess.PIPE,
                text = True,
                timeout = timeout
            )
        except Exception as err:
            txt.append("測資 " + input_file + " 執行錯誤：" + str(err))
            continue

        output = result.stdout.strip()
        error = result.stderr.strip()

        if error:
            txt.append("測資 " + input_file + " 程式錯誤：\n" + error)
            continue

        totalcount += 1

        if output == answer:
            txt.append("測資 " + input_file + " 正確")
            passcount += 1
        else:
            txt.append("測資 " + input_file + " 錯誤")
            txt.append("預期輸出：")
            txt.append(answer)
            txt.append("實際輸出：")
            txt.append(output)
            compare.compare(output, answer)

    if totalcount == 0:
        return txt, 0.0, fullscore

    score = (passcount / totalcount) * fullscore
    studentresult = "第 " + qid + " 題 答對 " + str(passcount) + "/" + str(totalcount) + " 測資，得分：" + str(round(score, 1))
    txt.append(studentresult)

    return txt, round(score, 1), fullscore
```

currect.py

```python
import os
import correctsup

studentsfile = "students"
inputfile = "input"
ansfile = "ans"
homeworknum = ['1', '2', '3']

def testcasecorrect(studentid):
    hwfile = os.path.join(studentsfile, studentid, "hw")
    totalscore = 0.0
    fullscore = 0.0
    txt = []
    scoredict = {}

    txt.append("開始批改 " + studentid)
    txt.append("=" * 30)

    for hwid in homeworknum:
        hwpath = os.path.join(hwfile, hwid + ".py")
        header = "作業 " + hwid + "："
        print("學生 " + studentid + "：" + header, end=' ')
        txt.append(header)

        if not os.path.exists(hwpath):
            msg = "未繳交，不計分"
            print(msg)
            txt.append(msg)
            scoredict[hwid] = 0.0
            continue

        txts, scoregot, fullscore555 = correctsup.testcasecorrectsup(hwpath, qid = hwid)

        for line in txts:
            print(line)
            txt.append(line)

        resultline = "得分：" + str(scoregot) + " / " + str(fullscore555)
        print(resultline)
        txt.append(resultline)

        totalscore += scoregot
        fullscore += fullscore555
        scoredict[hwid] = scoregot

    txt.append("=" * 30)
    studentresult = "學生 " + studentid + " 總得分：" + str(round(totalscore, 2)) + " / " + str(round(fullscore, 2))
    print(studentresult)
    txt.append(studentresult)

    txtpath = os.path.join(studentsfile, studentid, studentid + ".txt")
    with open(txtpath, "w", encoding = "utf-8") as file:
        for line in txt:
            file.write(line + "\n")

    print("結果已寫入：" + txtpath + "\n")
    return totalscore, fullscore, scoredict

```

main.py

```python
import os
import currect
import csv

studentsfile = "students"
studentids = []

homeworknum = ['1', '2', '3']

for i in os.listdir(studentsfile):
    path = os.path.join(studentsfile, i)
    if os.path.isdir(path):
        studentids.append(i)

print("有 " + str(len(studentids)) + " 位學生")

allstudentsscores = []

for sid in studentids:
    print("開始批改學生：" + sid)
    totalscore, fullscore, scoredict = currect.testcasecorrect(sid)

    studentrow = [sid]
    for qid in homeworknum:
        studentrow.append(scoredict.get(qid, 0.0))
    studentrow.append(round(totalscore, 2))
    allstudentsscores.append(studentrow)

csvpath = "studentscores.csv"
with open(csvpath, "w", newline = "", encoding = "utf-8") as csvfile:
    writer = csv.writer(csvfile)
    header = ["學生"] + ["題 " + qnum for qnum in homeworknum] + ["總分"]
    writer.writerow(header)
    for studentrow in allstudentsscores:
        writer.writerow(studentrow)

