CS50 Python 程式設計導論：Week 2 迴圈 (Loops)

Week 2 的課程核心在於學習如何利用程式迴圈來重複執行任務，並結合條件判斷與不同的資料結構來解決問題。

一、 Week 2 核心概念

本週課程涵蓋了以下重要的程式設計概念和技巧：

- **迴圈 (Loops)**
    - **`while` 迴圈**：根據「Coke Machine」影片中的說明，`while` 迴圈會在一組語句被執行，只要其條件為真 (true) [PROBLEM SET 2: COKE MACHINE | SOLUTION]。一旦條件變為假 (false)，迴圈便會停止，程式會跳出 `while` 區塊 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。例如，在計算販賣機找零時，只要 `amount_due`（應付金額）大於零，`while` 迴圈就會持續運行 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
    - **`for` 迴圈**：用於迭代字串中的每個字元，或字典中的鍵 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。在「Camel Case」問題中，`for` 迴圈被用來遍歷使用者輸入字串的每個字母，以檢查其大小寫 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。在字典的 `for` 迴圈中，預設回傳的是字典的鍵 (keys)，但也可以透過鍵來存取對應的值 (values) [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
    - **`break` 關鍵字**：根據「Vanity Plates」影片中的解釋，`break` 關鍵字可以在 `for` 或 `while` 迴圈中，無論迴圈條件是否已變為假，都可以提前跳出迴圈 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
    - **`continue` 關鍵字**：CS50 Python Week 2 的課程資料提到了 `continue` 關鍵字 [Week 2 Loops - CS50's Introduction to Programming with Python]。它用於跳過當前迭代的剩餘部分，並繼續下一個迭代（此功能在提供的問題解決方案中未直接詳細展示，但作為核心概念被提及）。
- **條件語句 (Conditional Statements)**
    - **`if` 和 `else` 語句**：Python 的 `if` 和 `else` 語句幫助程式設計師控制程式的流程。它們會評估一個表達式是真 (true) 還是假 (false)。如果條件為真，`if` 語句的區塊會執行；否則，`else` 語句

的區塊會執行 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。
    - **`elif` 語句**：在前面的條件不滿足時，`elif` 語句用於檢查另一個條件 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。可以根據需要使用多個 `elif` 語句 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。
- **資料結構 (Data Structures)**
    - **字典 (Dictionaries)**：根據「Nutrition Facts」影片的說明，字典用於儲存鍵值對 (key-value pairs) 的資料。字典是**有序** (ordered)、**可變** (changeable) 且**不允許重複** (do not allow duplicates) 的集合 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - **鍵 (Key)**：冒號左側的部分 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - **值 (Value)**：冒號右側的部分 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 可以透過鍵來存取對應的值 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
    - **列表 (Lists)**：根據 CS50 Python Week 2 的課程資料，列表是本週討論的數據結構之一 [Week 2 Loops - CS50's Introduction to Programming with Python]。在「Coke Machine」影片中，列表被用來儲存有效的硬幣面額 (25, 10, 5)，並配合 `in` 運算子檢查使用者輸入的硬幣是否有效 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
    - **元組 (Tuples)**：根據 CS50 Python Week 2 的課程資料，元組也是本週討論的數據結構之一 [Week 2 Loops - CS50's Introduction to Programming with Python]。
- **內建函數與方法 (Built-in Functions & Methods)**
    - **`input()` 函數**：根據多個問題的解決方案影片 (如「Camel Case」、「Coke Machine」、「Just Setting Up My TWTTR」、

「Nutrition Facts」、「Vanity Plates」)，input() 函數用於向使用者提問，並將使用者輸入的答案儲存在變數中 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。需要注意的是，input() 函數總是回傳一個字串 (string) [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
    - **print() 函數**：
        - 預設情況下，print() 會在輸出後換行 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 可以使用 end=" " 參數來避免換行，使輸出保持在同一行 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 單獨使用 print() 可以在輸出後產生新行 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
    - **len() 函數**：根據「Vanity Plates」影片的說明，len() 函數可以回傳字串的長度 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
    - **int() 函數**：在「Coke Machine」影片中，int() 函數被用來將使用者輸入的字串（硬幣金額）轉換為整數，以便進行數值運算 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
    - **abs() 函數**：根據「Coke Machine」影片的說明，abs() 函數回傳指定數字的絕對值，常用於計算找零金額，確保結果為正數 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
- **字串方法 (String Methods)**
    - **.isupper() 方法**：根據「Camel Case」影片中的說明，如果所有字元都是大寫，則 .isupper() 方法會回傳 True；它會忽略數字、符號和空格，只檢查字母字元 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
    - **.lower() 方法**：根據「Camel Case」、「Just Setting Up My TWTTR」和「Nutrition Facts」影片的說明，.lower() 方法用於將字串轉換為小寫 [PROBLEM SET 2: CAMEL CASE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。這在需要進行不區分大小寫的比較時非常有用。

