

AI 學習革命： 為什麼用 AI 學程式最有效？

本章學習重點

- 核心觀念：強調 AI 在程式學習中的角色，將 AI 工具定位為「AI 程式家教」、「AI 出題老師」、「AI 除錯助手」與「AI 程式優化顧問」。
- 提問黃金法則：提出五大提問技巧
 - 明確具體。
 - 分步拆解。
 - 設定角色與情境。
 - 附上範例、格式或預期輸出。
 - 反覆修正與優化。
- AI 設定學習路線圖：利用 AI 工具根據個人時間、目標和程度，量身打造學習計畫。」

學程式不再只是死背語法，而是學會如何「用對工具問對問題」。本章將帶你進入 AI 程式學習的嶄新時代，了解 大型語言模型（LLM）的強大能力，以及如何善用 AI 工具作為學習助教、改錯幫手與實作搭檔。

透過掌握提問技巧與學習策略設定，將建立起屬於自己的學習路線圖，真正進入 AI 協作式學習的最佳狀態！

1-5 用 AI 設定學習路線圖

——讓 AI 工具當專屬教練，規劃一條最適合的程式學習路線。

很多人學程式會有這些問題

- 「我該先學什麼？再學什麼？」
- 「每天要學多久？什麼時候能寫出一個小作品？」
- 「我只有三週時間，能不能幫忙排一套密集學習計畫？」

這一節講解：如何請 AI 工具根據程度與目標，量身打造一套學習計畫，就像請了一位全天候的教練與課程設計師。

目標不是「學很多」，而是「學對內容、用對方法」。

AI 工具可以根據需求與時間，幫助

告訴 AI 的條件	AI 可以幫忙做的事
有的時間（每天多久、總共幾週）	幫你分配學習節奏與進度。
目標（考 APCS、做出專案、打好基礎）	幫你安排適合的主題與練習。
程度（完全新手、有寫過一點）	幫你挑選入門到進階的學習順序。

範例提問句（立即可用）

可以直接這樣對 AI 工具說

1. 我是一位完全沒程式基礎的學生，想在一個月內學會 Python 入門，請幫忙安排每週學習計畫。」
2. 我有基礎，想練 APCS 題目，請幫忙設計兩週的題型強化路線。」

3. 我想做一個 BMI 計算器小工具，請告訴我需要先學哪些語法，多久可以完成？」
4. AI 會根據背景與需求，規劃每天、每週要做什麼，甚至包含：
 - a. 學哪些語法主題。
 - b. 練哪些題型。
 - c. 哪一天來寫一個小專案。

範例：AI 工具回答可能長這樣

- Week 1：學習變數、input/print、資料型別，完成實作練習：輸入姓名並問候。
- Week 2：練習 if/else、流程控制與邏輯判斷，完成挑戰：成績判定器。
- Week 3：迴圈與 list 應用，練習多筆資料操作。
- Week 4：整合練習，完成 BMI 計算器（加入 GUI 介面）。

實作練習：用 AI 工具幫助自己排一份「Python 學習計畫」

請打開 AI 工具，輸入

- 「我想從今天開始，每週學 3 天、每天 1 小時，請幫忙規劃 4 週的 Python 初學者學習計畫，請列出每週主題與練習方向。」

接著可以進一步要求

1. 請加入 AI 工具練習題互動。
2. 每週加入一個實作練習。
3. 請以表格格式輸出。

1-6 牛刀小試，用 AI 認識一下 Python

——AI 輔助，認識 Python 優點與語法結構。

在正式開始學習 Python 之前，我們先用 AI 幫助快速建立對 Python 的基本認識。這一節就像暖身運動，讓你理解為什麼大家都學 Python，它有哪些優點，以及入門要掌握哪些核心語法。