print("已匯出總成績表：" + csvpath)
```

使用方法

準備工作

在autoinputans.py裡面的第四行更改成你希望批改的數量和檔案名稱

”1”為第一題有更多題就照著寫下去

“ex1.py”為學生檔案的名稱也可照需求調整

在correct.py裡面的第四行也需更改成你所需要的題目數量

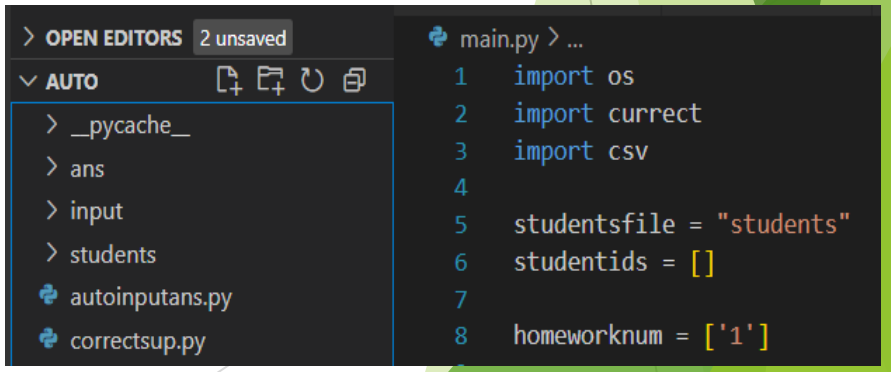
在第十行的地方可以更該檔案路徑hwfile = os.path.join(studentsfile,(此行為批改編號),(這格可刪除但txt和程會在同個資料夾))

在main的第八行也需要一起更改有幾行

第五行的studentsfile =“(為你放批改資料的資料夾名稱)”

```
4   expy = {
5       "1": "ex1.py",
6       "2": "ex2.py",
7       "3": "ex3.py"
8   }
```

```
4   studentsfile = "students"
5   inputfile = "input"
6   ansfile = "ans"
7   homeworknum = ['1', '2', '3']
8
9   def testcasecorrect(studentid):
10      hwfile = os.path.join(studentsfile, studentid, "hw")
```



使用方法

準備工作

在correctsup.py中可以更改每題的題目配分

```
4   ansscore = {
5       "1": 100,
6       "2": 50,
7       "3": 10
8   }
```

需再資料夾增加ex(題號).py為範例程式(記住input不能有提示字元)有幾題就需要幾個

Run autoinputans.py 輸入測資自動產生測資和答案的txt

使用方法

Run main.py跑完就會在學生資料夾內產生出每個學生專屬的txt檔和跑出一個額外的csv檔可供預覽所有學生的分數表

成果展現:

```
開始批改 11303063A
==========
作業 1

測資 1 1.txt 正確
測資 1 2.txt 正確
測資 1 3.txt 正確
第 1 題 答對 3/3 測資，得分：100.0
得分：100.0 / 100
作業 2

測資 2 1.txt 正確
第 2 題 答對 1/1 測資，得分：90.0
得分：90.0 / 90
作業 3

測資 3 1.txt 正確
第 3 題 答對 1/1 測資，得分：80.0
得分：80.0 / 80
作業 4

測資 4 1.txt 正確
第 4 題 答對 1/1 測資，得分：70.0
得分：70.0 / 70
作業 5

測資 5 1.txt 正確
測資 5 2.txt 正確
第 5 題 答對 2/2 測資，得分：60.0
得分：60.0 / 60
==========
學生 11303063A 總得分：400.0 / 400.0
```

```
開始批改 11303016A
==========
作業 1

未繳交，不計分
作業 2

測資 2 1.txt 執行錯誤：Command '['python', 'students\\11303016A\\hw\\2.py']' timed out after 5 seconds
得分：0.0 / 90
作業 3

