利用 Tkinter 製作簡易題庫

2022 / 05 ~ 2022 / 07

Situation

- 我的朋友在準備考試時，想大量練習考古題，但考古題的排序很固定且後續複習時找出錯誤的題目相當麻煩，因此我決定利用程式幫助我朋友

Task

- 我的朋友並沒有程式基礎，因此用程式做出題庫後必須要用圖形化的介面才能夠讓他輕鬆的進行練習
- 考試中心所釋出的考古題檔案是 **pdf** ，沒有辦法直接拿來取用，必須將它進行轉換再使用

Action

- 圖形化介面
  - 製作題庫是功能導向
  - 對於 **Python** 初學者的我也能夠開發出來
  - 因此選擇介面較簡陋，但相對簡單的 **Python Tkinter** 來做為本次開發的套件

- 題庫
  - 不同的 **pdf** 轉換套件可能因為演算法的差異導致轉換出來的結果不同
  - 選擇能對目標檔案轉換出完整結果，並且我能夠對結果進行操作的 **pdfminer** 來做為本次開發的套件

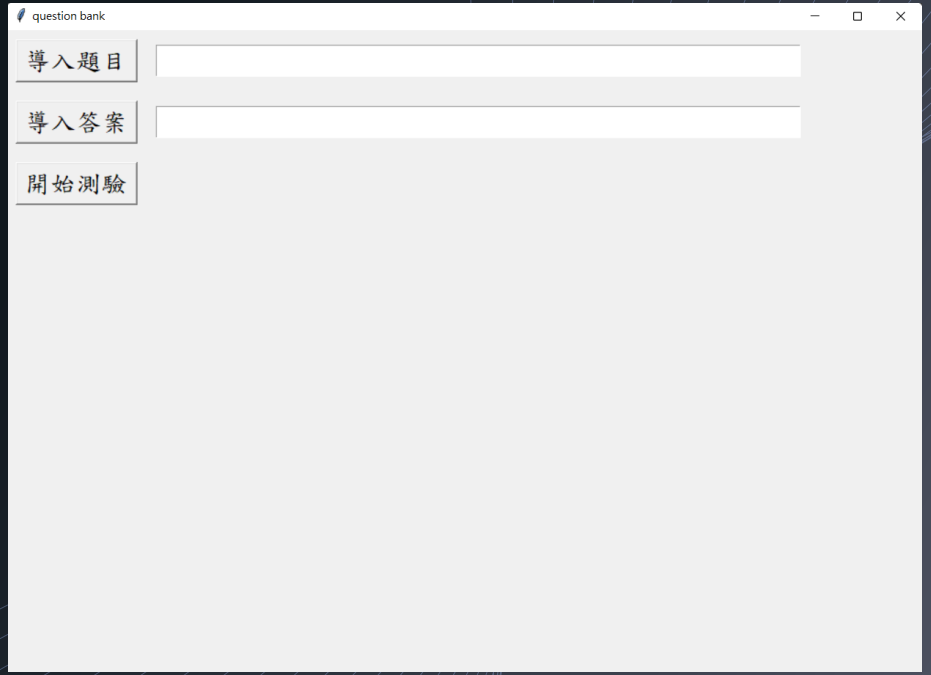
Action

- 定義介面
  - 利用 **tk.Frame** 進行介面的切換
  - 主頁面的建立
    - 以快速使用為目標，資料導入後即開始測驗，結束第一次測驗才進入主畫面
    - 還沒有資料庫的概念，沒有紀錄的功能，因此額外的功能都是在第一次測驗後才進行延伸

- 決定功能
  - 決定考古題範圍(考科)
  - 考古題練習
  - 錯題複習
    - 記錄錯誤題目並重複測驗
    - 將錯誤的題目下載下來以便考場複習