學 Python 的五大好處

1. 簡單好上手，適合新手

語法簡潔，接近英文，自然易讀，初學者不易挫折。

2. 用途廣泛，跨領域實用

可以用來做資料分析、AI、網站、遊戲、自動化、網頁爬蟲等。

3. 職場需求高，就業選擇多

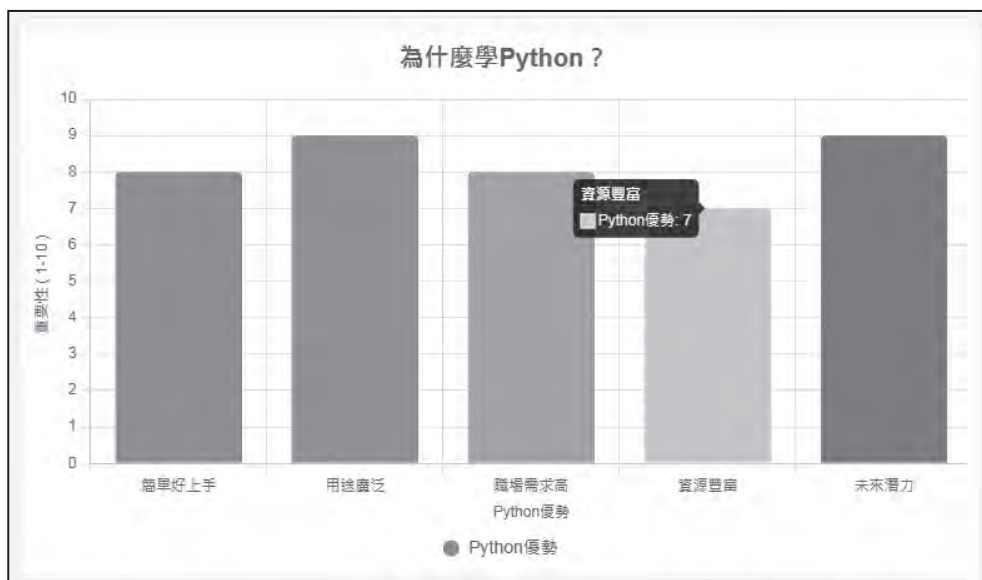
資料科學、AI 工程、後端開發、DevOps 等職位都重視 Python 能力。

4. 學習資源豐富，社群活躍

線上免費教材、影片、論壇多，找資料、解問題都很方便。

5. 未來潛力大，發展空間廣

是人工智慧和自動化趨勢下最重要的語言之一。



Python 學習的第一關：保留字

Python 有一組被保留的字詞，稱為「保留字 (keywords)」，這些字詞有特定用途，不能拿來當變數名稱。以下是常見的保留字分類與範例：

- 條件與邏輯判斷：if, else, elif, and, or, not
- 迴圈與流程控制：for, while, break, continue, pass
- 函式與類別定義：def, return, class, lambda
- 例外處理與錯誤偵測：try, except, raise, finally, assert
- 模組匯入與命名控制：import, from, as, with
- 其他常見：True, False, None, in, is, global, nonlocal

在 Python 中可以這樣列出所有保留字：

```
import keyword  
print(keyword.kwlist)
```

目前（Python 3.11）總共有 35 個保留字，初學者可先記住常用的基本組。
就 35 個英文字，不用擔心英文不好。

學 Python 常見語法句型總覽

下列是 Python 中最常見、也是最基本的語法句型，幫助快速熟悉語法結構。

1. 變數與資料型別

- 指定變數：x = 10
- 字串：name = "Amy"
- 清單：fruits = ["apple", "banana"]
- 布林值：is_ok = True

2. 輸入與輸出

- 輸出：print("Hello")
- 輸入：name = input(" 你的名字：")
- 轉數字：age = int(input(" 年齡："))

