

陳

鴻

哲

目錄

Introduction:	4
A. 卡牌遊戲簡介	4
Chapter 1: 程序界面	5
1.1 登錄畫面	5
1.2 注冊畫面	5
1.3 遊戲畫面	6
Chapter 2: 用戶數據的處理	7
2.1 數據儲存設計和處理	7
2.2 數據檢驗	7
Chapter 3: 程序編寫:	9
3.1 程式總預覽	9
3.2 程式基礎設定	11
3.3 數據類型和數據結構	12
3.4 模塊化編寫	13
3.5 程序內數據交互	18
3.6 算法編寫	20
3.7 算法優化	24

Chapter 4: 程式運行測試:	25
4.1 組件單元測試.....	25
4.2 登錄與注冊界面測試.....	26
4.3 游戲界面測試.....	29
 Chapter 5: 錯誤處理:	 31
5.1 主程序異常響應.....	31
 Conclusion:	 32
A. 總結	32
 Index	

卡牌遊戲簡介

本項目是基于 Python 編程語言開發的德州撲克遊戲 (Texas Hold'em)。通过设计和编码成功地实现了德州撲克这一经典紙牌遊戲的數字化版本。玩家可以在圖像界面上体验到真实的德州撲克遊戲乐趣，与计算机进行對弈，展现自己的牌技和策略。

項目特点：

1. 使用 Python 編程語言實現，簡潔高效
2. 包含德州撲克遊戲的核心規則和邏輯，玩家能够体验真实的遊戲过程
3. 提供了全面的用戶交互界面，使遊戲非常具互動性
4. 考慮了牌局隨機性和策略性，增加了遊戲的挑戰性和趣味性

技術亮点：

1. 使用面向對象編程 (OOP) 的思想，合理划分遊戲中的各个對象和功能
2. 利用 Python 的數據結構和算法，實現了牌局的隨機洗牌、發牌功能和評估牌組
3. 通过条件语句和循环結構，實現了遊戲規則的邏輯判斷和流程控制

德州撲克基本玩法：

1. **下盲注**：遊戲开始前，首兩名玩家必須分別下小盲注和大盲注
2. **發手牌**：每位玩家會先獲得兩張手牌，這兩張牌只有自己可以看到
3. **下注輪次**：第一輪下注开始，从第一个玩家开始，选择跟注、加注、弃牌或者过牌
4. **前三張公共牌**：接着發三張公共牌
5. **下注輪次**：第二輪下注开始，玩家下注方式和之前一樣
6. **第四張公牌**：下注结束后，發放第四張公共牌
7. **下注輪次**：第三輪下注开始，玩家下注方式和之前一樣
8. **第五張公共牌**：下注结束后，發放第五張公共牌，这是最后一張公共牌
9. **最终下注輪次**：第四輪下注开始，玩家可以进行最后一輪下注操作
10. **攤牌**：最后剩下的玩家将亮出手牌，与公共牌組成最好的五張牌組合以決定勝負
11. **決定勝負**：根据牌型大小和規則，確定獲勝者，贏得底池中的籌碼

1.程序界面

1.1 登錄畫面(默認畫面)



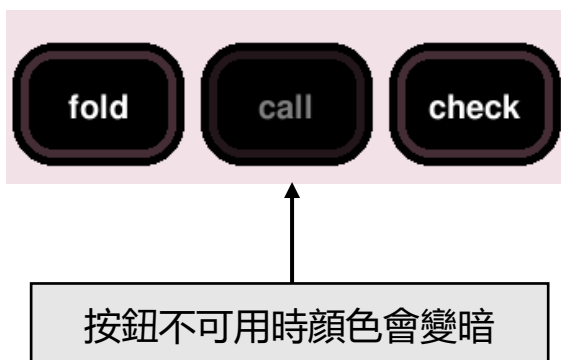
1.2 注冊畫面



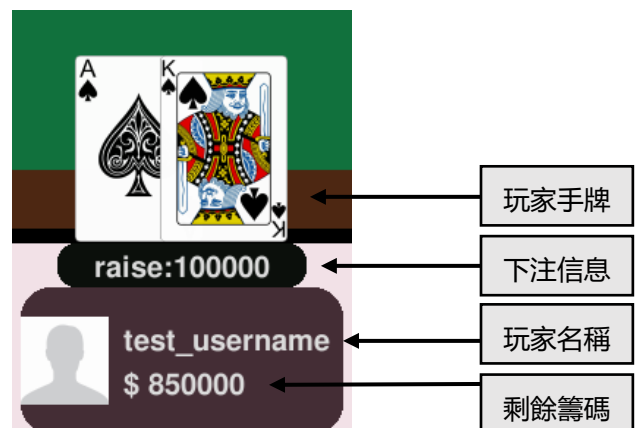
1.3 遊戲畫面



下注按鈕顯示：



用戶信息顯示：

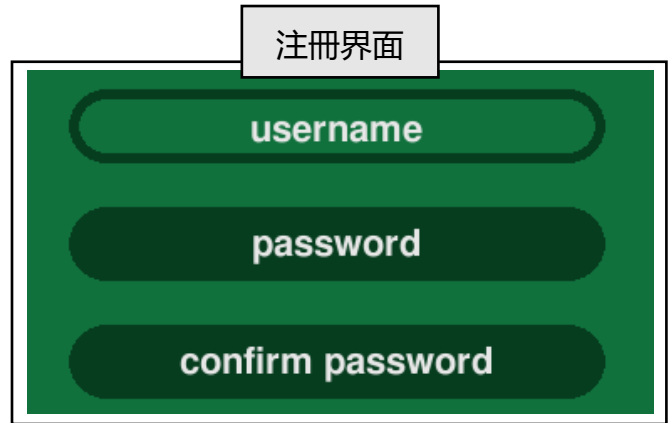


2.用戶數據的處理

2.1 用戶輸入的數據檢驗

註冊界面：

在註冊進程中，玩家需要為自己創建一個獨一無二的用戶名和長度必須為 8 位並包含數字和字母的密碼，遊戲中提示為(Password must be 8 characters long with numbers and letters)。



The diagram shows a registration interface with a title '註冊界面' (Registration Interface) in a grey box. Below it is a green rounded rectangle containing three dark green rounded input fields. The first field is labeled 'username', the second 'password', and the third 'confirm password'.

