# Python基礎教學 v3

這份教材以「初學者看得懂、做得出來」為目標：每個語法段落都包含**說明、一個範例題目（含參考解答）**、以及**兩個練習題**。

建議學習方式：

1. 先看「說明」理解概念
2. 自己先做「範例題目」再對照解答
3. 完成「練習題」並自己驗證輸出

---

## 目錄



---

## 1. 甚麼是程式語言? Python介紹

### 1.1 甚麼是程式語言？

你可以把「程式語言」想成：**跟電腦溝通的一套規則**。電腦本質上只會照著指令做事，而程式語言就是用來把「你想做的事情」寫成電腦能執行的步驟。

一個程式通常包含：

- **輸入（Input）**：從鍵盤、檔案、網路、資料庫取得資料
- **處理（Process）**：計算、判斷、整理、分類
- **輸出（Output）**：印在螢幕、寫入檔案、回傳到網頁、產生報表

### 1.2 為什麼選 Python？

Python 常被當作入門首選，原因包括：

- **語法接近人類語言**：可讀性高
- **用途廣**：自動化、資料分析、網頁後端、爬蟲、AI/機器學習
- **社群大、資源多**：遇到問題更容易找到答案

> 本教材假設你使用 Python 3.10 以上（因為會用到 `match`）。

### 1.3 到底該學哪個程式語言？

選語言最實用的方式是：**先想你要做什麼**。

- 想做網頁前端（畫面互動）：JavaScript / TypeScript
- 想做網站後端、API：Python / Node.js / Java / C#
- 想做手機 App：Swift（iOS）/ Kotlin（Android）
- 想做資料分析、AI、自動化：Python（最常見）
- 想做遊戲或高效能：C++ / C#

如果你現在還不確定方向，Python 的「通用性」會讓你學了不吃虧：學到的基本概念（變數、條件、迴圈、函式、資料結構）之後換語言也能用。

### 1.4 AI 已經很強大了，我還需要學程式語言嗎？

需要，原因很務實：

1. **你要能把需求說清楚**：AI 生成程式碼靠的是你的描述（prompt），懂程式你才知道要補哪些條件。
2. **你要能判斷對不對**：AI 可能會產生看似合理但有 bug 的程式；你需要能讀懂與測試。
3. **你要能整合與維護**：真實工作不是一段程式碼就結束，常常要串接資料、檔案、API、排程。

學程式語言不是跟 AI 競爭，而是讓你能「駕馭工具」。

---

## 2. 簡單的應用場景介紹

下面是 Python 常見的「入門就做得出來」的應用：

1. **自動化**
   - 批次改檔名、整理資料夾
   - 讀寫 Excel、產生報表
   - 每天定時下載資料
2. **資料處理與分析**
   - 清理 CSV/Excel
   - 計算平均、排名、趨勢

3. **網路爬蟲**（注意網站規範）
    - 把網頁上的表格抓下來整理
4. **網頁後端 / API**
    - 用 Flask/FastAPI 提供資料查詢
5. **AI 應用**
    - 串接大型語言模型 API
    - 建立簡單的聊天機器人

初學者最推薦從「自動化 + 讀寫檔案」開始，因為能很快看見成果。

---

# 3. Python安裝與設定（Windows）

## 3.1 安裝 Python

建議做法：使用官方安裝程式。

安裝重點：

1. 安裝時勾選 **Add Python to PATH**
2. 建議安裝 Python 3.11 或 3.12（新版本效能與相容性通常更好）

## 3.2 驗證是否安裝成功

打開 PowerShell 後輸入：

```
python --version
pip --version
```

如果你電腦同時裝了多個 Python，有些環境需要改用：

```
py -V
py -m pip --version
```

## 3.3 建立專案與虛擬環境（推薦）

「虛擬環境」可以把每個專案用到的套件隔離開來，避免版本互相打架。

```
cd "c:\Users\purem\OneDrive\文件\TextBook"
mkdir my_python_project
cd my_python_project

py -m venv .venv
.\.venv\Scripts\Activate.ps1

python -m pip install --upgrade pip
```