3. 條件判斷

- 單一條件：if score >= 60:
- 多條件：elif, else
- 巢狀判斷：條件中再放條件

4. 迴圈

- `for i in range(5):`
- `while x < 10:`
- `break, continue` 控制流程

5. 函式定義與呼叫

- 定義函式：`def greet(name):`
- 回傳值：`return "Hello " + name`
- 呼叫函式：`greet("Amy")`

6. 模組匯入與使用

- 匯入模組：`import math`
- 使用功能：`math.sqrt(16)`

就 10 幾個英文句型，就可以駕馭 python 語言了，趕快來看看該怎麼用吧！

寫一個遊戲： 邊問邊做的開發實戰

本章學習重點

- 專案整合：將前述所學的變數、條件、迴圈、函式和模組整合，開發「猜數字遊戲」和「打字練習遊戲」。
- AI 遊戲開發助理：展示 AI 工具如何協助遊戲邏輯設計、補語法、除錯、優化及增加新功能。
- 終端互動技巧：學習美化終端輸出，如換行、分隔線、延遲輸出等，提升使用者體驗。

學了那麼多基礎，現在來實戰吧！本章將帶你開發兩款經典文字小遊戲：「猜數字」與「打字練習」。這些遊戲不但能練邏輯、用迴圈與條件控制，還能實際操作變數、函式與隨機模組。

而且，你將學會用 AI 工具做遊戲開發助理：協助設計邏輯、補語法、改錯、優化，甚至協助增加新功能。這是一章最能體會「邊問邊寫」學習威力的章節！

7-1 猜數字遊戲設計

——從互動遊戲入門，把邏輯與語法連起來！

現在，你已經學會變數、條件判斷、迴圈、函式與模組該怎麼用了。

是時候把這些東西組合起來，做一點好玩又實用的事！

這一節，我們就要實作經典的《猜數字》遊戲。

猜數字遊戲怎麼玩？

目標：讓玩家猜出電腦「隨機想的一個數字」。

玩法：

1. 電腦使用 `random.randint(1, 100)` 產生一個神秘數字。
2. 玩家每次輸入一個猜測數字。
3. 程式會回應：「太大了」「太小了」或「猜對了！」。
4. 玩家在有限次數內猜中即勝利，否則失敗。

會用到的 Python 技能

技能	說明
<code>random.randint()</code>	產生隨機數字。
<code>input()</code> / <code>int()</code>	玩家輸入處理。
<code>if</code> / <code>elif</code> / <code>else</code>	判斷猜得太大或太小。
<code>while</code> 迴圈	反覆讓玩家猜。
變數控制	計算回合數與控制遊戲流程。

遊戲邏輯圖解（流程簡化版）

[開始遊戲]



產生隨機數字（1~100）



[重複以下流程直到猜中或次數用完]

玩家輸入數字 → 判斷：

→ 猜太小 → 提示太小

→ 猜太大 → 提示太大

→ 猜對了！→ 遊戲結束



延伸任務建議

功能	敘述
限制最多猜 5 次	用變數記錄猜的次數，超過就結束。
顯示猜過的數字	每次輸入後加入 list，結束後一併列出。
多次遊玩	遊戲結束後問玩家要不要再玩一次。
加入難度選項	請玩家選 Easy/Hard，改變數字範圍與次數。



AI 工具提問模板（你可以這樣問 AI）

1. 我想寫一個猜數字遊戲，請幫忙從 1~100 隨機產生數字，並讓玩家猜。
2. 我寫了一個猜數字遊戲，想加入「最多猜 5 次就結束」的功能，怎麼寫？
3. 請幫忙改成「猜對後顯示總共猜了幾次」。
4. 幫忙加入「難度選擇：簡單（1~50）／困難（1~1000）」的功能。