未繳交，不計分
作業 4

未繳交，不計分
作業 5

未繳交，不計分
==========
學生 11303016A 總得分：0.0 / 400.0
```

使用方法

成果展現:

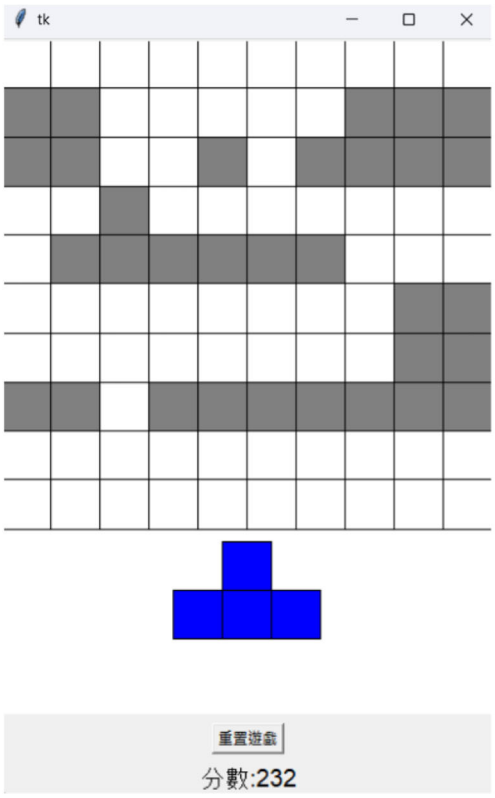
學生	題 1	題 2	題 3	題 4	題 5	總分
11303016A	0	0	0	0	0	0
11303053A	0	0	0	0	0	0
11303063A	100	90	80	70	60	400
11303069A	100	0	0	0	0	100

Thank You

113-2程式設計課程成果展
方塊拖放遊戲

指導老師:程凱

小組成員:吳宸宜 許鳳芫



簡介:使用Python的Tkinter套件為基礎所製作的方塊拼圖遊戲。

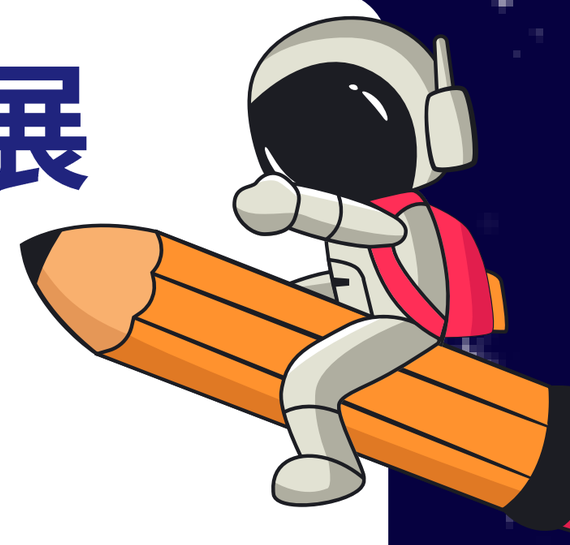
玩法:玩家需從畫面底部拖曳方塊至棋盤中，試圖完整填滿一整行或一整列以進行消除獲得分數。

整體流程

玩家拖動方塊→判斷是否可放置→可放就加分並清除行及列→補充新方塊→持續進行直到無法操作為止

