表單的頂端

Python 初學者指南：核心概念與程式實作

1. Python 基礎與執行環境

* **什麼是 Python？** Python 是一種多用途的程式語言，它支援多種程式典範，包括程序式、物件導向和函數式程式設計。
* **Python 版本：** 本文件專注於 Python 3 程式設計。
* **Python 程式的執行方式：互動式：** 使用 Python 直譯器即時執行程式碼。「Interactively Using the Python Interpreter」
* **執行 Python 檔案：** 透過執行一個包含 Python 程式碼的檔案。
* **執行 Python 腳本：** 類似於執行檔案，但通常指在命令列中直接運行。
* **整合開發環境 (IDE)：** 使用 PyCharm 等 IDE 進行開發和執行，這些 IDE 通常內建 Git 支援。
* **環境設定：** 文件涵蓋了在 Windows 和 Mac 上檢查及安裝 Python 的步驟。
* **模組與套件：模組：** 在 Python 中，一個模組等同於一個包含 Python 程式碼的檔案，可以包含函式、類別、變數和可執行程式碼。「In Python a module equates to a file containing Python code. A module can contain Functions, Classes, Variables, Executable code, Attributes associated with the module such as its name.」模組的名稱即是其檔案名稱（不含 .py 後綴）。
* **模組內容檢視：** 載入模組後，可以使用 dir(<module-name>) 函式獲取其內容列表。
* **將模組作為腳本執行：** Python 會根據檔案是否作為獨立腳本執行來設定模組屬性 \_\_name\_\_。當作為模組載入時，\_\_name\_\_ 設定為模組名稱；當作為獨立腳本執行時，\_\_name\_\_ 則為 \_\_main\_\_。這允許程式設計師區分兩種情況並執行特定的程式碼區塊。「This is because Python sets the module property \_\_name\_\_ to the name of the module when it is being loaded as a module; but if a file is being run as a standalone script (or the entry point of an application), the \_\_name\_\_ property is set to '**main**'.」
* **套件：** Python 允許將模組組織成層次結構的套件，這些套件是包含一個或多個 Python 原始碼檔案以及一個可選的 \_\_init\_\_.py 檔案的目錄。「A package is defined as a directory containing one or more Python source files and an optional source file named **init**.py.」

2. 基本資料型別

* **字串 (Strings)：定義：** 字串是文字字元序列。
* **操作：** 支援多種操作，例如：
* **串聯 (Concatenation)：** 使用 + 運算子連接字串。
* **長度 (Length)：** 使用 len() 函式獲取字串長度。
* **存取字元/子字串：** 使用索引和切片 (my\_string[1:5])。
* **重複 (Repeating)：** 使用 \* 運算子重複字串（例如 print('\*' \* 10) 會產生 \*\*\*\*\*\*\*\*\*\*）。
* **分割 (Splitting)：** 使用 split() 函式根據特定字元分割字串。
* **計數 (Counting)、替換 (Replacing)、尋找子字串 (Finding Sub Strings)。**
* **型別轉換：** 將其他型別轉換為字串。
* **比較 (Comparing)：** 比較字串。
* **注意事項：** Python 字串是區分大小寫的。「Python Strings Are Case Sensitive」。
* **數字 (Numbers)：型別：** 包含整數 (Integers)、浮點數 (Floating Point Numbers) 和複數 (Complex Numbers)。
* **型別轉換：** 可以將其他型別轉換為整數 (int()) 或浮點數 (float())。
* **算術運算子：** 支援標準算術運算子，例如加、減、乘、除、取模 (%)。取模運算子用於計算除法的餘數，這對於判斷數字的奇偶性特別有用。「represents the so-called modulo operator for modular arithmetic or at least in our case we're going to use it to calculate the remainder when dividing one number by another.」例如，偶數除以 2 的餘數為 0。
* **布林值 (Booleans)：定義：** 只有 True 或 False 兩種值。布林表達式是會產生「是/否」或「真/假」答案的問題。「technically called a boolean expression a boolean expression named after a mathematician named bull is simply a question that has a yes or no answer or technically a true or false answer.」
* **使用：** 常用於條件判斷。在 Python 中，布林值必須以大寫 T 和 F 開頭（True 和 False）。
* **None 值：** 表示缺少值。