看到命令列前面出現 `(.venv)` 通常表示啟動成功。

---

# 4. 執行環境介紹, IDE介紹

## 4.1 你會遇到的「執行方式」

1. **互動式（REPL）**：
   - 直接輸入一行跑一行，適合試語法
2. **執行檔案（.py）**：
   - 寫成程式檔，適合做完整功能
3. **Notebook（.ipynb）**：
   - 常用在資料分析與教學（Jupyter）

## 4.2 IDE 是什麼？

IDE（整合開發環境）可以幫你：

- 自動排版、提示錯誤
- 斷點除錯（Debug）
- 管理專案與套件

常見選擇：

- **VS Code**：輕量、外掛多、很常用
- **PyCharm**：功能完整，較重

## 4.3 VS Code 建議設定

1. 安裝 Python 外掛（Microsoft Python）
2. 選擇解譯器（Interpreter）為你的 `.venv`
3. 用內建終端機執行：

```
python your_script.py
```

---

# 5. Python基本語法

本章每一節都會用「說明 → 範例題目（含解答）→ 練習題」的方式。

## 5.1 變數與資料型態

**說明**

**變數**就是替資料取名字，讓你之後能拿來用。

Python 常見資料型態：

- `int`：整數，例如 `10`

- `float`：小數，例如 `3.14`
- `str`：字串，例如 `"hello"`
- `bool`：布林值（真假），`True` / `False`
- `None`：代表「沒有值」

你可以用 `type()` 看型態：

```
x = 10
print(type(x))  # <class 'int'>
```

**範例題目：計算總價**

題目：

輸入商品單價 `price` 與數量 `qty`，計算總價 `total`，並印出：

`單價=..., 數量=..., 總價=...`

參考解答：

```
price = 35
qty = 4

total = price * qty
print(f"單價={price}, 數量={qty}, 總價={total}")
```

**練習題**

1. 設定 `height_cm = 170`、`weight_kg = 65`，計算 BMI（$BMI=weight/(height_m^2)$），印出到小數點後 2 位。
2. 讓使用者輸入名字與年齡（用 `input()`），印出：`你好 {名字}，明年你 {年齡+1} 歲`。

---

## 5.2 運算子

**說明**

常用運算子：

- 算術：`+ - * / // % **`
  - `/`：除法（會有小數）
  - `//`：整除（只取商）
  - `%`：取餘數
  - `**`：次方
- 比較：`== != > < >= <=`（結果是 `True/False`）
- 邏輯：`and or not`

**範例題目：判斷能否被 3 整除**

題目：

給定整數 `n`，如果 `n` 可以被 3 整除，印出 `True`，否則印出 `False`。

參考解答：

```
n = 21
print(n % 3 == 0)
```

**練習題**

1. 給定 `a=17`、`b=5`，分別印出 `a/b`、`a//b`、`a%b`。
2. 給定分數 `score`，判斷是否在 60 到 100（含）之間，印出布林值。

---

## 5.3 控制流程

控制流程讓程式能「做選擇」或「重複執行」。

### 5.3.1 if

**說明**

`if` 用來做判斷：條件成立才做某件事。

```
if 條件:
    # 條件成立做這裡
else:
    # 否則做這裡
```

**範例題目：成績等第**

題目：

給定 `score`，

- `>= 90` 印 `A`
- `>= 80` 印 `B`
- `>= 70` 印 `C`
- 否則印 `D`

參考解答：

```
score = 83
```

```
if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
else:
    print("D")
```

**練習題**

1. 輸入年齡 `age`：`< 18` 印 `未成年`，否則印 `成年`。
2. 輸入兩個數字 `x, y`，印出較大的那個（相等就印 `一樣大`）。

---

### 5.3.2 for

**說明**

`for` 常用來「跑一段範圍」或「走訪序列（例如串列、字串）」。

```
for i in range(5):
    print(i)  # 0~4
```

**範例題目：計算 1 到 n 的總和**

題目：

給定 `n`，用 `for` 計算 `1+2+...+n`。

參考解答：

```
n = 10
total = 0

for i in range(1, n + 1):
    total += i

print(total)
```