113-2 程式設計課程成果展

似是而非



指導老師　程凱

電機一　11301034A　葉雨儒

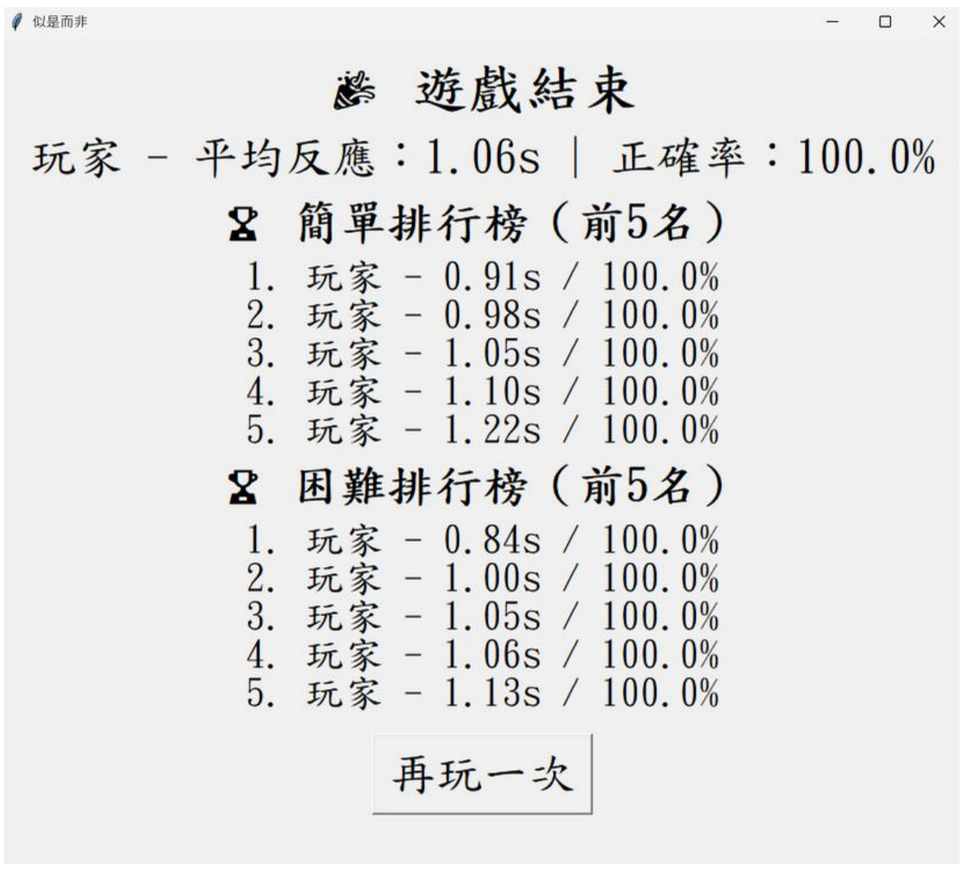
電機一　11301056A　張凱翔

遊戲設計

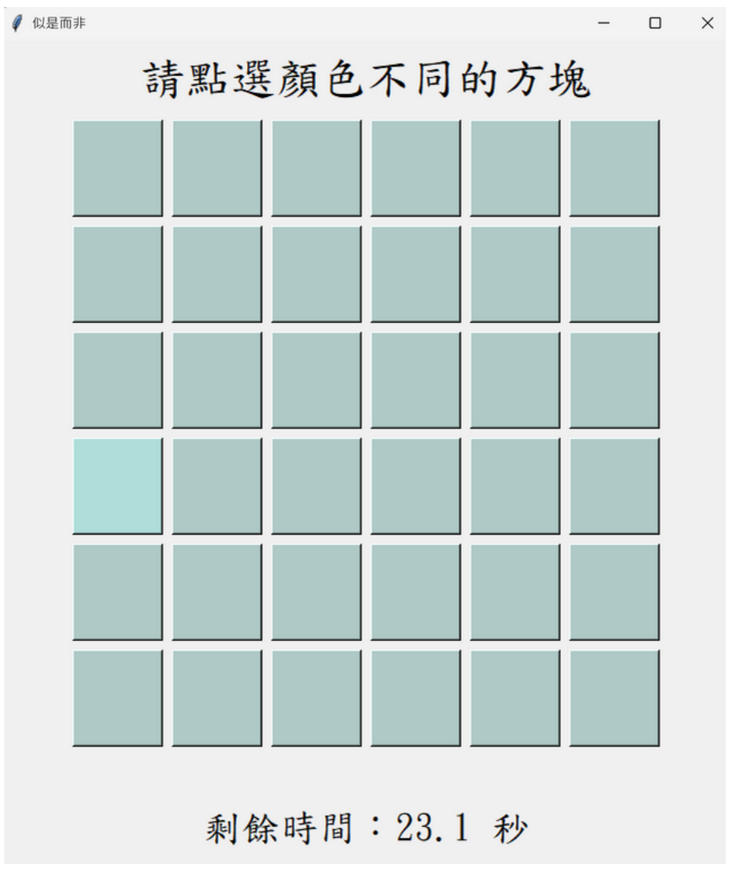
色差會隨著關卡逐步縮小，提升辨識難度
提供簡單與困難模式，適合不同挑戰程度
設計限制與累計時間機制，增加挑戰節奏
遊戲結束紀錄成績，統計前五位之紀錄

技術說明

使用 Python+Tkinter 製作互動介面
以 Random + RGB 色值控制色差範圍
利用 JSON 儲存紀錄與製作排行榜



排行榜示意圖



遊戲中畫面

智慧科技學院程式設計專題海報

點餐系統

這是一個用 Python 寫的點餐系統。功能包含：菜單顯示、點餐與購物車、結帳與總金額計算折扣優惠以及明細顯示。內容有程式設計的基礎概念，像是資料結構、迴圈、條件判斷，還有基本的錯誤處理。這個程式的優點在於結構清晰易懂、變數命名直觀，使得程式碼易於閱讀和理解，方便後續的維護和功能擴充。