3. 控制流程

* **條件語句 (If Statements)：if：** 用於基於布林表達式的結果執行程式碼塊。
* **elif：** else if 的簡寫，允許依序檢查多個條件。其優勢在於可以使條件互斥，一旦某個條件為 True，後續的條件將不再檢查，從而簡化邏輯並提高效率。「l if and this too is kind of a succinct one it's a conjunction of else if in english which allows us to ask a question that takes into account whether or not a previous question had a true or false answer.」
* **else：** 作為最終的「萬用」條件，當所有先前的 if 或 elif 條件都為 False 時執行。使用 else 可以避免檢查已知結果的條件，例如在判斷數字大小時，若不小於也不大於，則必然等於。「if the two conditionals evaluate to false there is only one um conditional that will evaluate to true and that is x is equal to y.」
* **語法：** if、elif、else 語句後接冒號 (:)，其下縮排的程式碼塊表示該條件成立時執行的內容。縮排是 Python 中定義程式碼塊的關鍵，通常為四個空格或一個 Tab。「i have typed a colon at the end of the line and even more importantly at the next line i have begun my line with some indentation hitting the spacebar four times or just hitting tab once which will automatically be converted to the same that indentation is what tells python that line 5 should only be executed if the answer to line 4's question is in fact true.」
* **比較運算子：** 例如 <, >, ==（用於相等比較，區別於 = 賦值運算子）。「if x equals equals y is how i express equality.」
* **邏輯運算子：** and, or, not。
* **巢狀 if 語句：** 在 if 語句內部再包含其他 if 語句。
* **迭代/迴圈 (Iteration/Looping)：while 迴圈：** 只要條件為真就重複執行程式碼塊。
* **for 迴圈：** 遍歷序列（如字串、列表、元組等）中的每個項目。
* **break 語句：** 終止當前迴圈的執行。
* **continue 語句：** 跳過當前迴圈的剩餘部分，進入下一個迭代。
* **for 迴圈與 else：** 當 for 迴圈正常完成（沒有被 break 終止）時，else 塊將被執行。

4. 函式 (Functions)

* **定義：** 函式是一段可重複使用的程式碼塊，用於執行特定任務。
* **好處：** 提高程式碼的組織性、可讀性和重用性。
* **型別：** 內建函式、使用者定義函式、匿名函式 (lambda)。
* **參數：** 函式可以接受參數，包括多個參數、預設參數值、命名參數和任意參數。
* **回傳值：** 函式可以使用 return 語句回傳值。
* **文件字串 (Docstring)：** 用於描述函式用途的文件。
* **作用域和生命週期：區域變數 (Local Variables)：** 在函式內部定義的變數，只在函式內部可存取。
* **全域變數 (Global Variables)：** 在函式外部定義的變數，可以在程式的任何地方存取。要修改函式內的全域變數，需要使用 global 關鍵字，否則 Python 會預設將其視為新的區域變數並報錯 UnboundLocalError。「To protect us from ourselves—Python is really saying ‘Do you really want to modify a global variable here?’. Instead it is treating max as a local variable 'max' referenced before assignment.」
* **非區域變數 (Nonlocal Variables)：** 用於巢狀函式中修改外層非全域變數。
* **高階函式 (Higher Order Functions)：** 接受函式作為參數或回傳函式的函式。
* **科里化函式 (Curried Functions)：** 將多參數函式轉換為一系列單參數函式的技術。
* **閉包 (Closures)：** 指一個函式記住了它被建立時的環境，即使該環境已經不存在，它仍然可以存取該環境中的變數。

5. 數據結構 (Collections)

* **元組 (Tuples)：特點：** 不可變 (immutable) 的有序序列。
* **建立：** 使用圓括號 () 或 tuple() 建構函式。
* **操作：** 可以存取元素、串聯、檢查元素是否存在、巢狀元組。不支援修改、新增或刪除元素。
* **列表 (Lists)：特點：** 可變 (mutable) 的有序序列。
* **建立：** 使用方括號 [] 或 list() 建構函式。
* **操作：** 存取元素、新增 (append(), insert())、移除 (remove(), pop(), del)、串聯。
* **集合 (Sets)：特點：** 無序、不重複元素的集合。
* **建立：** 使用花括號 {} 或 set() 建構函式。
* **操作：** 檢查元素是否存在、新增、移除、集合運算（聯集、交集、差集）。
* **字典 (Dictionaries)：特點：** 鍵值對 (key-value pairs) 的無序集合。鍵必須是不可變的，值可以是任意型別。
* **建立：** 使用花括號 {} 或 dict() 建構函式。
* **操作：** 透過鍵存取、新增、修改、移除條目、遍歷鍵、值或項目、檢查鍵是否存在。