為確保用戶在(password)框內準確輸入新密碼，還需在其下方(confirm password)框內再輸入一次密碼，最終按下註冊按鈕后提交信息至後台驗證

後台驗證：

數據傳入後台后，將新用戶名與所有用戶名做比對，通過後則繼續校驗兩次密碼是否相同，再而用數據類型和長度檢查進行校驗。最終通過后会通知用戶註冊成功，遊戲中提示為(sign up successful)。登錄則直接與數據庫中的用戶做匹配，成功后便顯示登陸成功，遊戲中提示為(sign in successful)。

2.2 數據存儲設計

所有用戶數據：

文件名稱	Player.json
目的	此文件包含用戶的屬性。用於登錄系統
數據層級關係	"player_data" 數組包含了所有用戶對象，每個對象的數據以獨立的字典方式存儲

用戶屬性：

屬性名	解釋	例子
type	用戶的類型，只有整數 0 和 1，分別是電腦和真實用戶	0
username	用戶名，類型為字符串	Peter
Passord	登陸密碼，類型為字符串	1234abcd
Chip	用戶剩餘籌碼，類型為非負整數	100000

示例數據文件與格式：

```
{
  "player_data": [
    {
      "type": 0,
      "username": "Peter",
      "password": "832jce83912deddwa",
      "chip": 10000
    },
    {
      "type": 0,
      "username": "Amy",
      "password": "832jce83912deddwa",
      "chip": 0
    },
    {
      "type": 1,
      "username": "Tom",
      "password": "1234hello",
      "chip": 5000
    },
    {
      "type": 0,
      "username": "John",
      "password": "832jce83912deddwa",
      "chip": 50000000
    },
    {
      "type": 0,
      "username": "Alex",
      "password": "832jce83912deddwa",
      "chip": 5400000
    },
    {
      "type": 1,
      "username": "Sam",
      "password": "heyictsba2025",
      "chip": 1000000
    }
  ]
}
```


3.程序編寫

3.1 程式總預覽

程序文件夾結構：

- code
 - base.py
 - button.py
 - evalutor.py
 - holdem.py
 - input_box.py
 - keywords.py
 - login.py
 - main.pyw
 - player.py
 - rendering.py
 - setting.py
- data
 - player.json
- font
 - FreeSansBold.ttf
- image
 - card_back.png
 - User.jpg
 - cards
 - 10_1.png
 - 10_2.png
 - 10_3.png
 - 10_4.png
 - ...
 - chips
 - black.png
 - ...