Action

- 進入 **GUI** 後能選擇題目和答案的範圍(題目和答案被分為兩個檔案)
- 選擇檔案後旁邊的文字輸入欄會自動填入檔案的絕對路徑
- 點擊開始測驗



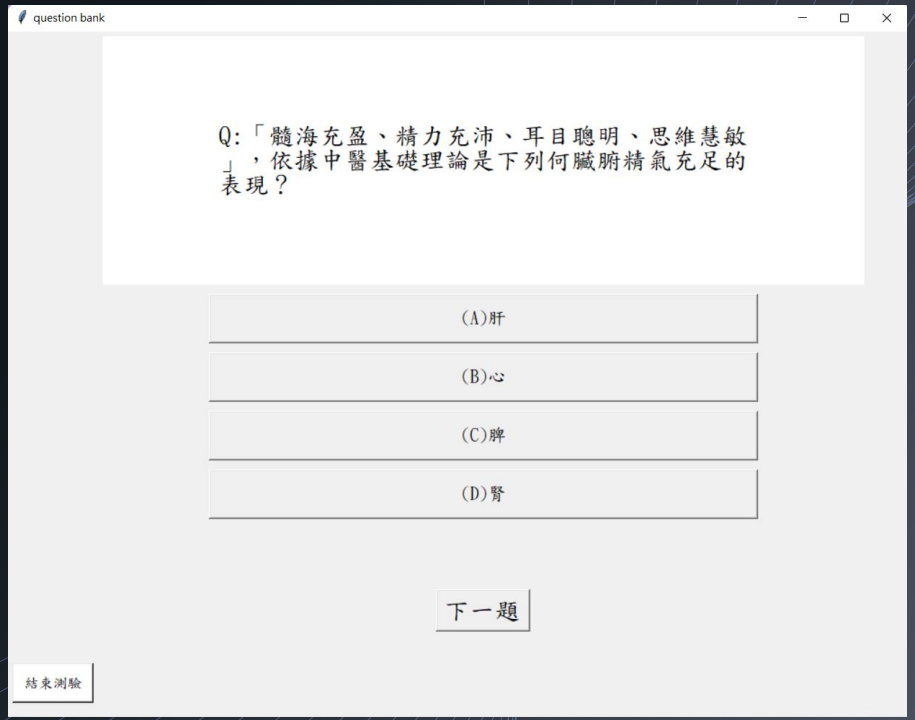
Action

- 處理 **pdf** 文字資料
  - 將上一步驟選擇的檔案進行處理
  - 利用 **pdfminer** 將 **pdf** 轉成 **txt**
- 將文字資料轉成 **Dataframe**
  - 將題目與答案分開成不同欄位
  - 容易進行資料處理
  - 呼叫方便

```python
# convert pdf to txt
def convert_pdf_to_txt(path):
    rsrcmgr = PDFResourceManager()
    retstr = StringIO()
    codec = "utf-8"
    laparams = LAParams()
    device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)
    fp = open(path, "rb")
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    password = ""
    maxpages = 0
    caching = True
    pagenos = set()
    for page in PDFPage.get_pages(
        fp, pagenos, maxpages=maxpages, password=password,
        caching=caching, check_extractable=True
    ):
        interpreter.process_page(page)
    text = retstr.getvalue()
    fp.close()
    device.close()
    retstr.close()
    return text
```