6. 程式設計典範

* **函數式程式設計 (Functional Programming)：概念：** 將計算視為數學函式的求值，避免使用可變狀態和可變數據。
* **優勢：** 提高程式碼的模組化、可測試性、可維護性和並行性。
* **劣勢：** 對於需要頻繁修改狀態的問題可能不適用。
* **參照透明性 (Referential Transparency)：** 函式在相同輸入下總是產生相同輸出，且沒有副作用。
* **物件導向程式設計 (Object Orientation)：概念：** 將程式組織成互相協作的「物件」，每個物件都是類別的實例，包含數據（屬性）和操作數據的方法（行為）。
* **核心原則：類別 (Classes)：** 藍圖或範本，定義物件的屬性與行為。
* **物件/實例 (Objects/Instances)：** 類別的具體化。
* **封裝 (Encapsulation)：** 將數據和處理數據的方法綁定在一起，隱藏內部實現細節。
* **繼承 (Inheritance)：** 允許一個類別（子類別）繼承另一個類別（父類別）的屬性與方法，實現程式碼重用和階層結構。Python 支援多重繼承，但需注意其潛在的複雜性。「Python and Multiple Inheritance. Multiple Inheritance Considered Harmful.」
* **多型 (Polymorphism)：** 允許不同類別的物件對相同的訊息作出不同的響應，或以相同的介面處理不同型別的物件。Python 的「鴨子型別 (Duck Typing)」是其多型的一種實現，即不關心物件的實際型別，只關心它是否提供了所需的方法。「Duck Typing: If it walks like a duck and quacks like a duck, then it must be a duck.」
* **屬性與方法：實例方法 (Instance Methods)：** 作用於特定物件的數據。
* **類別側數據 (Class Side Data) 與方法 (Class Side Methods)：** 屬於類別本身而非其實例的數據和方法。
* **靜態方法 (Static Methods)：** 不依賴於類別或其實例狀態的方法。
* **運算子重載 (Operator Overloading)：** 允許類別定義特定運算子（如 +, -, ==）的行為，使其能作用於自定義的物件。
* **屬性 (Properties)：** 提供控制屬性存取的方式，可以透過 setter 和 getter 方法來實現。
* **抽象基底類別 (Abstract Base Classes, ABCs)：** 定義介面或契約，強制子類別實現特定方法。「Abstract Classes as a Concept. Defining an Abstract Base Class. Defining an Interface.」
* **協議 (Protocols)：** Python 中實現多型的一種方式，通常是隱式契約，透過約定俗成的命名和行為來實現。「Implicit Contracts. Protocols. An Protocol Example.」
* **描述器 (Descriptors)：** 是一種 Python 物件，可以控制類別屬性的存取行為。
* **猴子補丁 (Monkey Patching)：概念：** 在執行時動態修改或擴展類別或模組的行為。
* **工作原理：** 透過直接修改物件的屬性字典來實現。「How Does Monkey Patching Work?」
* **屬性查找：** Python 在查找屬性時遵循特定的順序，這使得猴子補丁成為可能。
* **裝飾器 (Decorators)：概念：** 一種修改或增強函式或類別行為的語法糖，本質上是一個接受函式或類別作為參數並回傳新函式或類別的函式。「What Are Decorators?」
* **使用：** 透過 @decorator\_name 語法應用於函式或方法定義上方。
* **型別：** 函式裝飾器、方法裝飾器、類別裝飾器、帶參數的裝飾器和堆疊裝飾器。

7. 進階概念

* **可迭代對象 (Iterables)、迭代器 (Iterators) 與生成器 (Generators)：可迭代對象：** 任何可以被 for 迴圈遍歷的對象（例如列表、元組、字串）。
* **迭代器：** 實現了 \_\_iter\_\_() 和 \_\_next\_\_() 方法的對象，用於在遍歷過程中追蹤狀態。
* **生成器：** 一種特殊的函式，使用 yield 關鍵字，在每次調用時產生一個值，並在暫停執行後保留狀態，下次從上次停止的地方繼續執行。它們是建立迭代器的簡潔方式，並且記憶體效率更高。「Defining a Generator Function. When Do the Yield Statements Execute?」
* **協程 (Coroutines)：** 類似生成器但更通用，可以通過 yield from 暫停和恢復執行，並在它們之間傳遞數據。
* **map, filter, reduce：filter()：** 根據指定函式的結果（True 或 False）過濾可迭代對象中的元素。
* **map()：** 對可迭代對象中的每個元素應用指定函式，並回傳一個新的迭代器。
* **reduce()：** 將一個函式連續地應用於序列中的元素，將序列歸約為單一累積值（在 functools 模組中）。
* **抽象資料型別 (ADTs)、佇列 (Queues) 和堆疊 (Stacks)：抽象資料型別 (ADTs)：** 僅定義數據的行為和操作，而不涉及具體實現。
* **數據結構 (Data Structures)：** 實際儲存數據的物理或邏輯安排。
* **佇列 (Queues)：** 先進先出 (FIFO) 的數據結構。Python 列表可以用作基本佇列。
* **堆疊 (Stacks)：** 後進先出 (LIFO) 的數據結構。Python 列表可以用作基本堆疊。
* **錯誤與異常處理 (Error and Exception Handling)：錯誤 (Errors) 與異常 (Exceptions)：** 程式執行過程中發生的問題。
* **異常處理：** 使用 try, except, else, finally 塊來捕獲和處理異常，防止程式崩潰。
* **拋出異常 (Raising an Exception)：** 使用 raise 關鍵字觸發自定義或內建異常。
* **自定義異常 (Custom Exception)：** 透過繼承 Exception 類別來定義特定的異常型別。
* **異常鏈 (Chaining Exceptions)：** 保留原始異常資訊，便於追蹤問題。

8. 專案實例與練習

* **「Hello World」：** 基礎的入門程式。
* **數字猜謎遊戲 (Number Guessing Game)：** 整合隨機數生成、使用者輸入、條件判斷和迴圈的實作。
* **計算機 (Calculator)：** 應用函式組織不同運算，處理使用者輸入和選擇操作。
* **井字遊戲 (TicTacToe Game)：** 作為一個更複雜的物件導向程式設計範例，展示了類別（如 Counter, Move, Player, HumanPlayer, ComputerPlayer, Board, Game）如何協同工作來構建一個應用程式。

表單的底部