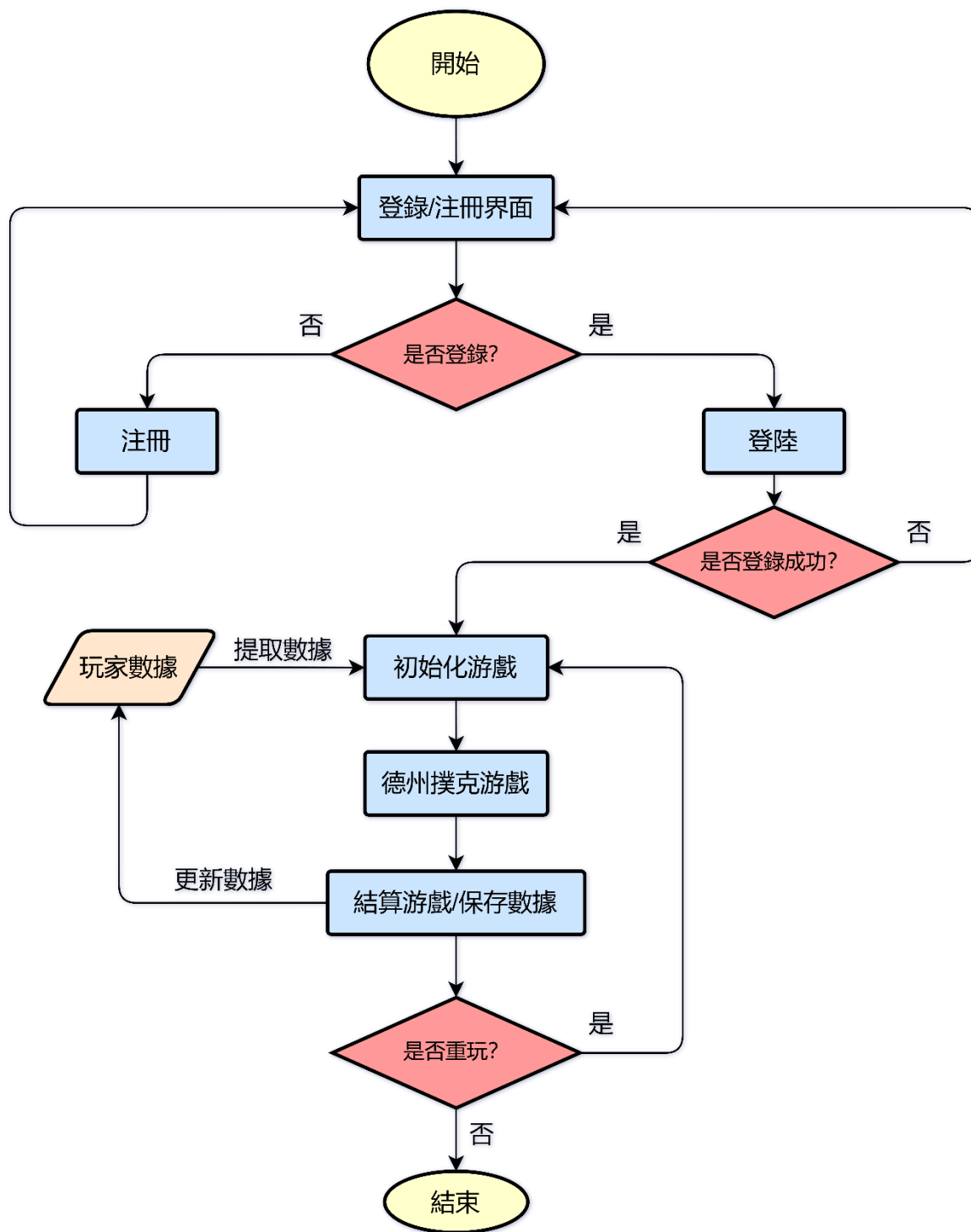
文件夾作用：

文件夾名稱	作用
code	程序原代碼
data	存儲用戶數據
Font	程序界面顯示的字體
image	程序內使用的圖片

code 文件夾：

文件名稱	作用
main.py	程序運行入口
setting.py	程序渲染配置文件
keywords.py	儲存程序中的關鍵字或字串
login.py	存儲用戶信息處理函數
base.py	存儲遊戲的基類(base class)
holdem.py	德州撲克模塊
evalutor.py	計算玩家手牌組合等級
player.py	儲存玩家操作類
rendering.py	圖像渲染模塊
button.py	存儲按鈕的類
input_box.py	存儲輸入框的類

程式運行結構圖：



3.2 程式基礎設定

運行環境：

1. 程式僅支持 ASCII 編碼中字符作輸入，不支持的字符將呈現「□」符號
2. 程式所需的解釋器(interpreter)版本至少為 python 3.10(應用了新版本語句 match... case ...)
3. 程式依賴內置模塊 random、os、json、itertools 和第三方模塊 pygame)
4. 安裝 pygame 需要在命令行(cmd)內輸入 pip install pygame (需要將 pip 添加至環境變量)

程式個性化設定：

1. 可在 setting.py 文件中更改程式中畫面的設定，如幀數、畫面長度、畫面寬度、顏色、位置等
2. 可在 keywords.py 文件中更改程式中顯示的文字內容，如的登陸提示、下注提示等

setting.py 文件示例：

```
# screen setting
SCREEN_HEIGHT= 750
SCREEN_WIDTH = 1380
SCREEN_COLOR = (242, 225, 231)

# FPS setting
FPS = 120

# title
TITLE_FONT_SIZE = 72
```

keywords.py 文件示例：

```
# button
ALL_IN = 'all in'
CHECK = 'check'
BET_RAISE = 'raise'
FOLD = 'fold'
CALL = 'call'
INCREASE = '+'
DECREASE = '-'

#login info
SIGN_IN_INFO = 'Please login'
SIGN_IN_SUCCESSE = 'sign in successful'
```

3.3 數據類型和數據結構

數據類型：

程式中使用的數據類型有布爾值(bool)、整數(int)、字符串(str)、浮點數(float)、列表(list)、元組(tuple)、字典(dict) 與類(class)

類型	作用	例子
布爾值	儲存二元狀態	<code>CHEATING_MODE = False</code>
整數	存儲遊戲基本整數數據	<code>FPS = 120</code>
字符串	存儲遊戲內提示和字典的鍵名	<code>SIGN_IN_INFO = 'Please login'</code>
浮點數	存儲進行精確計算后的值，如坐標變換后的值	<code>POKER_HEIGHT = 127.05</code>
列表	存儲整合后的數據	<code>buttonList = [b1,b2,b3,...]</code>
元組	存儲畫面設置，如坐標，RGB 顏色	<code>BUTTON_COLOR = (0,0,0)</code>
字典	儲存鍵值對數據，如圖片名所對應的圖片對象	<code>{file_name:image_surface,...}</code>
類	自定義數據，以創建遊戲規則	<code>class BaseGame:...</code>

數據結構：

程式使用的數據結構有二維數組(two-dimensional array)，和優先隊列(priority queue)，其餘則繼續使用列表(list)和元組(tuple)

類型	原因	例子
二維數組	有直觀的嵌套數據表示方式，方便提取數據	<code>HAND_POSITION = [[pos_1,pos_2],...]</code>
優先隊列	能夠直接獲取優先度最高的數據，e.g. 以玩家手牌等級作為優先度，來獲得做高優先玩家，即勝家	<code>winnersList = PriorityQueue()</code>

