D1210799 王建葦 H03

5.5.7.1 函式練習題（一）：計算投手勝敗投

題目：

編寫一個名為 determine_pitcher_result(innings_pitched, earned_runs, total_runs, is_winning_team) 的 Python 函式。這個函式接收四個參數：

1. innings_pitched（浮點數）：投手投球局數。
2. earned_runs（整數）：投手自責分。
3. total_runs（整數）：該隊總得分。
4. is_winning_team（布林值）：一個布林值，表示該投手所屬的球隊是否贏得比賽。

函式應該根據以下規則判斷投手的勝敗投，並回傳一個字串：

1. 如果投手投滿 5 局（含）以上，且其所屬球隊贏得比賽，則回傳 "勝投"。
2. 如果投手投滿 5 局（含）以上，且其所屬球隊輸掉比賽，則回傳 "敗投"。
3. 如果投手投球未滿 5 局，無論勝敗，都回傳 "中繼"。

設計方法概述：

目標

設計一個簡單的程式，根據投手的投球局數、失分、是否勝隊等資訊，判定並輸出該場比賽中投手的結果（如：勝投、敗投、中繼）。

核心功能

- **判定規則**：
    - **勝投**：投手投滿至少 5 局，且球隊獲勝。
    - **敗投**：投手投滿至少 5 局，但球隊落敗。
    - **中繼**：投球不足 5 局，不論球隊勝敗。
- 其他狀況則預設為空字串。

函式設計

- determine_pitcher_result(innings_pitched, earned_runs, total_runs, is_winning_team) -> str
    - **輸入參數**：
        - innings_pitched (float)：投球局數。
        - earned_runs (int)：責任失分。
        - total_runs (int)：總失分（目前未用到，但保留可擴充）。
        - is_winning_team (bool)：球隊是否獲勝。
    - **邏輯**：
        - 若投滿 5 局且球隊獲勝，回傳 "勝投"。
        - 若投滿 5 局但球隊落敗，回傳 "敗投"。
        - 若未投滿 5 局，回傳 "中繼"。
        - 其他狀況回傳空字串（理論上不會進入這個情況）。
- main() -> int
    - 呼叫 determine_pitcher_result 測試不同情境。
    - 使用 print() 將結果輸出到螢幕。

- 最後回傳 0 作為程式正常結束的訊號。

程式流程

1. 主程式 main() 執行。
2. 依序測試不同投手數據組合。
3. 每次呼叫 determine_pitcher_result 判定並輸出結果。

擴充性

- earned_runs 和 total_runs 目前未參與判定，日後可加入更細緻的判斷（如資格敗投、中繼失敗等情境）。
- 可進一步考慮「救援成功」、「救援失敗」等情況，讓系統更完整。

程式、執行畫面：

```python
def determine_pitcher_result ( innings_pitched: float ,earned_runs: int ,total_runs: int ,is_winning_team: bool ) -> str:
    result = ""

    if ( innings_pitched >= 5.0 and is_winning_team ):
        result = "勝投"
    elif ( innings_pitched >= 5.0 and not is_winning_team ):
        result = "敗投"
    elif (innings_pitched < 5.0):
        result = "中繼"
    else :
        result = ""

    return result

def main () -> int:
    print(determine_pitcher_result(6.0 ,2 ,5 ,True))   # 預期輸出: 勝投
    print(determine_pitcher_result(7.0 ,1 ,3 ,False))  # 預期輸出: 敗投
    print(determine_pitcher_result(4.2 ,0 ,2 ,True))   # 預期輸出: 中繼
    print(determine_pitcher_result(3.0 ,3 ,1 ,False))  # 預期輸出: 中繼

    return 0

if ( __name__ == "__main__" ) :
    main()
```

```
PS E:\webDesignClass> & C:/Users/Albert/AppData/Local/Programs/Python/Python313/python.exe e:/webDesignClass/H03/src/5.5.7.1.py
勝投
敗投
中繼
中繼
```

參考資料與使用工具及比例：

ChatGPT: 5% → 設計方法概述內容修飾

5.5.7.2 函式練習題（二）：格式化球員打擊數據

題目：

編寫一個名為 format_batting_stats(name, at_bats, hits, home_runs) 的 Python 函式。這個函式接收四個參數：

1. name（字串）：球員的姓名。
2. at_bats（整數）：球員的打數。
3. hits（整數）：球員的安打數。
4. home_runs（整數）：球員的全壘打數。

函式應該計算球員的打擊率（安打數除以打數，保留三位小數），並回傳一個格式化的字串，包含球員的姓名、打數、安打數、全壘打數和打擊率，格式如下：

[球員姓名] 的打擊數據：

打數: [打數]

安打: [安打]

全壘打: [全壘打]

打擊率: [打擊率]

設計方法概述：

目標

設計一個簡單的函式，計算打者的打擊率，並將打者的基本打擊數據（打數、安打數、全壘打數、打擊率）以格式化的字串方式輸出。

核心功能

- 計算打擊率：
  - 打擊率 = 安打數 / 打數。
  - 小數點後保留三位數顯示。
- 格式化輸出：
  - 按照指定格式，逐行輸出打數、安打數、全壘打數、打擊率。