```
==== 菜單 ====
鮮脆雞腿堡: 66 元
波浪薯: 40 元
無骨雞塊: 47 元
香酥米糕: 33 元
超長熱狗: 27 元
可樂: 20 元
熱咖啡: 30 元
玉米濃湯: 26 元
================
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：鮮脆雞腿堡 2
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：波浪薯 3
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：超長熱狗 1
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：送出
==== 購物車 ====
鮮脆雞腿堡 x 2 = 132 元
波浪薯 x 3 = 120 元
超長熱狗 x 1 = 27 元
總金額: 279 元
```

```
==== 菜單 ====
鮮脆雞腿堡: 66 元
波浪薯: 40 元
無骨雞塊: 47 元
香酥米糕: 33 元
超長熱狗: 27 元
可樂: 20 元
熱咖啡: 30 元
玉米濃湯: 26 元
================
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：可樂 2
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：超長熱狗 3
請輸入餐點與數量（格式：餐點 數量），點餐完畢後如需結帳則輸入[送出]：送出
==== 購物車 ====
可樂 x 2 = 40 元
超長熱狗 x 3 = 81 元
總金額: 121 元
```

總體來說，這個 Python 點餐系統程式 是一個設計良好、功能實用的範例。它不僅展示了 Python 在處理基本使用者互動、資料結構（如字典）和流程控制（如迴圈）方面的能力，更在使用者體驗和錯誤處理上考慮周全。

電機系

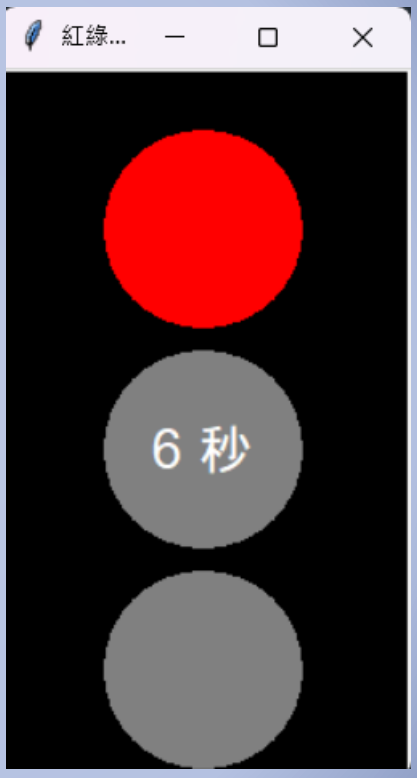
指導老師: 張恩誌

學生： 黃彥評 11301039A
吳岳倫 11301098A
陳楷程 11301074A
黃凱俊 11301082A

智慧科技學院程式設計專題海報

紅綠燈

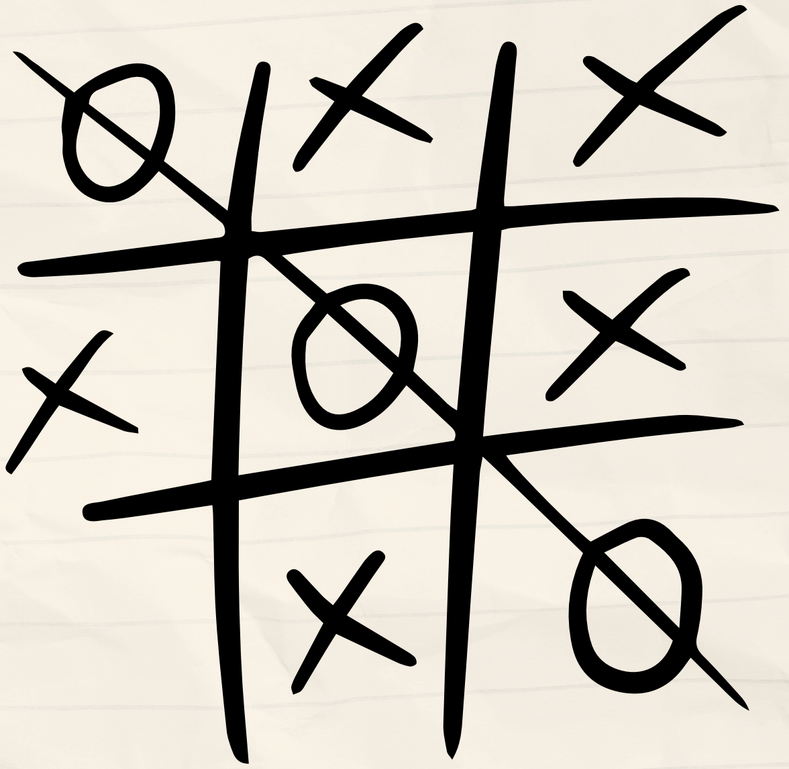
摘要：本程式使用 Tkinter 製作紅綠燈模擬器，讓使用者可輸入紅、黃、綠燈的秒數，模擬真實交通燈的切換過程與倒數時間顯示。



結論：透過本程式，使用者能直觀學習狀態切換與倒數邏輯，並熟悉 Tkinter 畫布與互動式 GUI 的應用與設計概念。

電機系
指導老師: 張恩誌
學生: 周祐陞11322047A、許桐銘11322028A、黃聖倫11322023A、劉耘成11322043A

程式設計



設計圈圈叉叉遊戲(人v.s電腦)

題目參考 : b121. 井字遊戲 (TTT)

11301096A 王豐造
11302020A 翁呈光

報告日期 : 2025/6/12

目錄

題意說明

如題：井字遊戲
就是在井字內三個連成一線者獲勝，反之平手

參考於 b121. 井字遊戲 (TTT)
它原本是要判斷一局未完成的對局
但我們想要讓這個程式是要和電腦對決一場遊戲

解題方法

遊戲目標：3x3棋盤上先連成一條線（橫、直、斜）者勝，無法連線則平手。

介面設計：

- 使用 Tkinter 建立視覺棋盤與按鈕。
- 玩家點擊空格下「X」，電腦下「O」。

遊戲流程：

- 玩家選擇難度（簡單、普通、困難）後開始。
- 玩家與電腦輪流下棋。
- 每步判斷勝負或平手。
- 遊戲結束顯示結果，提供重玩功能。

程式重點：

- 棋盤狀態用陣列管理。
- 按鈕與陣列同步更新。
- 遊戲結束後禁用按鈕避免繼續下棋。

電腦策略差異：

1. 簡單：隨機下棋。
2. 普通：先阻擋玩家可能勝利，否則隨機。
3. 困難：先尋找自己勝利位置，無則阻擋玩家，再無則隨機。

判斷勝負方法：透過列舉8種可能勝利組合，判斷是否達成連線。