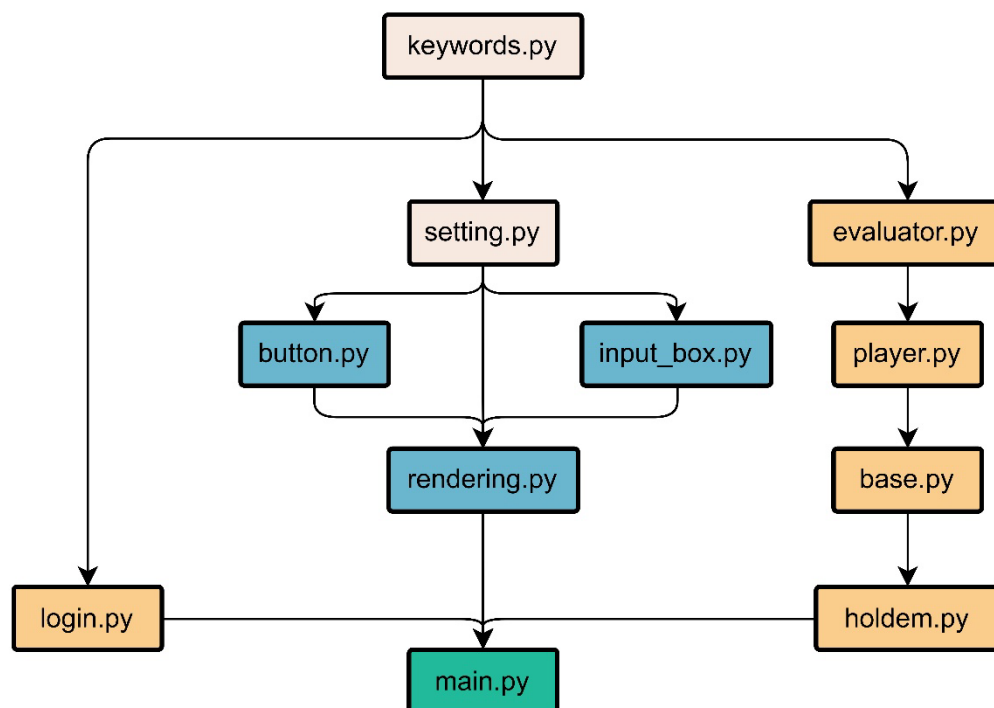
3.4 模塊化編寫

程式大致分爲四大模塊——配置文件、渲染組件、計算組件、主程序：

1. 配置文件：管理所有可配置變量
2. 渲染組件：接收計算組件傳入的數據以渲染給玩家
3. 計算組件：計算提交給程式數據，並返回信息，沒有任何畫面反饋
4. 主程序：將渲染組件和遊戲組件合并以運行遊戲

配置文件	渲染組件	計算組件	主程序
setting.py	rendering.py	login.py	main.py
keywords.py	button.py	base.py	
	input_box.py	holdem.py	
		player.py	
		evalutor.py	

模塊關係圖：



配置文件:

1. setting.py——特殊定義

定義	作用
<code>class QuitGame(Exception):...</code>	繼承錯誤基類(Exception), 調用此類將觸發異常, 安全退出游戲
<code>poker_id</code>	e.g(14_4)前部以 2-14 代表 2-A 的撲克點數,后部 1-4 代表花色
<code>imageList</code>	存儲所有卡牌的字典{卡牌 id:圖片對象,...}
<code>seat</code>	代表為玩家的座位索引, 範圍是 0-4, 以最左側為 0 到最右側 4

渲染組件:

1. Button.py——存儲 Button 類,內置 3 個方法(method),創建按鈕實例(instance) 以供用戶點擊按鈕

Button 類

方法	作用
<code>def adjust_value():...</code>	調整按鈕隱藏數值, 即可能自定義點擊按鈕后自增 1
<code>def check():...</code>	檢查按鈕是否被點擊
<code>def draw():...</code>	渲染按鈕至用戶界面

2. Input_box.py——存儲 Input 類,內置 3 種方法,創建輸入框實例以供用戶輸入信息

Input 類

方法	作用
<code>def content_init():...</code>	重置輸入框內容
<code>def check():...</code>	檢查輸入框是否被輸入
<code>def draw():...</code>	渲染輸入框至用戶界面

3. rendering.py——內置 2 個預渲染函數, 15 個渲染函數

預渲染函數(調用將返回圖形對象, 一般供渲染函數使用)

函數	作用
<code>def render_reminder():</code>	返回包含提示語的圖形對象
<code>def render_pot_info():</code>	返回獎金池圖形對象

渲染函數(調用即渲染在用戶界面)

函數	作用
<code>def draw_ranking():...</code>	在登陸/注冊界面渲染用戶籌碼排行榜
<code>def draw_introduction():...</code>	在登陸/注冊界面渲染入場動畫
<code>def draw_temp_reminder():...</code>	渲染短暫提示
<code>def draw_table():...</code>	渲染賭桌在遊戲界面
<code>def draw_players():...</code>	渲染玩家用戶名和剩餘籌碼在遊戲界面
<code>def draw_combo():...</code>	渲染玩家牌組名稱(e.g. One Pair, Flush)在遊戲界面
<code>def draw_pot():...</code>	渲染遊戲總獎金在遊戲界面
<code>def draw_betting():...</code>	渲染玩家最新的下注信息(e.g. call 3000, check)在遊戲界面
<code>def draw_winner():...</code>	渲染獲勝玩家用戶名和牌組名稱在遊戲界面
<code>def draw_card():</code>	渲染單張卡牌
<code>def draw_hand():...</code>	渲染單個玩家的手牌在遊戲界面
<code>def draw_community():...</code>	渲染公牌在遊戲界面
<code>def draw_consideration():...</code>	渲染進度條在遊戲界面，以模擬電腦思考時間
<code>def move_animation():...</code>	傳入渲染對象和起始和終點坐標，對象將按直線軌跡運動
<code>def interact():</code>	用戶與程式交互的接口，將渲染按鈕和輸入框

計算組件：

1. `login.py`——內置 7 個函數

函數	作用
<code>def load_players():...</code>	返回所有用戶的賬號數據
<code>def sign_in():...</code>	傳入賬號密碼以登錄，成功后將返回該賬號的數據，否則 False
<code>def sign_up():...</code>	傳入賬號密碼以創建賬號，成功后將返回 True，否則 False
<code>def get_testing_data():...</code>	返回測試賬號數據，其數據為臨時創建，不會作保存
<code>def get_bot():...</code>	返回機器人的賬號數據
<code>def save_game():...</code>	傳入新數據以更新用戶數據
<code>def calculate_ranking():...</code>	返回根據籌碼排名的用戶列表

2. base.py——存儲 BaseGame 類，內置 **5** 個方法，所有遊戲將繼承該類作基本處理

BaseGame 類

方法	作用
<code>def init_player():...</code>	初始化包含玩家對象的列表
<code>def save_game():...</code>	返回當前遊戲的全部玩家數據
<code>def get_players_info():...</code>	返回全部玩家的單一屬性，e.g. 按 seat 返回所有玩家的用戶名
<code>def playersList():...</code>	返回包含玩家對象的列表，未初始化時列表為空
<code>def game_loop():...</code>	遊戲回合的抽象方法

3. holdem.py——存儲 Holdem 類，內置 **6** 個方法，繼承 BaseGame 類

Holdem 類

方法	作用
<code>def check_game():...</code>	檢查遊戲初始化是否正確
<code>def game_loop():...</code>	調用將運行單個遊戲回合
<code>def betting_round():...</code>	單個回合中的下注細節運算
<code>def check_winner():...</code>	將勝家資訊儲存在 self.winnerList 列表中
<code>def deal_player():...</code>	發放手牌給玩家對象和二維數組 self.handList
<code>def deal_community():...</code>	發放公派給 self.communityCardsList 列表