**練習題**

1. 用 `for` 印出 1 到 100 之間所有偶數。
2. 給定字串 `s = "python"`，用 `for` 逐字印出每個字母。

---

### 5.3.3 break

#### 說明

`break` 用來「提前結束迴圈」。

#### 範例題目：找到第一個能被 7 整除的數

題目：

從 1 開始往上找，找到第一個能被 7 整除、且同時能被 5 整除的數字，找到後印出並停止。

參考解答：

```
for n in range(1, 10000):
    if n % 7 == 0 and n % 5 == 0:
        print(n)
        break
```

#### 練習題

1. 從 1 開始找，找到第一個平方大於 500 的整數 `n`，印出 `n` 並停止。
2. 用 `for` 走訪一個串列 `nums=[3, 8, 2, 9, 7]`，找到 `9` 就印 `found` 然後停止。

---

### 5.3.4 continue

#### 說明

`continue` 用來「跳過本次迴圈剩下的程式」，直接進入下一次。

#### 範例題目：加總奇數

題目：

將 1 到 100 的奇數加總。

參考解答：

```
total = 0
for n in range(1, 101):
    if n % 2 == 0:
        continue
    total += n

print(total)
```

**練習題**

1. 印出 1 到 50，但遇到 3 的倍數不要印（跳過）。
2. 走訪字串 `"a1b2c3"`，只印出其中的字母（略過數字）。

---

### 5.3.5 pass

**說明**

`pass` 表示「先留空」，讓程式在語法上完整但暫時不做事。常用於：

- 先把架構寫好
- 之後再補內容

**範例題目：先建立判斷架構**

題目：

寫一個程式：如果 `mode == "dev"` 先留空；如果 `mode == "prod"` 印出 `running`。

參考解答：

```
mode = "dev"

if mode == "dev":
    pass
elif mode == "prod":
    print("running")
```

**練習題**

1. 建立一個函式 `todo()`，先用 `pass` 佔位（不要報錯）。
2. 用 `if/elif/else` 建立三種狀態的架構，先全部 `pass`，之後再補功能。

---

### 5.3.6 match（Python 3.10+）

**說明**

`match` 是「模式比對」，很像其他語言的 switch，但更強。

```
match value:
    case 1:
        ...
    case 2:
        ...
```

```
    case _:
        ...  # 預設
```

**範例題目：星期轉中文**

題目：

給定 `day`（1~7），印出對應中文：1=一、2=二…7=日，其它印出 `輸入錯誤`。

參考解答：

```
day = 6

match day:
    case 1:
        print("星期一")
    case 2:
        print("星期二")
    case 3:
        print("星期三")
    case 4:
        print("星期四")
    case 5:
        print("星期五")
    case 6:
        print("星期六")
    case 7:
        print("星期日")
    case _:
        print("輸入錯誤")
```

**練習題**

1. 用 `match` 做簡單計算器：`op` 可能是 `+ - * /`，根據 `op` 計算 `a op b`。
2. 用 `match` 根據字串 `"start"/"stop"/"pause"` 印出不同訊息，其它印 `unknown`。

---

## 5.4 函式

**說明**

函式（function）就是把一段可以重複用的程式「包起來」，取一個名字。

```
def add(a, b):
    return a + b
```

重點：

- **參數**：函式需要的輸入
- **回傳值**：函式算完的結果

**範例題目：寫一個判斷偶數的函式**

題目：

寫 `is_even(n)`，如果 `n` 是偶數回傳 `True`，否則回傳 `False`。

參考解答：

```
def is_even(n: int) -> bool:
    return n % 2 == 0


print(is_even(10))  # True
print(is_even(7))   # False
```

**練習題**

1. 寫 `circle_area(r)` 回傳圓面積（$\pi r^2$），並呼叫它印出半徑 3、5 的結果。
2. 寫 `count_vowels(s)` 回傳字串中母音（a,e,i,o,u）數量（不分大小寫）。

---

## 5.5 資料結構（串列、字典、集合、元組）

**說明**

資料結構用來「把多個資料放在一起」。

- 串列 `list`：有順序、可修改
  - 例：`["apple", "banana"]`
- 元組 `tuple`：有順序、通常不修改
  - 例：`("台北", "台中")`
- 字典 `dict`：用 key 找 value
  - 例：`{"name": "Amy", "age": 18}`
- 集合 `set`：不重複、無順序
  - 例：`{1, 2, 3}`