AI 工具實作

我想寫一個猜數字遊戲，請幫忙從 1~100 隨機產生數字，並讓玩家猜。



我想寫一個猜數字遊戲，請幫我從 1~100 隨機產生數字，並讓玩家猜。

這是一個基本的「猜數字」Python 遊戲範例，從 1 到 100 隨機產生一個整數，然後讓玩家反覆猜，直到猜中為止：

```
python 📋 複製 📄 編輯  
  
import random  
  
# 產生 1 到 100 的隨機整數  
answer = random.randint(1, 100)  
  
print("歡迎來到猜數字遊戲！")  
print("我已經想好了一個 1 到 100 之間的數字。")
```

```
# 玩家反覆猜直到答對為止  
while True:  
    guess = input("請輸入你的猜測 (1~100): ")  
  
    # 判斷輸入是否為數字  
    if not guess.isdigit():  
        print("請輸入一個有效的整數！")  
        continue  
  
    guess = int(guess)  
  
    # 檢查範圍  
    if guess < 1 or guess > 100:  
        print("請輸入 1 到 100 之間的數字！")  
        continue  
  
    if guess < answer:  
        print("太小了！")  
    elif guess > answer:  
        print("太大了！")  
    else:  
        print("恭喜你，猜對了！")  
        break
```

加入「最多猜 5 次就結束」的功能

➡ 加入「最多猜 5 次就結束」的功能

當然可以！以下是加入「最多只能猜 5 次」限制的版本：

```
python 📄 複製 📄 編輯  
  
import random  
  
# 產生 1 到 100 的隨機整數  
answer = random.randint(1, 100)  
  
print("歡迎來到猜數字遊戲！")  
print("我已經想好了一個 1 到 100 之間的數字 = 你有 5 次機會猜對它！")  
  
# 最多猜 5 次  
max_attempts = 5  
attempts = 0  
  
while attempts < max_attempts:  
    guess = input(f"第 {attempts + 1} 次猜測，請輸入你的猜測 (1~100) :")  
  
    # 判斷輸入是否為數字  
    if not guess.isdigit():  
        print("請輸入一個有效的整數！")  
        continue  
  
    guess = int(guess)
```

```
# 檢查範圍
if guess < 1 or guess > 100:
    print("請輸入 1 到 100 之間的數字！")
    continue

attempts += 1

if guess < answer:
    print("太小了！")
elif guess > answer:
    print("太大了！")
else:
    print(f"恭喜你，猜對了！答案就是 {answer}！你總共猜了 {attempts} 次。")
    break
else:
    print(f"很可惜，你已經猜了 {max_attempts} 次。答案是 {answer}，下次加油！")
```

功能摘要：

- 玩家最多只能猜 5 次。
- 每次會顯示目前是第幾次猜。
- 猜對即結束，猜錯滿 5 次也結束，並公布答案。

是否還需要加入其他提示或介面優化？例如提示「你只剩下幾次機會」？

📄 📌 🗨 🔊 ✎

最後結果

```
import random

print("歡迎來到猜數字遊戲！")
print("請選擇遊戲難度：")
print("1. 簡單（1 ~ 50）")
print("2. 困難（1 ~ 1000）")

# 選擇難度
while True:
    level = input("請輸入 1 或 2：")
    if level == "1":
        max_number = 50
        break
    elif level == "2":
        max_number = 1000
        break
    else:
        print("請輸入有效的選項（1 或 2）")

# 產生隨機答案
answer = random.randint(1, max_number)
print(f"\n我已經想好了一個 1 到 {max_number} 之間的數字。你有 5 次機會猜中它！")

# 猜測流程
max_attempts = 5
attempts = 0

while attempts < max_attempts:
    guess = input(f"第 {attempts + 1} 次猜測，請輸入你的猜測（1~{max_number}）：")
```

```
# 檢查是否為整數
if not guess.isdigit():
    print("請輸入一個有效的整數！")
    continue

guess = int(guess)

# 檢查是否在範圍內
if guess < 1 or guess > max_number:
    print(f"請輸入 1 到 {max_number} 之間的數字！")
    continue

attempts += 1

if guess < answer:
    print("太小了！")
elif guess > answer:
    print("太大了！")
else:
    print(f"恭喜你，猜對了！答案是 {answer}，你總共猜了 {attempts} 次。")
    break
else:
    print(f"很可惜，你已經猜了 {max_attempts} 次。答案是 {answer}，下次加油！")
```