    - **`.isalpha()` 方法**：根據「Vanity Plates」影片中的說明，如果所有字元都是字母，則 `.isalpha()` 方法會回傳 `True`；它會忽略空格、問號等非字母字元 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
- **成員運算子 (Membership Operators)**
    - **`in` 和 `not in`**：根據多個問題的解決方案影片 (如「Coke Machine」、「Just Setting Up My TWTTR」、「Nutrition Facts」、「Vanity Plates」)，`in` 和 `not in` 運算子是成員運算子 (membership operators)，它們評估一個參數是否是另一個參數的成員 [PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。`in` 運算子如果找到要查找的內容，則回傳 `True`；否則回傳 `False` [PROBLEM SET 2: COKE MACHINE | SOLUTION, PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION, PROBLEM SET 2: NUTRITION FACTS | SOLUTION, PROBLEM SET 2: VANITY PLATES | SOLUTION]。
- **索引 (Indexes)**：在「Vanity Plates」影片中，提到可以使用索引來存取字串中特定位置的字元，例如 `s` 可以獲取第一個字元 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
- **`if __name__ == "__main__":` 結構**：根據「Vanity Plates」影片的解釋，在 Python 腳本中，此條件語句用於確保只有當腳本作為主程式執行時，特定的程式碼（通常是 `main()` 函數）才會被執行 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。

二、 Problem Set 2 解決方案概述

Problem Set 2 包含多個編程問題，每個問題都應用了本週學到的概念。以下是各問題的解決思路和概念應用：

1. **Camel Case（駝峰式大小寫）**
    - **目標**：根據「Camel Case」影片的說明，這個問題要求將使用者輸入的駝峰式大小寫 (camelCase) 變數名稱轉換為蛇形大小寫 (snake_case)，即用底線 `_` 分隔單詞，且所有字母小寫 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
    - **概念應用**：
        - 透過 `input()` 函數獲取使用者輸入的駝峰式字串 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。

        - 使用 **for 迴圈** 迭代輸入字串的每個字元 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
        - 在迴圈內部，使用 **if 條件** 和 **.isupper() 方法** 檢查當前字元是否為大寫 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
        - 如果字元為大寫（表示這是一個新單詞的開頭），則在輸出時在其前面列印下劃線 _，並將該字元轉換為小寫，使用 **.lower() 方法** [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
        - 在每次 print() 輸出字元或下劃線時，使用 end="" 參數來避免換行，使所有輸出保持在同一行 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
        - 最後，在迴圈結束後，單獨使用一個空的 print() 函數來產生新行，以正確結束輸出 [PROBLEM SET 2: CAMEL CASE | SOLUTION]。
2. **Coke Machine（可樂販賣機）**
    - **目標**：根據「Coke Machine」影片的說明，模擬一台可樂販賣機，可樂價格為 50 美分。程式需要接受使用者一次投入一枚硬幣，每次投入後告知使用者還需支付的金額，並在達到或超過 50 美分後計算應找零的金額 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。販賣機只接受 25、10、5 美分的硬幣 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
    - **概念應用**：
        - 初始化 amount_due（應付金額）變數為 50 美分 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
        - 使用 **while 迴圈** 持續詢問使用者投入硬幣，直到 amount_due 小於等於零（即已付清或多付） [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
        - 在每次迴圈開始時，列印當前的 amount_due [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
        - 使用 input() 函數獲取使用者輸入的硬幣金額，並用 **int() 函數** 將其轉換為整數 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
        - 使用 **if 條件** 和 **in 運算子** 檢查投入的硬幣是否為有效面額 (25, 10, 或 5 美分)。這比使用多個 or 條件更簡潔 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
        - 如果硬幣有效，則從 amount_due 中減去硬幣金額 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。