函式設計

format_batting_stats(name, at_bats, hits, home_runs) -> str

- 輸入參數：
  - name (str)：球員姓名。
  - at_bats (int)：打數（出場打擊的次數）。
  - hits (int)：安打數。
  - home_runs (int)：全壘打數。
- 邏輯：
  - 計算打擊率 (hits / at_bats)。
  - 組合成格式化的多行文字字串。
  - 回傳格式化好的字串。

程式流程

1. 呼叫 format_batting_stats，傳入球員的打數、安打數、全壘打數。
2. 函式內計算打擊率。
3. 將各項數據組成整齊的文字。
4. 回傳並輸出結果。

測試案例

- 測試資料：
  - 姓名：陳金鋒
  - 打數：450
  - 安打：150
  - 全壘打：30
- 預期輸出格式：
  - 陳金鋒 的打擊數據：
  - 打數: 450
  - 安打: 150
  - 全壘打: 30
  - 打擊率: 0.333

程式、執行畫面：

```python
def format_batting_stats ( name: str ,at_bats: int , hits: int, home_runs: int ) -> str:
    batting_average = 0.0

    # 在這裡計算打擊率
    batting_average = hits / at_bats

    # 在這裡格式化字串
    formatted_string = f"{name} 的打擊數據：\n"
    formatted_string += f"打數: {at_bats}\n"
    formatted_string += f"安打: {hits}\n"
    formatted_string += f"全壘打: {home_runs}\n"
    formatted_string += f"打擊率: {batting_average:.3f}\n"

    return formatted_string

# 測試你的函式
stats = format_batting_stats ( "陳金鋒" ,450 ,150 ,30 )
print ( stats )

# 預期輸出 (打擊率可能因計算精度略有不同):
# 陳金鋒 的打擊數據：
# 打數: 450
# 安打: 150
# 全壘打: 30
# 打擊率: 0.333
```

```
PS E:\webDesignClass> & C:/Users/Albert/AppData/Local/Programs/Python/Python313/python.exe e:/webDesignClass/H03/src/5.5.7.2.py
陳金鋒 的打擊數據：
打數: 450
安打: 150
全壘打: 30
打擊率: 0.333
```

參考資料與使用工具及比例：

ChatGPT: 5% → 設計方法概述內容修飾

5.6.8.1 棒球員

題目：

請設計一個名為 BaseballPlayer 的類別，用於表示一位棒球員。這個類別應該包含以下屬性：

- name（字串）：球員的姓名。
- number（整數）：球員的背號。
- position（字串）：球員守備位置（例如："投手", "捕手", "一壘手"等）。
- batting_average（浮點數）：球員的打擊率（初始值為 0.0）。
- home_runs（整數）：球員的全壘打數（初始值為 0）。

這個類別應該包含以下方法：

- __init__(self, name, number, position)：建構子，用於初始化球員的姓名、背號和守備位置。打擊率和全壘打數應初始化為預設值。
- display_info(self)：顯示球員的所有資訊，包括姓名、背號、守備位置、打擊率和全壘打數。
- increase_batting_record(self, hits, at_bats)：更新球員的打擊紀錄。這個方法接收本次打擊的安打數 (hits) 和打數 (at_bats)，並根據這些資訊更新球員的 batting_average。請注意，打擊率的計算方式是總安打數除以總打數。
- hit_home_run(self)：當球員擊出全壘打時呼叫此方法，將球員的 home_runs 增加 1。

操作要求：

1. 創建至少三位 BaseballPlayer 物件，並賦予他們不同的姓名、背號和守備位置。
2. 針對其中一位球員，呼叫 display_info() 方法顯示其初始資訊。
3. 讓其中一位球員進行兩次打擊紀錄更新：
    - 第一次打擊：2 安打，3 打數。
    - 第二次打擊：1 安打，4 打數。

    每次更新後，都呼叫 display_info() 方法查看該球員的最新資訊。
4. 讓其中一位球員擊出一支全壘打，並呼叫 display_info() 方法查看其全壘打數是否已更新。

進階挑戰：

- 為 BaseballPlayer 類別添加一個 __str__(self) 方法，使其在使用 print() 函數列印球員物件時，能以更友好的格式顯示球員資訊。
- 創建一個球隊（Team）類別，該類別可以儲存多個 BaseballPlayer 物件，並提供方法來新增球員、顯示球隊所有球員資訊，以及計算球隊的平均打擊率。

設計方法概述：

目標

建立一套基礎的棒球球員與球隊管理程式，能夠紀錄個別球員的打擊表現、全壘打數，以及統整球隊的成績統計（如球隊總打擊率）。

系統設計概覽

- **球員類別（BaseballPlayer）：**
  - 記錄個別球員的基本資料與打擊成績。
  - 支援更新打擊紀錄、擊出全壘打的行為。
  - 可以顯示球員資訊，或以自訂格式列印球員摘要。
- **球隊類別（Team）：**
  - 管理一群球員成員。
  - 提供新增球員、列出所有球員、計算球隊總打擊率等功能。
- **主程式流程（main()）：**
  - 建立球員 → 建立球隊 → 加入球員 → 操作球員記錄 → 輸出結果。

