

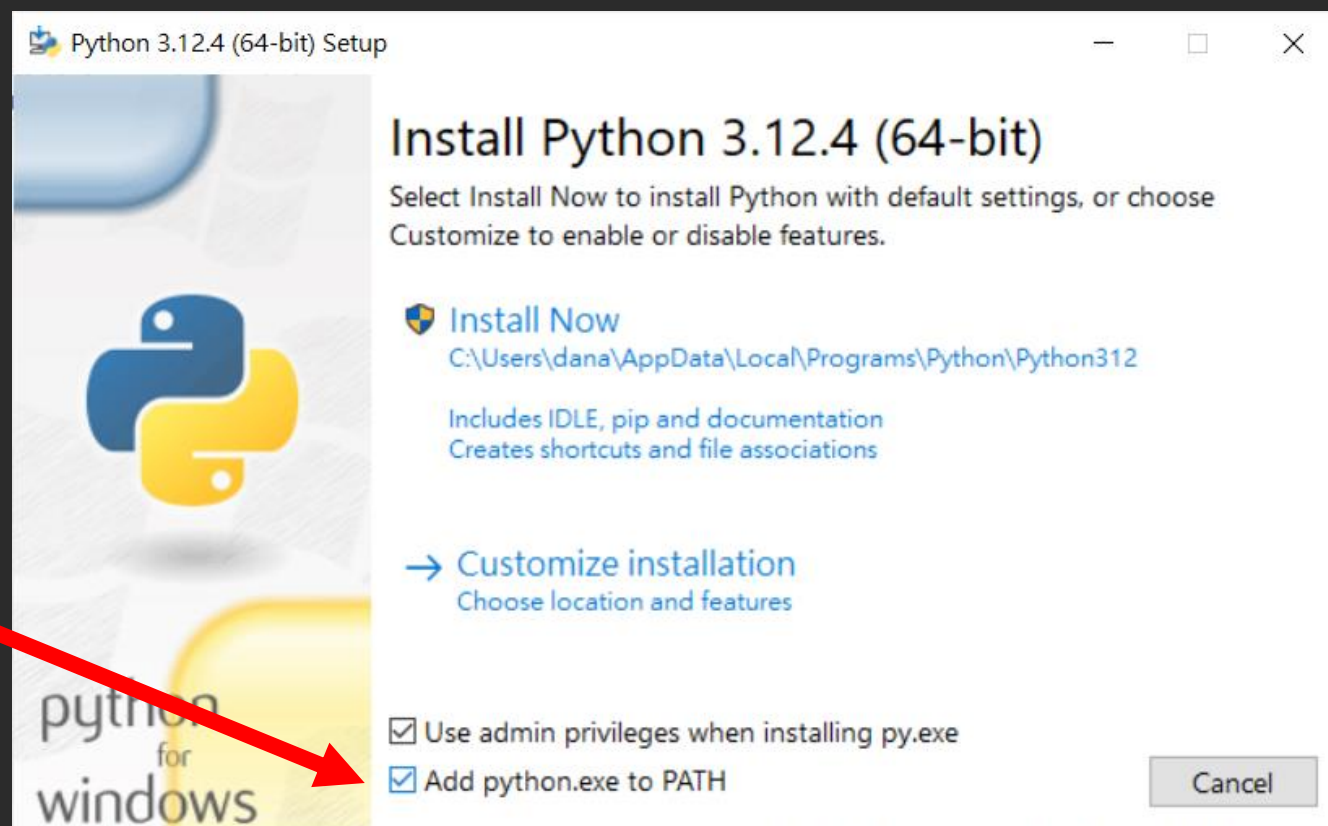
Python 基礎教學

什麼是程式語言

- - 軟體工程師和電腦溝通的橋樑
- - 用來命令電腦做出我們想要他做的事
- - Python是一種簡單易學的程式語言

安裝環境

安裝Python，記得下面的選項要勾起來



去官網下載Visual Studio Code (<https://code.visualstudio.com/>)



Visual Studio Code

Docs

Updates

Blog

API

Extensions

FAQ

Learn



Search Docs



Download

[Version 1.90](#) is now available! Read about the new features and fixes from May.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