        - 迴圈結束後，使用 **abs() 函數** 計算應找零的金額。這是因為如果使用者多投了硬幣，amount_due 可能會變成負值，abs() 函數能確保找零金額為正數 [PROBLEM SET 2: COKE MACHINE | SOLUTION]。
3. **Just Setting Up My TWTTR（只是在設定我的 Twitter）**
    - **目標**：根據「Just Setting Up My TWTTR」影片的說明，這個問題要求從使用者輸入的字串中移除所有的元音 (a, e, i, o, u)，無論大小寫 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
    - **概念應用**：
        - 透過 input() 函數獲取使用者輸入的字串 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 初始化輸出字串或直接使用 print() 配合 end=" " 參數 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 使用 **for 迴圈** 迭代輸入字串的每個字元 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 在檢查每個字元時，先將其轉換為小寫，使用 **.lower() 方法**。這樣可以實現**不區分大小寫**的元音判斷 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 使用 **if not 條件** 和 **in 運算子** 檢查字元是否**不在**預定義的元音列表中。如果不在，則列印該字元 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 如果字元是元音，則不執行任何操作，即跳過列印 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
        - 每次列印字元時，使用 end=" " 參數確保輸出在同一行 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。最後，使用一個空的 print() 函數來換行 [PROBLEM SET 2: JUST SETTING UP MY TWTTR | SOLUTION]。
4. **Nutrition Facts（營養成分）**
    - **目標**：根據「Nutrition Facts」影片的說明，程式應根據使用者輸入的水果名稱，輸出該水果的卡路里含量 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。關鍵是，這要求使用**字典**來儲存水果及其卡路里資訊 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
    - **概念應用**：

        - 建立一個**字典**，其中水果名稱為鍵 (keys)，卡路里含量為值 (values) [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 透過 `input()` 函數獲取使用者輸入的水果名稱 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 為了實現**不區分大小寫**的比較，將使用者輸入的水果名稱和字典中的鍵都轉換為小寫，使用 **`.lower()` 方法** [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。同時，為處理輸入中可能存在的前後空格，可以使用 `in` 運算子來檢查輸入的字串是否包含字典中的鍵 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 使用 **`for` 迴圈** 迭代字典的鍵 (keys) [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 使用 **`if` 條件** 檢查當前迭代的鍵（經過小寫轉換）是否與使用者輸入的水果名稱（經過小寫轉換）匹配 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
        - 如果匹配，則列印該水果的卡路里，透過 `dictionary_name[key]` 的方式存取值 [PROBLEM SET 2: NUTRITION FACTS | SOLUTION]。
5. **Vanity Plates（車牌號碼）**
    - **目標**：根據「Vanity Plates」影片的說明，編寫一個名為 `is_valid` 的函數來檢查使用者輸入的車牌號碼是否符合一系列特定規則，並在主程式中根據函數回傳的布林值（`True` 或 `False`）列印「Valid」或「Invalid」 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
    - **概念應用**：
        - **函數定義**：實現 `is_valid(s)` 函數，該函數接收車牌字串 `s` 作為參數，並回傳 `True` 或 `False` [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
        - **`if __name__ == "__main__":` 結構**：確保 `main()` 函數在腳本作為主程式執行時被調用 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
        - **長度檢查**：使用 **`len()` 函數** 檢查車牌長度是否介於 2 到 6 個字元之間。如果不在這個範圍內，則回傳 `False` [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
        - **開頭字母檢查**：使用索引 `s` 和 `s` 以及 **`.isalpha()` 方法** 檢查車牌的前兩個字元是否為字母。如果不是，則回傳 `False` [PROBLEM SET 2: VANITY PLATES | SOLUTION]。

- **數字位置檢查**：
  - 數字不能在車牌中間使用，必須出現在末尾 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
  - 第一個使用的數字不能是零 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
  - 使用 **while 迴圈** 遍歷字串，當遇到第一個數字時，檢查它是否為零 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。如果為零，則回傳 `False` [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
  - 在找到第一個數字後，應使用 **break 關鍵字** 跳出 `while` 迴圈，因為後續的字元也應該是數字 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。如果之後出現字母，也應判斷為無效 [未在來源中明確說明此邏輯，但這是解決該規則的常見方法]。
- **禁止字元檢查**：使用 **for 迴圈** 迭代車牌中的每個字元，並使用 **in 運算子** 檢查是否包含句點、空格、驚嘆號或問號等標點符號 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。如果包含，則回傳 `False` [PROBLEM SET 2: VANITY PLATES | SOLUTION]。
- 如果所有檢查都通過，則回傳 `True`，表示車牌有效 [PROBLEM SET 2: VANITY PLATES | SOLUTION]。