類別設計詳述

BaseballPlayer 類

- **屬性：**
  - name：球員姓名
  - number：球員背號
  - position：守備位置（如：內野手 IF、外野手 OF）
  - batting_average：打擊率（自動計算）
  - home_runs：全壘打數
  - hits：安打數
  - at_bats：打數

- **方法**：
  - display_info()：列印球員詳細資料。
  - increase_batting_record(hits, at_bats)：增加安打數與打數，並更新打擊率。
  - hit_home_run()：全壘打數加 1。
  - __str__()：定義球員物件的簡潔列印格式。

Team 類

- **屬性**：
  - name：球隊名稱
  - players：球員列表
- **方法**：
  - add_player(player)：將球員加入球隊。
  - display_all_players()：列印所有球員的資訊。
  - calculate_team_batting_average()：計算整支球隊的平均打擊率。

程式運行流程（main()）

1. **建立球員物件**（player1, player2, player3）。
2. **建立球隊物件**（team）。
3. **將球員加入球隊**。
4. **顯示單一球員的初始資訊**。
5. **模擬球員打擊與全壘打表現**，更新打擊紀錄。
6. **列印全隊所有球員的資訊**。

7. 計算並顯示球隊總打擊率。

程式、執行畫面：

```python
class BaseballPlayer() :
    def __init__ ( self ,name: str ,number: int ,position: str ,batting_average: float = 0.0 ,home_runs: int = 0 ,hits: int = 0 ,at_bats: int = 0 ) -> None :
        self.name = name
        self.number = number
        self.position = position
        self.batting_average = batting_average
        self.home_runs = home_runs
        self.hits = hits
        self.at_bats = at_bats

    def display_info ( self ) -> None :
        print ( f"球員姓名: {self.name}" )
        print ( f"球員號碼: {self.number}" )
        print ( f"球員位置: {self.position}" )
        print ( f"打擊率: {self.batting_average:.3f}" )
        print ( f"全壘打數: {self.home_runs}" )

    def increase_batting_record ( self ,hits: int ,at_bats: int ) -> None:
        self.hits += hits
        self.at_bats += at_bats

        self.batting_average = self.hits / self.at_bats

    def hit_home_run ( self ) -> None:
        self.home_runs += 1

    def __str__ ( self ) -> str:
        return (f"球員姓名: {self.name:^20s}"
                f" 球員號碼: {self.number:^3d}"
                f" 球員位置: {self.position:^6s}"
                f" 打擊率: {self.batting_average:^.3f}"
                f" 全壘打數: {self.home_runs:^3d}"
                f" 安打數: {self.hits:^3d}"
                f" 打數: {self.at_bats:^3d}")
```

```python
class Team:
    def __init__ ( self ,name: str ) -> None:
        self.name = name
        self.players = []

    def add_player ( self ,player: BaseballPlayer ) -> None:
        self.players.append ( player )

    def display_all_players ( self ) -> None:
        print ( f"球隊名稱: {self.name}" )
        for player in self.players :
            print ( player )

    def calculate_team_batting_average ( self ) -> float:
        total_hits = sum( player.hits for player in self.players )
        total_at_bats = sum( player.at_bats for player in self.players )

        if total_at_bats == 0:
            return 0.0

        return total_hits / total_at_bats
```

```python
def main () -> int:
    player1 = BaseballPlayer("CHANG Cheng-Yu", 9, "IF")
    player2 = BaseballPlayer("CHEN Chieh-Hsien", 24, "OF")
    player3 = BaseballPlayer("CHIANG Kun-Yu", 90, "IF")

    # 建立球隊
    team = Team("TPE")

    # 加入球員
    team.add_player(player1)
    team.add_player(player2)
    team.add_player(player3)

    # 顯示其中一位球員初始資料
    print("\n--- 初始資訊 ---")
    player1.display_info()

    # 第一次打擊：2 安打，3 打數
    print("\n--- 第一次打擊紀錄更新 ---")
    player1.increase_batting_record(2, 3)
    player1.display_info()

    # 第二次打擊：1 安打，4 打數
    print("\n--- 第二次打擊紀錄更新 ---")
    player1.increase_batting_record(1, 4)
    player1.display_info()

    # 擊出一支全壘打
    print("\n--- 擊出全壘打 ---")
    player1.hit_home_run()
    player1.display_info()

    # 顯示全隊球員資訊
    print("\n--- 全隊球員資訊 ---")
    team.display_all_players()

    # 計算並顯示全隊平均打擊率
    print("\n--- 球隊平均打擊率 ---")
    print(f"球隊平均打擊率: {team.calculate_team_batting_average():.3f}")

    return 0

if ( __name__ == "__main__" ) :
    main()
```

參考資料與使用工具及比例：

ChatGPT: 5% → 設計方法概述內容修飾

ChatGPT: 5% → 加分題諮詢