講義編輯

--程式語言: python

--課程主題: (每個主題後，請出2~3題實做題)

----1. 算術運算子

----2. 比較運算子、邏輯運算子、選擇結構語法

----3. 位元運算子

Python 程式語言學習講義

主題一：算術運算子 (Arithmetic Operators)

算術運算子用於執行常見的數學運算。Python 提供了以下幾種常見的算術運算子：

- + (加)：將兩個數值相加。
- - (減)：將兩個數值相減。
- * (乘)：將兩個數值相乘。
- / (除)：將兩個數值相除，結果永遠為浮點數（小數）。
- // (整數除法 / 取商數)：將兩個數值相除，無條件捨去小數點，只保留整數部分。
- % (取餘數)：計算兩個數值相除後的餘數。這在判斷倍數時非常實用。
- ** (指數 / 次方)：計算某數的次方。例如 2 ** 3 等於 8。

範例：

Python

```
a = 10
b = 3
print(a / b)   # 輸出: 3.3333333333333335
print(a // b)  # 輸出: 3
print(a % b)   # 輸出: 1
print(a ** b)  # 輸出: 1000
```

✍️ 主題一 實作題

實作題 1:【溫度轉換器】

請撰寫一個程式，讓使用者輸入攝氏溫度（Celsius），程式透過算術運算將其轉換為華氏溫度（Fahrenheit）並輸出。

(提示：華氏溫度 = 攝氏溫度 * (9/5) + 32)

實作題 2:【時間換算小幫手】

請撰寫一個程式，給定一個總秒數（例如 total_seconds = 3665），請利用整數除法 // 與取餘數 %，計算並輸出這等於幾小時、幾分鐘、幾秒。

實作題 3:【**BMI** 身體質量指數計算】

請讓使用者輸入身高（公分）與體重（公斤），先將身高轉換為公尺，接著利用算術運算子計算並印出 BMI 值。

(提示：*BMI = 體重(kg) / 身高(m)*的平方)

主題二：比較運算子、邏輯運算子、選擇結構

此主題是讓程式擁有「判斷能力」與「執行不同分支」的關鍵。

1. 比較運算子 **(Comparison Operators)**

用於比較兩個數值，結果會回傳布林值 (True 或 False)。

- == (等於)、!= (不等於)
- > (大於)、< (小於)
- >= (大於等於)、<= (小於等於)

範例題 1：【通關密碼檢驗】

- 使用到的比較運算子：== (等於)、!= (不等於)
- 情境說明：假設系統設定了一組通關密碼，我們需要請使用者輸入密碼，並利用比較運算子來檢查輸入的密碼是否與預設密碼完全相同。

```python
# 系統預設的正確密碼
correct_password = "python_is_fun"

# 讓使用者輸入密碼
user_input = input("請輸入通關密碼: ")

# 判斷輸入的密碼是否「等於」正確密碼
if user_input == correct_password:
    print("密碼正確，登入成功！🔓")
else:
    # 這裡隱含了 user_input != correct_password 的情況
    print("密碼錯誤，請重新嘗試。🔒")
```

範例題 2:【投票資格審查】

使用到的比較運算子: >= (大於等於)、< (小於)

情境說明: 根據台灣的法律, 年滿 20 歲的公民才具有完整的投票權。請撰寫一個程式, 讓使用者輸入年齡, 並判斷他是否已經具備投票資格。

Python

```python
# 讓使用者輸入年齡，並轉換為整數 (int)
age = int(input("請輸入您的年齡: "))

# 判斷年齡是否「大於或等於」20
if age >= 20:
    print("您已經年滿 20 歲，具備投票資格！🗳️")
else:
    # 也就是 age < 20 的情況
    years_left = 20 - age
    print(f"您還未滿 20 歲，再等 {years_left} 年就可以投票囉！")
```

2. 邏輯運算子 (Logical Operators)

用於組合多個條件判斷：在 Python 與大多數的程式語言中，我們通常將 `1` 視為 `True` (成立 / 真)，將 `0` 視為 `False` (不成立 / 假)。

`and` (且) 運算子

- 判斷規則：必須「全部的條件都是 1 (True)」，結果才會是 1。只要遇到任何一個 0 (False)，結果就是 0。

條件 A	運算子	條件 B	結果 (A and B)	記憶口訣
1	and	1	1	兩者皆真才為真
1	and	0	0	有假即為假
0	and	1	0	有假即為假
0	and	0	0	兩者皆假必為假