```python
import tkinter as tk  # 匯入 tkinter 庫來製作 GUI（圖形介面）
from tkinter import messagebox  # 匯入 tkinter 的訊息方塊功能
import random  # 匯入隨機模組，供電腦選擇格子用

# 開始遊戲的函式
def start_game():
    global difficulty  # 宣告全域變數 difficulty
    diff = difficulty.get()  # 取得使用者選擇的難度
    if diff not in ["簡單", "普通", "困難"]:  # 如果選的不是三種難度之一
        diff = "簡單"  # 預設為「簡單」
    window.diff = diff  # 把難度存進 window 裡面（當作屬性）
    for widget in window.winfo_children():  # 找出視窗裡所有元件
        widget.destroy()  # 把這些元件都移除（清空畫面）
    create_game_screen()  # 建立遊戲畫面（畫出井字格）

# 建立遊戲畫面的函式
def create_game_screen():
    global board, buttons  # 宣告全域變數
    board = [""] * 9  # 建立 9 格棋盤，每格預設為空字串
    buttons = []  # 建立按鈕列表，稍後把每個格子的按鈕放進來

    # 建立 3x3 的棋盤按鈕
    for i in range(9):
        btn = tk.Button(window, text="", width=6, height=3, font=('Arial', 24),
                        command=lambda i=i: player_move(i))  # 按鈕按下去時呼叫 player_move(i)
        btn.grid(row=i//3, column=i%3, padx=5, pady=5)  # 決定按鈕顯示在第幾列第幾欄
        buttons.append(btn)  # 把這個按鈕加入列表

    # 建立「重來」按鈕
    restart_btn = tk.Button(window, text="重來", command=restart_game)  # 點擊後呼叫 restart_game
    restart_btn.grid(row=3, column=0, columnspan=3, pady=10)  # 放在最下面（橫跨三欄）

# 玩家下棋的函式
def player_move(i):
    if board[i] == "":  # 如果這一格是空的
        board[i] = "X"  # 玩家放 X
        buttons[i].config(text="X", state="disabled")  # 更新畫面並停用這個按鈕

        # 檢查玩家是否贏了或平手
        if check_win("X"):  # 玩家贏了
```

Tkinter 是 Python 內建的圖形介面（GUI）套件，
用來製作視窗、按鈕、標籤、輸入框等視覺化應用程式。

Arial 是一種簡單、清晰、常用的字體，
適合做介面、文件、標題。

lambda 就是快速定義一個小函式，通常用在需要「一次性小功能」的地方。

for widget in window.winfo_children():
這行是用迴圈逐一取出清單裡的每個元件。
widget.destroy():
這行會把目前迴圈裡的元件「刪除」掉，讓它不再顯示在視窗中，也釋放資源。