**範例題目：統計成績平均**

題目：

給定成績串列 `scores`，計算平均並印出。

參考解答：

```
scores = [80, 90, 75, 88]

avg = sum(scores) / len(scores)
print(avg)
```

**練習題**

1. 給一個字典 `student = {"name": "Tom", "scores": [70, 85, 90]}`，印出名字與平均分數。
2. 給兩個集合 `a={1,2,3,4}`、`b={3,4,5}`，印出交集、聯集、差集（`a-b`）。

---

## 5.6 模組

**說明**

模組就是「把程式碼整理在檔案裡，讓別人/自己可以重複使用」。

- 內建模組（不用安裝）：`math`, `random`, `datetime`, `pathlib`…
- 第三方模組（需要安裝）：`requests`, `pandas`, `openpyxl`…

常見用法：

```
import math
print(math.sqrt(9))

from datetime import datetime
print(datetime.now())
```

**範例題目：隨機抽籤**

題目：

用 `random` 從名單中隨機抽一個人。

參考解答：

```
import random

names = ["Amy", "Bob", "Cindy", "David"]
winner = random.choice(names)
print(winner)
```

**練習題**

1. 用 `math` 計算 `sqrt(2)` 並印出小數點後 4 位。
2. 用 `datetime` 印出今天日期（年-月-日）。

### 5.7 字串處理

**說明**

字串（`str`）非常常用，常見操作：

- 取長度：`len(s)`
- 切片：`s[start:end]`
- 分割：`s.split(",")`
- 取代：`s.replace("a", "b")`
- 去空白：`s.strip()`
- 轉大小寫：`s.lower()` / `s.upper()`
- f-string 格式化：`f"...{var}..."`

**範例題目：Email 簡單檢查**

題目：

給定字串 `email`，檢查是否同時包含 `@` 與 `.`，如果是印出 `OK`，否則印出 `Invalid`。

參考解答：

```
email = "test@example.com"

if "@" in email and "." in email:
    print("OK")
else:
    print("Invalid")
```

**練習題**

1. 給定 `s = "  Hello, Python  "`，去除前後空白後轉成小寫並印出。
2. 給定 `csv_line = "Tom,80,90,70"`，用 `split` 解析後印出名字與三科平均。

---

## 6. Exception處理

### 說明

程式執行時可能出錯，例如：

- 使用者輸入不是數字
- 檔案不存在
- 除以 0

Exception（例外）處理讓程式「出錯時不會直接崩潰」，你可以：

- 顯示友善訊息

- 做補救處理
- 或記錄錯誤

基本語法：

```
try:
    # 可能出錯的程式
except 某種錯誤:
    # 發生該錯誤時怎麼做
else:
    # 沒出錯才會跑
finally:
    # 不管有沒有出錯都會跑（常用於釋放資源）
```

### 範例題目：安全的數字除法

題目：

讓使用者輸入 `a` 與 `b`，印出 `a/b`。若輸入不是數字或`b=0`，要印出友善訊息。

參考解答：

```
try:
    a = float(input("請輸入 a: "))
    b = float(input("請輸入 b: "))
    result = a / b
except ValueError:
    print("輸入錯誤：請輸入數字")
except ZeroDivisionError:
    print("不能除以 0")
else:
    print(f"結果：{result}")
```

### 練習題

1. 讀取檔案 `data.txt` 並印出內容；若檔案不存在，要印出 找不到檔案。
2. 寫一段程式把使用者輸入轉成整數；若失敗要一直請他重輸，直到成功為止（提示：`while True` + `try/except`）。

---

## 7. Excel檔案操作

### 說明

Python 操作 Excel 常見方式：

1. **openpyxl**：讀寫 `.xlsx`（最常見、入門友善）
2. **pandas**：資料分析很強，讀寫 Excel 也方便（底層常搭配 openpyxl）

安裝（在啟動虛擬環境後）：

```
python -m pip install openpyxl pandas
```

### 範例題目：建立一個簡單的成績表 Excel

題目：

用 `openpyxl` 建立 `scores.xlsx`，內容如下：

- A1:B1 為標題：`Name`, `Score`
- 從第 2 列開始寫入三筆資料：`Tom 80`, `Amy 92`, `Bob 75`
- 最後再寫一列 `Average` 與平均分數