4. player.py——存儲 Player 和 Bot 類，共 **3** 個方法，Bot 繼承 Player 類，供玩家進行操作

Player 類

方法	作用
<code>def decision():...</code>	以字典形式返回玩家對遊戲回合的操作
<code>def combination():...</code>	評估為玩家的牌組等級，並存儲在 self.combo 中

Bot 類

方法	作用
<code>def decision():...</code>	重寫了真人操作部分，以隨機數做操作代表機器人，返回值一樣

5. evaluator.py ——存儲 Card、LookupTable 和 Evaluator 類，共 **19** 個方法(細節見 3.6 算法設計)

Card 類

方法	作用
<code>def new():...</code>	傳入卡牌 id 以返回卡牌的唯一標識碼
<code>def int_to_str():...</code>	傳入卡牌的唯一標識碼以返回卡牌 id
<code>def get_rank_int():...</code>	傳入卡牌的唯一標識碼以返回 0 為始的卡牌等級
<code>def get_suit_int():...</code>	傳入卡牌的唯一標識碼以返回花色數值
<code>def get_bitrank_int():...</code>	傳入卡牌的唯一標識碼以返回卡牌等級
<code>def get_prime():...</code>	傳入卡牌的唯一標識碼以返回卡牌對應的質數
<code>def hand_to_binary():...</code>	傳入卡牌的唯一標識碼以返回二進制標識碼
<code>def prime_product_from_hand():...</code>	傳入卡牌列表以通過卡牌質數計算牌組數值
<code>def prime_product_from_rankbits():...</code>	傳入卡牌列表以通過卡牌等級計算牌組數值

LookupTable 類

方法	作用
<code>def flushes():...</code>	更新全為同花的查找表
<code>def straight_and_highcards():...</code>	更新非同花的查找表
<code>def multiples():...</code>	更新非同花的查找表
<code>def get_lexographically_next_bit_sequence():...</code>	按字典顺序计算下一个位排列

Evaluator 類

方法	作用
<code>def evaluate():...</code>	傳入牌組以返回牌組編號
<code>def _two():...</code>	計算兩張牌時的牌組編號
<code>def _five():...</code>	計算五張牌時的牌組編號
<code>def _six():...</code>	計算六張牌時的牌組編號
<code>def _seven():...</code>	計算七張牌時的牌組編號
<code>def get_rank_string(): ...</code>	傳入牌組編號以返回牌組等級名稱

主程序：

1. main.py——內置 3 個函數

函數	作用
<code>def login_page():</code>	登錄/注冊界面
<code>def holdem_page():</code>	德州撲克游戲界面
<code>def main():</code>	程式入口

3.5 程序內數據交互

實時渲染：

在 main.py 內，holdem.py 與 rendering.py，login.py 與 rendering.py 之間的函數或方法互相傳參，讓用戶界面的操作反饋到計算組件，計算後再返回數據讓渲染組件更新畫面

Login_page 函數示例

代碼	解釋
<code>draw_ranking(calculate_ranking(load_players()))</code>	<ol style="list-style-type: none">1. 將 <code>load_players()</code> 返回的玩家數據傳給 <code>calculate_ranking()</code>2. <code>calculate_ranking()</code> 將返回名次排序3. 將名次排序傳入 <code>draw_ranking()</code>，進而渲染到用戶界面
<pre>result = interact(...) if result == SIGN_IN: ... login = sign_in(result) if login: draw_temp_reminder() else:...</pre>	<ol style="list-style-type: none">1. <code>Interact()</code> 返回的用戶操作存入 <code>result</code>2. 判斷是否登錄3. 將用戶結果(輸入的賬號密碼)傳入 <code>sign_in()</code> 驗證4. 登錄成功將渲染成功提示

holdem_page 函數的數據交互示例:

holdem_page 函數中的用戶界面交互較為複雜，以高階函數(Higher-order function)為基礎編寫了遊戲主循環，原因在於單個下注循環中需要實時渲染玩家的下注操作，必須接收輸入類函數以做到同時計算和接收反饋，輸出類函數則是可選的(默認為無輸出)

代碼	解釋
<pre>game.game_loop(interact, draw_players, draw_hand, draw_betting, draw_consideration)</pre>	<p>遊戲主循環 game_loop()</p> <ol style="list-style-type: none">1. Interact (用戶操作反饋),2. draw_players(渲染玩家實時籌碼等信息)3. draw_hand(渲染玩家手牌, e.g.是否棄牌)4. draw_betting(渲染玩家最新的下注信息)5. draw_consideration(模擬電腦思考時間)

內部調用:

當渲染函數傳入 game_loop 后將繼續傳給有需要調用的方法，包括中的 Holdem 類的方法 betting_round 和 Player 類的方法 decision

Holdem.betting_round 方法示例(輸出):

代碼	解釋
<pre>if updateChipFunc: updateChipFunc(...) if updatePotFunc : updatePotFunc(...) if updateCardFunc and ...: updateCardFunc(...)</pre>	<ol style="list-style-type: none">1. 條件判斷是否傳入了輸出函數2. 輸出格式需參考文檔字符串(docString)3. 輸出函數可以以任何方式輸出，需按照輸出格式處理數據

Player.decision 方法示例(輸入):

代碼	解釋
<pre>result:dict = choiceFunc(...)</pre>	<ol style="list-style-type: none">1. 輸入函數指定了輸出類型為字典2. 輸入函數將被傳入一些限制條件以限制用戶某些操作3. 輸入函數可以以任何方式接收反饋，只需遵守返回值類型

3.6 算法編寫

算法簡介：

本程式除去基本德州撲克的回合算法，渲染圖像算法外，evaluator.py——存儲用以演算玩家手牌組合等級的算法，其算法概念為 Cactus Kev's Poker Hand Evaluator，實際代碼為第三方開源模塊 treys (可通過 pip 下載)，但在本程式中已經對此模塊中的代碼進行更改以迎合一些額外的對局規則