```
            game_over("你贏了！🎉")
            return
        elif is_draw():  # 平手
            game_over("平手！😊")
            return
        computer_move()  # 換電腦下棋

# 電腦下棋的函式
def computer_move():
    empty = [i for i, val in enumerate(board) if val == ""]  # 找出所有還沒下的格子
    move = None  # 還沒決定要下哪一格

    # 簡單模式：隨機選一格
    if window.diff == "簡單":
        move = random.choice(empty)  # 隨機選一個空格

    # 普通模式：如果玩家下一步會贏，就先擋住
    elif window.diff == "普通":
        for i in empty:
            board[i] = "X"  # 假裝玩家在這格下
            if check_win("X"):  # 如果玩家會贏
                move = i  # 電腦就記住這格
            board[i] = ""  # 還原棋盤
        if move is None:
            move = random.choice(empty)  # 沒找到威脅就隨便選

    # 困難模式：先檢查自己能不能贏，再擋玩家，最後才隨機選
    elif window.diff == "困難":
        # 檢查自己有沒有贏的機會
        for i in empty:
            board[i] = "O"
            if check_win("O"):  # 如果這一格可以讓電腦贏
                move = i
            board[i] = ""
        # 如果沒辦法贏，再看玩家會不會贏
        if move is None:
            for i in empty:
                board[i] = "X"
                if check_win("X"):  # 玩家會贏就擋住
                    move = i
```

is_draw()」 是一個會回傳「是或不是」的檢查功能

def computer_move():
定義一個名叫 computer_move 的功能（電腦要怎麼下棋）

empty = [...] 建立一個清單(叫做 empty)放入「棋盤上所有還沒下棋的格子」

enumerate(board) 把 board 中每一格都編號（第幾格＋內容）

```
                board[i] = ""
        # 如果也沒有人快贏，就隨機選
        if move is None:
            move = random.choice(empty)

    # 電腦實際下棋
    board[move] = "O"
    buttons[move].config(text="O", state="disabled")  # 顯示在畫面上並停用

    # 檢查電腦是否贏或平手
    if check_win("O"):
        game_over("電腦贏了！😄")
    elif is_draw():
        game_over("平手！😄")

# 檢查有沒有人贏
def check_win(player):
    win_cases = [(0,1,2),(3,4,5),(6,7,8),  # 三橫列
                 (0,3,6),(1,4,7),(2,5,8),  # 三直行
                 (0,4,8),(2,4,6)]  # 兩個對角線
    # 檢查這些排列中有沒有三個都是同一個玩家的棋子
    return any(board[a]==board[b]==board[c]==player for a,b,c in win_cases)

# 檢查是否平手（所有格子都不是空的）
def is_draw():
    return all(cell != "" for cell in board)

# 顯示遊戲結果的訊息
def game_over(msg):
    for btn in buttons:
        btn.config(state="disabled")  # 停用所有按鈕
    messagebox.showinfo("遊戲結束", msg)  # 顯示提示訊息

# 重來按鈕的功能
def restart_game():
    for widget in window.winfo_children():  # 清除所有畫面元件
        widget.destroy()
    create_game_screen()  # 重新建立畫面
```

disabled :
讓按鈕「變灰、不能按」，防止玩家再次點擊。

messagebox.showinfo:
顯示一個訊息框，提示使用者一個普通訊息

config :可以隨時修改元件外觀、行為，像改字、改顏色、禁用等。
非常常用，能讓視窗介面「動態變化」。

```
# 建立主視窗
window = tk.Tk()
window.title("簡化版井字棋")  # 設定視窗標題
window.geometry("500x500")  # 設定視窗大小（寬500 高500）

# 建立難度變數（簡單、普通、困難）
difficulty = tk.StringVar(value="簡單")  # 預設值是「簡單」

# 初始畫面：顯示標題、難度選單、開始按鈕
tk.Label(window, text="井字棋遊戲", font=('Arial', 18)).pack(pady=20)  # 標題
tk.Label(window, text="選擇難度：").pack()  # 難度說明文字
tk.OptionMenu(window, difficulty, "簡單", "普通", "困難").pack(pady=10)  # 下拉選單
tk.Button(window, text="開始遊戲", font=('Arial', 14), command=start_game).pack(pady=20)  # 開始按鈕