參考解答：

```
from openpyxl import Workbook

wb = Workbook()
ws = wb.active
ws.title = "Scores"

ws.append(["Name", "Score"])
rows = [
    ["Tom", 80],
    ["Amy", 92],
    ["Bob", 75],
]

for r in rows:
    ws.append(r)

avg = sum(score for _, score in rows) / len(rows)
ws.append(["Average", avg])

wb.save("scores.xlsx")
print("已輸出 scores.xlsx")
```

### 練習題

1. 讀取 `scores.xlsx`，把所有分數加總並印出總分與平均。
2. 用 `pandas` 讀取 Excel，新增一欄 `Pass`（分數 >= 60 為 True），再輸出成 `scores_with_pass.xlsx`。

---

## 8. 物件導向程式設計基礎

### 8.1 類別與物件

**說明**

物件導向（OOP）把程式世界想成由「物件」組成。

- **類別（class）**：物件的藍圖（定義有哪些資料、有哪些行為）
- **物件（object / instance）**：照著藍圖做出來的實體

類別裡常見：

- 屬性（attributes）：這個物件「有什麼資料」
- 方法（methods）：這個物件「會做什麼事」

**範例題目：建立 `Student` 類別**

題目：

建立 `Student`：

- 屬性：`name`、`scores`（串列）
- 方法：`average()` 回傳平均分數

建立兩個學生並印出平均。

參考解答：

```
class Student:
    def __init__(self, name: str, scores: list[int]):
        self.name = name
        self.scores = scores

    def average(self) -> float:
        return sum(self.scores) / len(self.scores)


s1 = Student("Tom", [80, 70, 90])
s2 = Student("Amy", [95, 88, 92])

print(s1.name, s1.average())
print(s2.name, s2.average())
```

**練習題**

1. 寫一個 `Rectangle` 類別，有 `width`、`height`，提供 `area()` 與 `perimeter()`。
2. 寫一個 `BankAccount` 類別，有 `balance`，提供 `deposit(amount)`、`withdraw(amount)`（餘額不足要提示）。

---

## 8.2 繼承與多型

**說明**

**繼承（Inheritance）**：用現有類別當基底，延伸出更具體的類別。

- 父類別（base / parent）提供共用功能
- 子類別（subclass / child）可以新增或覆寫（override）方法

**多型（Polymorphism）**：不同類別的物件可以用「相同的方法名稱」被呼叫，表現出各自的行為。

**範例題目：不同員工計算薪資**

題目：

建立 `Employee` 父類別與兩個子類別：

- `FullTimeEmployee`：月薪固定
- `PartTimeEmployee`：時薪 * 工時

都提供 `pay()` 回傳薪資，並用同一段程式走訪清單印出每個人的薪資（多型）。

參考解答：

```
class Employee:
    def __init__(self, name: str):
        self.name = name

    def pay(self) -> float:
        raise NotImplementedError()


class FullTimeEmployee(Employee):
    def __init__(self, name: str, monthly_salary: float):
        super().__init__(name)
        self.monthly_salary = monthly_salary

    def pay(self) -> float:
        return self.monthly_salary


class PartTimeEmployee(Employee):
    def __init__(self, name: str, hourly_wage: float, hours: float):
        super().__init__(name)
        self.hourly_wage = hourly_wage
        self.hours = hours

    def pay(self) -> float:
        return self.hourly_wage * self.hours


employees = [
    FullTimeEmployee("Amy", 52000),
    PartTimeEmployee("Bob", 220, 60),
]

for e in employees:
    print(e.name, e.pay())
```

**練習題**

1. 設計 `Animal` 父類別與 `Dog`、`Cat` 子類別，各自覆寫 `speak()`，走訪清單印出叫聲。
2. 設計 `Shape` 父類別與 `Circle`、`Rectangle` 子類別，提供 `area()`；用多型計算所有形狀面積總和。

---

# 附錄：初學者常見問題

## Q1：為什麼我的縮排會報錯？

Python 用「縮排」表示程式區塊，同一層級必須對齊。建議使用 4 個空格，不要混用 Tab。

## Q2：我輸入中文路徑/檔名會不會有問題？

大多數情況沒問題，但遇到套件或系統設定差異時可能出現編碼問題。初學時可以先用英文檔名，等熟悉後再處理中文情境。