Cactus Kev's Poker Hand Evaluator 算法：

該算法將玩家的牌組等級分化成 1 至 7462 個編號，但顯然玩家能獲得的 5 張牌組合數為 $C(52,5)$ ，即 2,598,960 個組合，遠遠超過 7462。在這約 260 萬個組合中包含了花色的組合，例如梅花 A 和方塊 A，但他們的等級在德州撲克規則裏的等級是一樣的(不看花色)，所以只看如何平局以分勝負的組合可以是四人持有 Q 高同花順(四種花色)、四人持有 J 高同花順、四人持有 10 高同花順，雖然花色不同，但等級一樣導致最終平局。將一種同花順化為一個編號，從關注花色的 40 種同花順減少至共 10 種同花順 (A 至 10)

組合數表格

組合名	唯一的編號數量	分類后的編號數量
Straight Flush(同花順)	40	10
Four of a Kind(四條)	624	156
Full Houses(骷髏)	3744	156
Flush(同花)	5108	1277
Straight(順子)	10200	10
Three of a Kind(三條)	54912	858
Two Pair(兩對)	123552	858
One Pair(一對)	1098240	2860
High Card(高牌)	1302540	1277
總和	2598960	7462

各組合編號的枚舉：

同花順	同花	兩對
1: 4 人持有 A 高同花順	323: 4 人持有 AKQJ9 同花	2468: 144 人持有 AAKKQ
2: 4 人持有 K 高同花順	324: 4 人持有 AKQJ8 同花	2469: 144 人持有 AAKKJ
...
9: 4 人持有 6 高同花順	1598: 4 人持有 76432 張同花	3324: 144 人持有 33225
10: 4 人持有 5 高同花順	1599: 4 人持有 75432 張同花	3325: 144 人持有 33224
四條	順子	一對
11: 4 人持四張 A 和一張 K	1600: 1,020 人持有 A 高順子	3326: 384 人持有 AAKQJ
12: 4 人持四張 A 和一張 Q	1601: 1,020 人持有 K 高順子	3327: 384 人持有 AAKQT
...
165: 4 人持有四張 2 和一張 4	1608: 1,020 人持 6 高順子	6184: 384 人持有 22643
166: 4 人持有四張 2 和一張 3	1609: 1,020 人持 5 高順子	6185: 384 人持有 22543
骷髏	三條	高牌
167: 24 人持有三張 A 和兩張 K	1610: 64 人持有 AAKKQ	6186: 1,020 人持有 AKQJ9
168: 24 人持有三張 A 和兩張 Q	1611: 64 人持有 AAKKJ	6187: 1,020 人持有 AKQJ8
...
321: 24 人持有三張 2 和兩張 4	2466: 64 人持有 22253	7461: 1,020 人持有 76432
322: 24 人持有三張 2 和兩張 3	2467: 64 人持有 22243	7462: 1,020 人持有 75432

生成點數編號：

需要將一組牌，不管排序的情況下(A,2,3,4,5 或 2,4,A,5,3)換算成對應的組合編號運用了質數乘積的唯一性，算數基本定理(The fundamental theorem of arithmetic)，將每張卡牌點數都分配一個質數，這樣無論是什麼牌組都會獲得一個唯一的數值(類似 hash 值)，進而可以將該數值進行質因數分解以重新獲得到每張牌的點數

分配質數表

點數	2	3	4	5	6	7	8	9	10	J	Q	K	A
質數	2	3	5	7	11	13	17	19	23	29	31	37	41

唯一識別碼：

爲了讓花色也有識別碼，並和的點數合并，算法以 4 個字節的二進制數值創建了一套卡牌存儲方案

二進制數據(x 為禁用)

xxxAKQJT98765432	CDHS	rrrr	xxppppppp
------------------	------	------	-----------

- 1. 紅色格子：牌的等級,根据牌的花色將對應的位調為 1
- 2. 黃色格子：牌的花色(梅花=C,方塊=D,紅心=H ,黑桃=S)根据牌的花色將對應的位調為 1
- 3. 藍色格子：r = 以 0 為始的卡牌等級(點數 2=0,點數 3=1,點數 4=2,點數 5=3,...,點數 A=12)
- 4. 綠色格子：p = 等級的質數 (點數 2=2,點數 3=3,點數 K=37,...,點數 A=41)

卡牌編號示例

	<i>xxxAKQJT 98765432</i>	<i>CDHS</i>	<i>rrrr</i>	<i>xxppppppp</i>
方块 K	00001000 00000000	0100	1011	00100101
黑桃 5	00000000 00001000	0001	0011	00000111
梅花 J	00000010 00000000	1000	1001	00011101

計算組合：

算法將同花與同花順歸為一類先進行檢測，因爲只檢測點數編號會忽略同花的組合，檢測方法便是利用計算出的二進制卡牌數值進行位元運算(Bitwise operation)，將 c1 至 c5 標記為 5 張手牌，如果得出為 0 則代表沒有同花，反之非零則有同花：

$$c1 \& c2 \& c3 \& c4 \& c5 \& 0xF000$$

如有同花將繼續檢測是否為順子，繼續使用位元運算：

$$q = (c1 \mid c2 \mid c3 \mid c4 \mid c5) \gg 16$$

這樣，便有了最低為 31(0b0001 1111),最高為 7936(0b1 1110 0000 0000),程式中 LookupTable 類的 flushes 方法裏創建了長度為 10 的 straight_flushes 列表，儲存了全部同花順的 q 值，只要將手牌的 q 值計算出再與該列表作遍歷查找便可得出是否為同花順，整體時間複雜度僅為 O(n)