`or` (或) 運算子

- 判斷規則：只要「其中有任何一個條件是 1 (True)」，結果就會是 1。必須全部都是 0 (False)，結果才會是 0。

條件 A	運算子	條件 B	結果 (A or B)	記憶口訣
1	or	1	1	有真即為真
1	or	0	1	有真即為真
0	or	1	1	有真即為真
0	or	0	0	兩者皆假才為假

`not` (反轉/非) 運算子

- 判斷規則：直接唱反調，將 1 變成 0，將 0 變成 1。它只需要搭配一個條件。

運算子	條件 A	結果 (not A)	記憶口訣
not	1	0	真變假
not	0	1	假變真

綜合範例題:【遊樂園門票優惠判定系統】

情境說明: 一家遊樂園有著特定的門票優惠規則, 我們需要寫一段程式來判斷遊客應該買哪種票。

- 免費票: 年齡小於 3 歲「或 **(or)**」年齡大於等於 80 歲的老少遊客。
- 學生票: 年齡介於 12 到 25 歲之間(包含 12 與 25 歲) ,「且 **(and)**」持有有效學生證的遊客。
- 一般票: 不符合以上任何條件的遊客。

```python
# 假設我們有一位遊客的資料
age = 20                    # 年齡 20 歲
has_student_id = True       # 有學生證 (True 代表有，False 代表沒有)

# 1. 先判斷是否符合「免費票」資格 (只要滿足其中一個年齡條件即可)
if age < 3 or age >= 80:
    print("您符合【免費票】資格，祝您玩得開心！🎡")

# 2. 判斷是否符合「學生票」資格 (必須同時滿足年齡區間與持有學生證)
elif age >= 12 and age <= 25 and has_student_id == True:
    print("您符合【半價學生票】資格！🎓")

# 3. 若以上皆不符合，就是一般票
else:
    print("您需購買【全票 (一般票)】。🎟️")
```

💡 學習重點拆解:

- `age < 3 or age >= 80`:因為使用了 `or`, 只要這位遊客是 2 歲(左邊成立)或是 85 歲(右邊成立), 整個條件就會過關。
- `age >= 12 and age <= 25 and has_student_id == True`:這裡使用了兩個 `and` 把三個條件串起來。這位 20 歲且有學生證的遊客, 這三個條件的真值表計算結果會是 `1 and 1 and 1`, 最終結果為 1 (True), 因此程式會印出他符合學生票資格。
- (進階小技巧):在 Python 中, 判斷區間 `age >= 12 and age <= 25` 其實可以簡寫成更像數學的寫法:`12 <= age <= 25`, 程式看起來會更簡潔喔！

3. 選擇結構 (Selection Structure: if / elif / else)

選擇結構(條件判斷)是讓程式變聰明的關鍵，它就像是給程式一張地圖，教它遇到不同的岔路時該怎麼走。

在 Python 中，我們主要使用 `if`、`elif`(else if 的縮寫)和 `else` 來建構選擇結構。我們將從最簡單的單一條件，一路進階到多重條件判斷。

1. 單向選擇：`if` (如果...)

這是最基礎的判斷。只有當條件成立(結果為 **`True`**)時，才會執行指定的程式碼；如果條件不成立(`False`)，程式會直接略過這段，繼續往下執行。

語法結構：

Python

```
if 條件判斷:
    # 條件成立時，要執行的程式碼 (注意前面要有縮排)
```

實際範例：

Python

```
weather = "下雨"

if weather == "下雨":
    print("記得帶傘出門！☔")

print("出發！") # 這行沒有縮排，無論是否下雨都會執行
```

2. 雙向選擇：`if ... else ...` (如果...否則...)

這就像是一個「非 A 即 B」的岔路口。如果條件成立，走第一條路；如果不成立，就走第二條路。

語法結構：

Python

```
if 條件判斷:
    # 條件成立時，執行的程式碼
else:
    # 條件【不成立】時，執行的程式碼
```

實際範例：

Python

```
score = 55

if score >= 60:
    print("恭喜，你及格了！🎉")
else:
    print("很遺憾，你需要補考。📝")
```

3. 多向選擇：`if ... elif ... else ...`（如果…又如果…否則…）

當遇到三個或更多選項時，我們就需要加入 `elif`。程式會由上往下依序檢查條件，一旦遇到「第一個」成立的條件，就會執行該區塊，執行完後便會直接跳出整個選擇結構，不會再檢查後面的條件。

語法結構：

Python

```
if 條件判斷 1:
    # 條件 1 成立時執行
elif 條件判斷 2:
    # 條件 1 不成立，但條件 2 成立時執行
elif 條件判斷 3:
    # 前面都不成立，但條件 3 成立時執行
else:
    # 所有條件都不成立時，執行這裡（else 是可省略的）
```