# 開始主迴圈（保持視窗顯示）
window.mainloop()
```

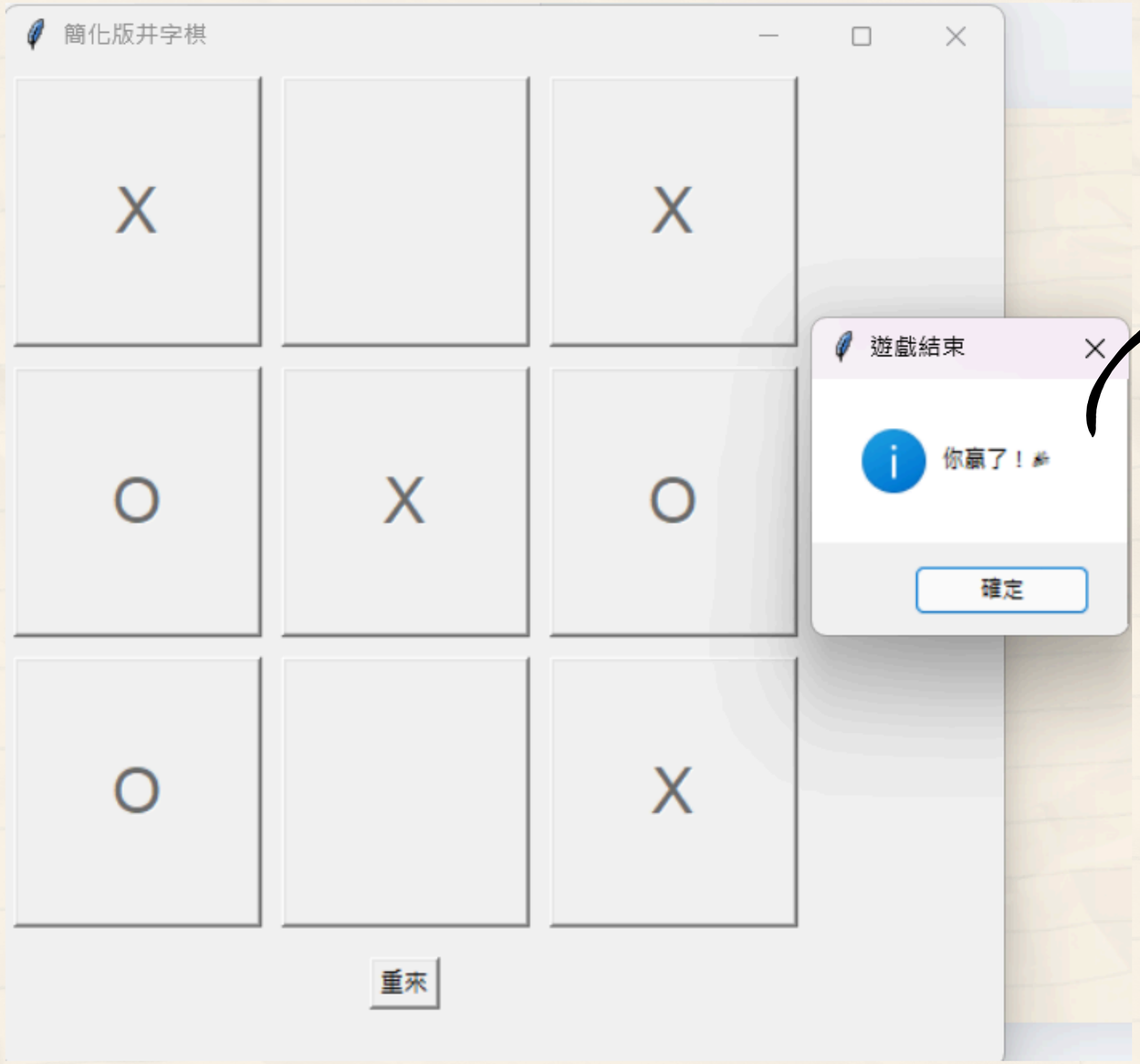
geometry :
用來設定主視窗的尺寸大小。

window.mainloop()功能：
讓 GUI 視窗不會一閃即逝
沒寫這行，視窗跑完程式就關閉了

程式執行

選單

程式執行



跳出win視窗

心得

這次撰寫井字遊戲的過程，讓我更清楚如何將遊戲規則轉換為具體的程式邏輯。
從設計棋盤結構、判斷勝負條件，到安排玩家的操作流程，每個步驟都需要仔細思考與規劃。
雖然中間遇到不少困難，但當我順利完成後，不僅提升了解決問題的能力，也讓我對程式設計產生更濃厚的興趣。