雖然有 `straight_flushes` 列表能直接進行查找，但對於其他同花或其他非同花組合需要創建兩個查找表(lookuptable)結構的字典，因此用循環創建了以 `q` 值為鍵，組合編號為值的 `flush_lookup` 和 `unsuited_lookup` 分別儲存有同花的組合和沒同花的組合，所以普通同花只需訪問查找表便能直接得到組合編號，例如 `flush_lookup[7808]` 會返回 323(*AKQJ9 同花的組合編號*)，從而無需使用列表查找

當確定了牌組為非同花后，便能直接使用點數編號繼續計算了，對於這些手牌只需將 5 張牌的質數相乘：

$$q = (c1 \& 0xFF) * (c2 \& 0xFF) * (c3 \& 0xFF) * (c4 \& 0xFF) * (c5 \& 0xFF)$$

得出 `q` 后便能直接訪問 `unsuited_lookup` 字典，獲得組合編號，時間複雜度為 $O(1)$ ，程式在犧牲空間的情況下擁有很快的執行效率

創建查找表的時間：

在創建 `Evaluator` 類的實例時會開始創建兩個查找表，雖然執行時間會較長(4.5-5ms)但這只會運行一次，兩個查找表便會一直在內存中直到程序關閉，只要直接用計算後的牌組數值去訪問查找表便能直接得到牌組編號，所以整體效率將比每獲得一張新牌就計算該牌組的等級快得多

使用該演算法：

在主程序運行時將會自動在 `evalutor.py` 內創建一個 `Evaluator` 的實例供使用，需要配和 `Card` 類以產生的單個卡牌的唯一識別碼

代碼示例：

代碼	解釋
<code>evaluator = Evaluator()</code>	創建實例
<code>hand = [Card.new('2_1'), Card.new('2_2')]</code>	通過卡牌 id 創建有為唯一標識碼的手牌列表
<code>result = evaluator.evaluate(hand, [])</code>	進行演算，將返回牌組編號

3.7 算法優化

舊算法：

在使用 Cactus Kev's Poker Hand Evaluator 前演算牌組的算法是每有新的牌組更新等都需要約 1ms 的時間進行演算，其中大部分函數的時間複雜度都為 $O(n^2)$ ，雖然在正常遊戲中只會處理長度 2 至 7 的列表，優化效果并不明顯，但如果在未來有新卡牌遊戲並且要處理更長的卡牌列表，這些算法所需的時間將會以平方的速度增加

檢測順子的函數(已棄用)示例

```
def straight(cards_rank_list):
    power = []
    ranks = list(set(cards_rank_list))
    length= 5 # detect 5 consecutive number, length = 5
    if {2,3,4,5,14}.issubset(ranks): # special combination A,2,3,4,5
        power.append([1,2,3,4,5])
    for i in range(len(ranks) - length + 1):
        tempList = ranks[i:i + length]
        if all((tempList[j] + 1 == tempList[j + 1]) for j in range(length - 1)):
            power.append(tempList)
    return max(power) if power else power
```

檢測勝家算法的優化：

在 Holdem 類中的 check_winner 方法中運用了優先隊列的數據結構來尋找勝家，訪問最高優先級數據時的時間複雜度僅為 $O(1)$ ，比起普通列表遍歷查找或二分查找的時間複雜度至少也要 $O(\log n)$ ，減少循環代碼次數的同時增加了代碼的可讀性

check_winner 方法示例

```
def check_winner(self):
    nonFoldList = PriorityQueue()
    # 獲得未棄牌的玩家信息
    for player in self.playersList:
        if not player.fold:
            nonFoldList.put((player.combo[0],player,player.combo[1]))
    # 獲勝玩家
    self.winnerList.append(nonFoldList.get())
```


4. 程序運行測試

4.1 組件單元測試

測試流程：

將會選擇一部分關鍵的函數或方法進行測試，將輸出結果和預期效果做比對查看是否通過測試

Holdem 類：

測試代碼	預期輸出	實際輸出
<pre>game = Holdme() game.deal_community(range(5)) print(game.communityCardsList)</pre>	包含 5 個卡牌 id 的列表	<pre>['3_4', '4_3', '6_2', '11_1', '6_1']</pre>

login.py 內的函數：

測試代碼	預期輸出	實際輸出
<pre>print(sign_up('rex', '1234abcd', '1234abcd'))</pre>	True	True
<pre>print(sign_in('rex', '1234abcd'))</pre>	該用戶的賬號信息 (除了密碼)	<pre>{'type': 1, 'username': 'rex', 'chip': 500000}</pre>

evaluator.py 內的方法：

測試代碼	預期輸出	實際輸出
<pre>evaluator = Evaluator() hand = [Card.new('10_1'), Card.new('11_1')] board = [Card.new('12_1'), Card.new('13_1'), Card.new('14_1')] rank = evaluator.evaluate(hand, board) print(evaluator.get_rank_string(rank))</pre>	Royal Flush	Royal Flush

4.2 登錄與註冊界面測試

測試流程：

將使用邊界案例(boundary case)進行測試，使用三個值，一個恰好在邊界上，另外兩個分別在邊界的兩側，將輸出結果和預期效果做比對查看是否通過測試

註冊界面測試：

- 將輸入用戶名，兩次相同的密碼
- 對用戶名無任何排列要求，但確保是唯一的
- 密碼最低要求必須為 8 位，並包含字母和數字
- 兩次密碼需確保一致

已知信息：

- 已註冊的用戶名有(*Peter, Amy, Tom, John, Alex, Sam*)

第一組(用戶名是否存在)

	輸入測試	預期輸出	實際輸出
用戶名	John	User exist or password invalid	User exist or password invalid
密碼	1234abcd		
密碼	1234abcd		
	輸入測試	預期輸出	實際輸出
用戶名	Texas	sign up successful	sign up successful
密碼	Texas1234		
密碼	Texas1234		

第二組(兩次密碼是否相同)

	輸入測試	預期輸出	實際輸出
用戶名	Oscar	User exist or password invalid	User exist or password invalid
密碼	Oscar1234		
密碼	Oscar123		

	輸入測試	預期輸出	實際輸出
用戶名	Oscar	sign up successful	sign up successful
密碼	Oscar1234		
密碼	Oscar1234		

