CS50 Python Lecture 2 重點整理

引入迴圈 (loops) 的核心概念：讓程式重複執行動作，例如使用 while 或 for 在序列上反覆處理

while 迴圈：當條件為真時持續執行；常搭配 break（強制跳出）與 continue（跳過本次迭代）使用。

for 迴圈：用於遍歷 list、string、range 產生器，較明確且常用於已知次數的重複。

常見控制流程：break 可終止迴圈，continue 跳過當前循環並進入下一次。

實作範例：包含 snake_case、Coke 機器投幣模擬、Twitter 簡訊過濾（去掉元音）、車牌驗證、營養資訊查詢等題目

✅CS50P Week 2（CS50's Introduction to Programming with Python）

Week 2 的主要學習內容：

- **迴圈**：while, for
- **資料結構操控**：list, dict
- **工具函式**：range, len
- **流程控制**：break, continue
- **None 類型** 與其應用
- **列表與字典推導式**（list/dict comprehensions）
- **string/list 方法**：像是 append(), join(), replace(), isupper(), isdigit() 等
- **問題集（Problem Set 2）包含**：
  - camel.py（CamelCase → snake_case）
  - coke.py（模擬投幣機）
  - twttr.py（過濾元音）
  - vanity.py（驗證 "vanity" 車牌）
  - nutrition.py（查詢水果卡路里）

🔁 loop function（迴圈重點整理）

1. **基本語法**

   ```
   python
   複製編輯
   while condition:
       # 程式碼區塊
   for item in iterable:
       # 程式碼區塊
   ```

2. **常用搭配**
   - range(start, stop, step) 回傳可迭代整數序列。
   - len(data_structure) 取得長度。
   - continue 用於跳過當前迭代。
   - break 用於跳出迴圈。
3. **字串/列表操作**
   - for letter in word: 常用於字串逐字處理。
   - 使用 str.isupper(), str.lower(), str.isdigit() 進行條件篩選。
   - 修改字串時避免同時遍歷與改變長度，常見技巧：建立新串（''.join(...) 或累加字元）。

4. **練習題技巧**
    - CamelCase → snake_case：遇大寫字母時插入 _ 並轉小寫。
    - **投幣機**：用 while total < 50: 持續接受輸入，並計算找零。

✓ 小結

主題	要點
迴圈結構	while, for, break, continue
資料結構操作	list, dict, range, len, **推導式**
字串處理	isupper(), lower(), join(...)
常見範例	CamelCase → snake, **投幣機, 過濾器, 車牌驗證, 判定營養**

loop 函數使用注意事項

1. ❗避免無窮迴圈（Infinite loop）

- 最常見的錯誤之一是 while 條件永遠為 True，導致程式永遠跑不完。
- ✓ 確保迴圈內部有機制讓條件**最終會變為 False**。

```
# 錯誤：條件永遠成立
while True:
    print("Hello")  # 除非用 break，否則無限執行
```

---

2. ⚠ 變數更新要正確

- 特別是在 while 迴圈中，如果你忘記更新計數器或條件變數，就會造成無窮迴圈。

```
i = 0
while i < 5:
    print(i)
    # i += 1 ← 忘記更新 i，將無限列印
```

---

3. 🔁 for 和 while 適用不同場景

- for 適合用在**明確知道重複次數**或可遍歷的物件（如 list, str, range）
- while 則適合用在**條件不確定**的情況（例如直到使用者輸入正確為止）

---

4. ⚠ 不要在遍歷時修改原始資料結構

- 如果你在 for 迴圈中嘗試刪除或插入 list 內的元素，可能會導致不可預期的錯誤。

```
numbers = [1, 2, 3, 4, 5]
for n in numbers:
    if n % 2 == 0:
        numbers.remove(n)  # 結果可能不是你預期的
```

✅ 解法：遍歷複製品或建立新結果：

```
filtered = [n for n in numbers if n % 2 != 0]
```

---

5. 💡 使用 break/continue 要小心邏輯

- break: 提早跳出整個迴圈
- continue: 跳過當次迴圈，繼續下一次

```
while True:
    x = input("Number: ")
    if not x.isdigit():
        continue  # 跳過非數字輸入
    if int(x) == 0:
        break     # 0 為結束條件
```

---

6. 📌 避免不必要的巢狀迴圈（Nested loops）

- 兩層以上的巢狀迴圈效率通常很差，要特別注意資料量是否過大。
- 可以用 dict, set, comprehensions 等技巧優化結構。

---

7. 🧹 初始化與清除變數的習慣

- 確保每次執行 loop 前後變數的狀態都在可控制範圍內。

---

8. 🐞 Debug 技巧

- 在 loop 裡加上 print() 或 logging 可以追蹤變數變化，有助於除錯。
- 但注意最後要移除這些 print，以免污染輸出。

✅ 加分小技巧

技巧	說明
enumerate()	同時取得索引與值：for i, val in enumerate(list)
zip()	同步遍歷多個 iterable
range(start, stop, step)	控制迴圈步長，適用於 for
List Comprehension	比 for 更簡潔、效能更佳的寫法