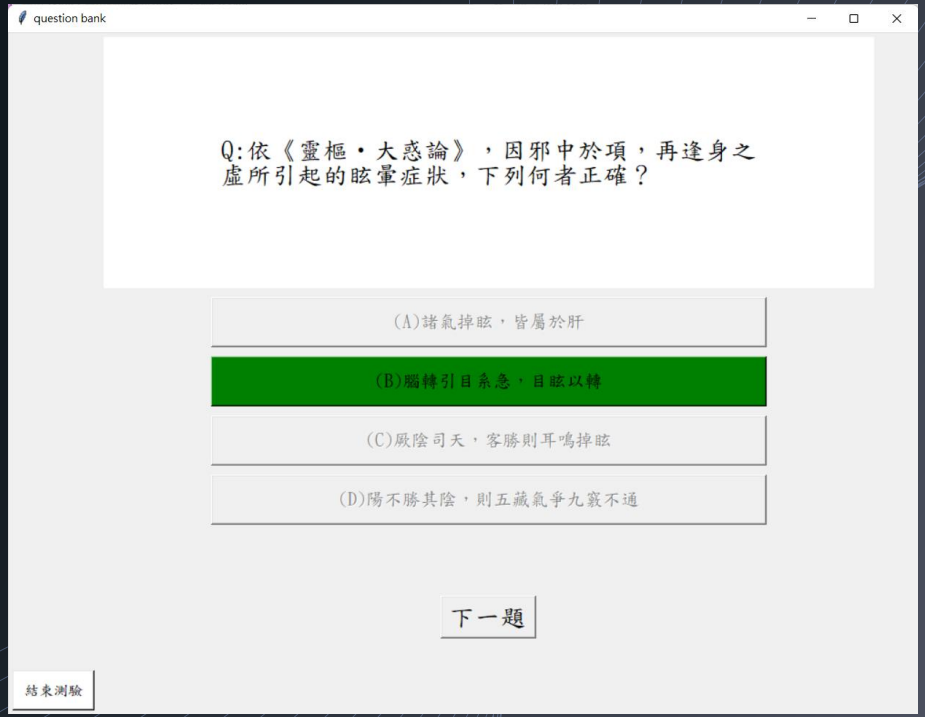
Action

- 開始後顯示題目畫面與選項
- 新增結束測驗以及下一題的按鈕



Action

- 答題正確會使選擇的選項背景變成綠色
- 此時所有選項按鈕會變成無法再點選

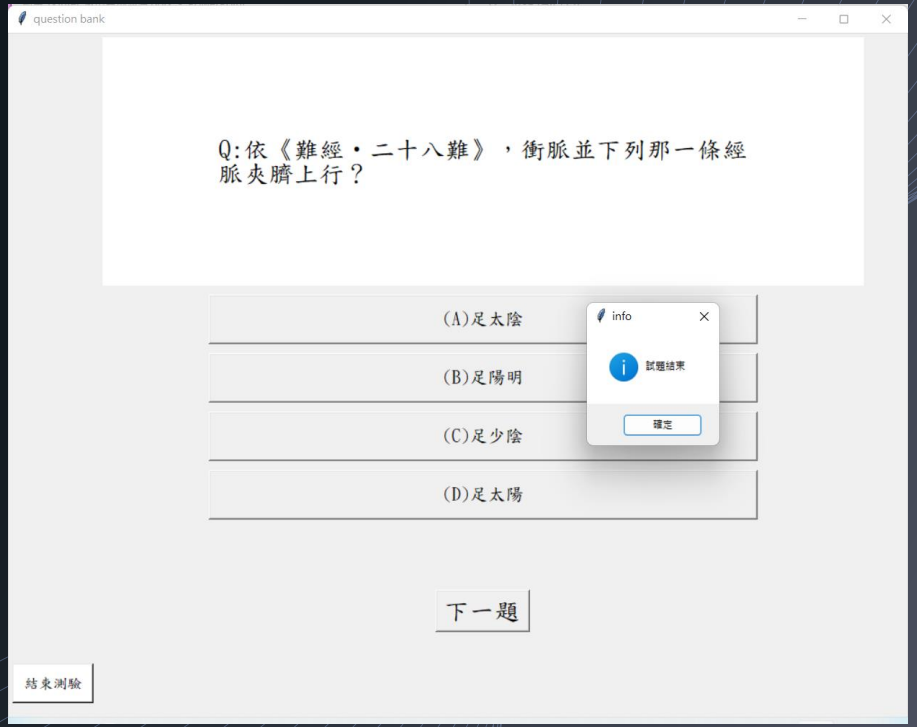


Action

- 答題錯誤會使選擇的選項背景變成紅色
- 此時所有選項按鈕會變成無法再點選
- 會顯示正確答案在所有選項下方

Action

- 所有題目作答完畢以後，再按下下一題會跳出訊息框提示試題結束
- 按下確定後會直接將畫面切換到主畫面



Action

- 按下結束測驗或試題結束後畫面會切換到主畫面
- 重新作答會讓所有題目隨機順序進行作答
- 錯題練習會記錄第一次做錯的題目並重新練習
- 在錯題練習仍然答錯的問題會被記錄，下載常錯題目會將這些題目的問題與選項寫入 **excel**

Result

- 能夠節省大量練習時間並能即時發現作答錯誤的題目
- 錯誤的題目能夠重複練習
- 常錯的題目能夠記錄下來，考前複習用
- 介面簡單，能夠快速上手

Self-Critism

- 可變性以及可擴充性差
  - 學習 **Python** 進階語法、演算法、資料結構
- 程式碼過於冗長
  - 學習 **OOP**
    - 把一些功能模組化**(ex.** 導入題目和答案**)**，可以簡化 **python** 檔案，在 **code review** 時易讀性增加
  - 重構程式碼
- 如果能結合資料庫功能，就不需要每次先做完一輪題目才能延伸出其他功能
  - **back-end development**
- 對於物件擺放的方式和配色可以多加考慮進去，對於使用者體驗會有幫助
  - 學習其他 **Python GUI** 套件，嘗試更多不同功能與風格
  - 學習 **UX / UI**

GitHub Link

- https://github.com/hncp025/question-bank

PPT Template Resource

- https://www.slidescarnival.com/mutius-free-presentation-template/9823