第三組(密碼長度)

	輸入測試	預期輸出	實際輸出
用戶名	Jason	User exist or password invalid	User exist or password invalid
密碼	1234abc		
密碼	1234abc		
	輸入測試	預期輸出	實際輸出
用戶名	Jason	sign up successful	sign up successful
密碼	1234abcd		
密碼	1234abcd		
	輸入測試	預期輸出	實際輸出
用戶名	Jason	sign up successful	sign up successful
密碼	1234abcde		
密碼	1234abcde		

第四組(密碼是否包含字母)

	輸入測試	預期輸出	實際輸出
用戶名	Sunday	User exist or password invalid	User exist or password invalid
密碼	12345678		
密碼	12345678		
	輸入測試	預期輸出	實際輸出
用戶名	Sunday	sign up successful	sign up successful
密碼	12345678a		
密碼	12345678a		

第五組(密碼是否包含數字)

	輸入測試	預期輸出	實際輸出
用戶名	Felix	User exist or password invalid	User exist or password invalid
密碼	abcdefgh		
密碼	abcdefgh		
	輸入測試	預期輸出	實際輸出
用戶名	Felix	sign up successful	sign up successful
密碼	abcdefgh1		
密碼	abcdefgh1		

登錄界面測試:

- 將輸入用戶名和密碼

已知信息:

- 已注冊的用戶信息(用戶名: *rex* , 密碼: *1234abcd*)

第一組

	輸入測試	預期輸出	實際輸出
用戶名	rex	sign up successful	sign up successful
密碼	1234abcd		
密碼	1234abcd		
	輸入測試	預期輸出	實際輸出
用戶名	Rex	User exist or password invalid	User exist or password invalid
密碼	1234abcd		
密碼	1234abcd		
	輸入測試	預期輸出	實際輸出
用戶名	rex	User exist or password invalid	User exist or password invalid
密碼	1234abcde		
密碼	1234abcde		

4.3 遊戲界面測試

測試流程：

遊戲界面的輸入方法只有按鈕(渲染組件)，因此只需關注按鈕能否在某些條件下設置為無效，例如有玩家加注后，check(過牌)按鈕將會被設置為無效

按鈕關鍵字列表(keywords.py 內)：

- CHECK(過牌)
- BET_RAISE(加注)
- FOLD(棄牌)
- CALL(跟住)
- INCREASE(增加注碼)
- DECREASE(減少注碼)

遊戲界面按鈕測試

狀態	條件	預期渲染輸出	實際渲染輸出
玩家為盲注位	玩家只能下固定注碼	只有 call 按鈕有效	只有 call 按鈕有效
盲注回合但不是盲注位	玩家只能過牌	只有 check 按鈕有效	只有 check 按鈕有效
有其他玩家加注	玩家不能過牌	只有 check 按鈕無效	只有 check 按鈕無效
沒任何玩家加注	玩家不能跟住	只有 call 按鈕無效	只有 call 按鈕無效
玩家未獲得手牌	玩家不能棄牌	只有 fold 按鈕無效	只有 fold 按鈕無效
玩家只能全押(all in)	玩家不能加注或過牌	check, raise 按鈕無效	check, raise 按鈕無效

5.錯誤處理

5.1 主程序異常響應

處理異常：

程序在執行期間正常不會發生報錯現象，但發生概率也不為 0，所以編寫了一些緊急措施用作處理運行時的報錯，讓用戶能繼續使用程式，而不是突然關閉

主程序入口：

在 main 函數中編寫了 try 語句，用來放置可能會引發異常的的代碼，except 后寫了 QuitGame，用來檢測是否關閉游戲，而 Exception 則用作接收除 QuitGame 外的其他異常類(Exceptions)。如果觸發了報錯現象則重新調用 main 函數自身，回到登錄頁面，從而避免關閉程序

代碼

```
def main():
    try:
        login_page()
        holdem_page()
    except QuitGame:
        ...
        quit()
    except Exception:
        main()
```

總結

面向对象设计

面向对象编程的引入使得程序的各个组成部分能够更好地进行模块化。每个类负责不同的功能。例如，创建了 `Player` 类来管理玩家信息，`Holdem` 类来处理牌组的生成和洗牌。这种设计使得代码更易于理解和扩展，新的功能可以通过添加新类或方法来实现，而不需要对现有代码进行大幅修改。

登录系统

为了提升游戏的互动性和安全性，实现了一个用户登录系统。用户可以创建账户并登录，这样可以保存其游戏记录和统计数据。此系统使用简单的文件存储机制，确保用户数据的安全性和隐私。在用户登录时，系统会验证用户名和密码，并提供相应的反馈。

德州扑克规则实现

游戏的核心是德州扑克的规则实现。我编写了一系列函数来处理发牌、下注、比牌等操作。这些函数的设计遵循单一职责原则，每个函数只处理特定的任务，从而提高了代码的可读性和可测试性。

Cactus Kev's Poker Hand Evaluator

为了准确评估手牌，集成了 `Cactus Kev's Poker Hand Evaluator` 算法。这一算法能够高效地判断不同手牌的强度，通过使用这个算法能够在游戏中快速且准确地比较玩家的手牌，提高了游戏的性能。

错误处理机制

在开发过程中，特别是在处理用户输入和游戏逻辑时。我实现了全面的错误处理机制，确保在发生异常时，程序能够优雅地恢复，而不是崩溃。通过使用 `try` 和 `except` 块，我能够捕获常见的错误，如输入格式错误、运行时异常等。

结论

通过这个项目，我不仅提升了自己的 Python 编程能力，还深入理解了面向对象编程的原则和实践。德州扑克游戏的开发过程让我体会到设计模式的重要性，以及如何将复杂的逻辑分解为可管理的组件。未来，我计划继续优化游戏的功能，增加更多的互动元素和图形效果，为玩家提供更加丰富的游戏体验。这个项目不仅是一个有趣的实践，更是我学习和成长的一个重要里程碑。