實際範例（以電影分級為例）：

Python

```
age = 15

if age >= 18:
    print("你可以觀看【限制級】電影。")
elif age >= 15:
    print("你可以觀看【輔15級】電影。")
elif age >= 12:
    print("你可以觀看【輔12級】電影。")
elif age >= 6:
    print("你可以觀看【保護級】電影。")
else:
    print("你只能觀看【普遍級】電影。")
```

💡 思考一下：在上面的例子中，如果把 `age >= 6` 寫在最上面第一行，當 `age = 15` 輸入時會發生什麼事呢？程式會印出「保護級」，然後就結束判斷了！這就是為什麼多重判斷時，條件的順序非常重要（通常由嚴格排到寬鬆，或由大排到小）。

⚠️ Python 語法的三大鐵則（新手最常踩的坑）

1. 冒號結尾 **(:)**：在 `if`、`elif`、`else` 的該行最後面，一定要加上英文半形的冒號 `:`，這是告訴 Python「接下來是這個條件的專屬地盤」。
2. 嚴格的縮排 **(Indentation)**：Python 不使用大括號 `{}` 來包覆程式碼，而是依賴「縮排」（通常是按一次 `Tab` 鍵，或是 4 個半形空白鍵）。只要縮排在同一層，Python 就會認為它們屬於同一個區塊。
3. 區分大小寫：`if` 必須全部小寫，寫成 `If` 或 `IF` 都會導致程式報錯。

✍️ 主題二 實作題

實作題 **1**：【奇偶數判斷】

請讓使用者輸入一個整數，利用選擇結構與餘數運算子 (%)，判斷並輸出該數字是「奇數」還是「偶數」。

實作題 **2**：【成績等第評估系統】

讓使用者輸入一個 0 到 100 的分數，根據以下規則輸出對應的等第：

- 90分(含)以上：A
- 80分 ~ 89分：B
- 70分 ~ 79分：C
- 60分 ~ 69分：D
- 60分以下：F (不及格)

實作題 **3**：【閏年判斷程式】

請使用者輸入一個西元年份（例如 2024），程式判斷並輸出該年是否為閏年。

(閏年規則：1. 西元年份除以 4 不可餘數 0 者為平年。2. 西元年份除以 4 可餘數 0 且除以 100 不可餘數 0 為閏年。3. 西元年份除以 100 可餘數 0 且除以 400 不可餘數 0 為平年。4. 西元年份除以 400 可餘數 0 為閏年。簡言之：逢 4 的倍數為閏年，但逢 100 的倍數不是閏年，而逢 400 的倍數又是閏年。)

主題三：位元運算子 (Bitwise Operators)

位元運算子是將數字轉換成「二進位 (Binary)」後，逐一對每個位元進行運算。這在底層系統開發、密碼學或追求極致效能時非常有用。

- & (位元 AND)：對應的兩個位元都為 1，結果才為 1。
- | (位元 OR)：對應的兩個位元只要有一個是 1，結果就是 1。
- ^ (位元 XOR)：對應的兩個位元不同時，結果為 1；相同時為 0。
- ~ (位元 NOT)：將二進位的 0 變 1，1 變 0（包含符號位元反轉）。
- << (左移)：將二進位數字向左移動指定的位數，右側補 0。（相當於乘以 2 的 n 次方）
- >> (右移)：將二進位數字向右移動指定的位數。（相當於除以 2 的 n 次方）

範例：

Python

```
a = 5      # 二進位: 0101
b = 3      # 二進位: 0011
print(a & b)  # 0001 (十進位: 1)
print(a | b)  # 0111 (十進位: 7)
print(a << 1) # 1010 (十進位: 10，相當於 5 * 2)
```

✍️ 主題三 實作題

實作題 **1**：【不使用第三變數交換數值】

在一般程式中交換兩個變數 a 和 b 的值通常需要一個暫存變數 temp。請嘗試只使用位元 XOR (^) 運算子，達成交換 a 與 b 數值的目的。

實作題 **2**：【極速奇偶數判斷】

在主題二我們用了 % 2 來判斷奇偶數，請改用位元 **AND (&)** 運算子來完成這件事。請讓使用者輸入一個整數，判斷它是奇數還是偶數。

(提示：任何奇數的最右邊一個二進位位元必定是 *1*。思考 *n & 1* 的結果會是什麼？)

實作題 **3**：【極速乘除法】

請宣告一個整數變數 num = 15。

1. 使用左移運算子 (<<) 將 num 乘以 4，並印出結果。
2. 使用右移運算子 (>>) 將 num 除以 2 取整數，並印出結果。