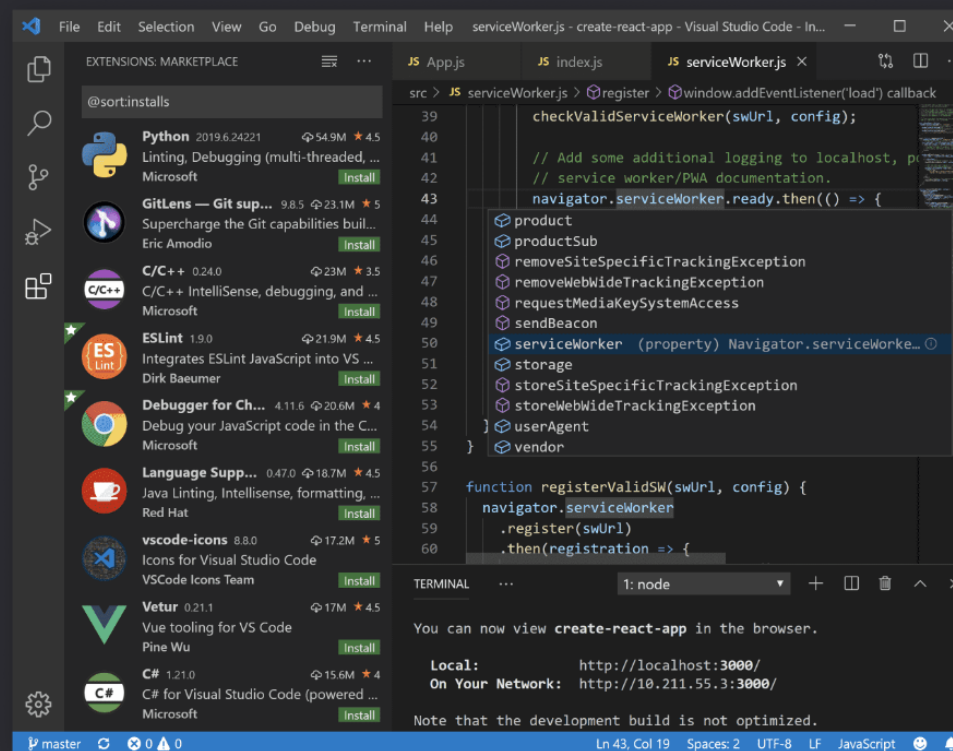
Download for Windows

Stable Build

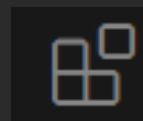


[Web](#), [Insiders edition](#), or [other platforms](#)

By using VS Code, you agree to its
[license and privacy statement](#).



在左邊的Extensions欄位中搜尋Python，安裝延伸模組



檔案(F) 編輯(E) 選取項目(S) 檢視(V) 移至(G) ...

python

python

Python 635ms
Python language supp...
Microsoft

Python 256K
Extensions for Python
shiro 安裝

Python Deb... 1723ms
Python Debugger exte...
Microsoft

Python Indent 9.8M
Correct Python indent...
Kevin Rose 安裝

Python Exte... 8.6M
Popular Visual Studio ...
Don Jayamanne 安裝

Python Envir... 8.2M
View and manage Pyth...
Don Jayamanne 安裝

Python for V... 5.5M
Python language exten...
Thomas Ha... 安裝

autoDocstrin... 9.6M
Generates python docs...
Nils Werner 安裝

Python Previ... 1.6M
Provide Preview for Pyt...
dongli 安裝

Python Exte... 1.4M
Python Extended is a v...

Python v2024.8.1

Microsoft microsoft.com 127,167,119 594

Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatting, refacto...

停用 解除安裝 切換到發行前版本

詳細資料 功能 變更記錄 延伸模組套件

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported Python versions), providing access points for extensions to seamlessly integrate and offer support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more!

Support for `vscode.dev`

The Python extension does offer some support when running on `vscode.dev` (which includes `github.dev`). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the following extensions by default to provide the best Python development experience in VS Code:

- Pylance - to provide performant Python language support
- Python Debugger - to provide a seamless debug experience with debugpy

These extensions are optional dependencies, meaning the Python extension will remain fully functional if they fail to be installed. Any or all of these extensions can be disabled or uninstalled at the expense of some features. Extensions installed through the marketplace are subject to the [Marketplace Terms of Use](#).

Extensibility

類別

Programming Languages Debuggers

Other Data Science

Machine Learning

資源

市集

問題

儲存庫

授權

Microsoft

其他資訊

已發佈	2016-01-19, 23:03:11
上次發行	2024-06-25, 18:44:03
上次更新時間	2024-06-25, 09:00:52
識別碼	ms-python.python

Codeium: ()

你還可以搜尋Chinese，安裝中文
裝完後右下角會跳出通知，要求你重開VSCode

The screenshot shows the Visual Studio Code interface with the 'Extension: Chinese (Traditional) Language Pack for Visual Studio Code' page open. The left sidebar shows a list of extensions, with 'Chinese (Traditional) Language Pack' selected. The main panel displays the extension's details, including its icon, name, publisher (Microsoft), version (v1.91.2024061909), and a list of features. The 'Features' section is expanded, showing the 'VS Code 的中文 (繁體) 語言套件' (Chinese (Traditional) Language Pack for VS Code) and its usage instructions. The right sidebar shows 'Categories' (Language Packs), 'Resources' (Marketplace, Issues, Repository, Microsoft), and 'More Info' (Published, Last released, Identifier).

File Edit Selection View Go Run Terminal Help python

EXTENSIONS: ... chinese

Chinese (Traditional) Language Pack for Visual Studio Code v1.91.2024061909

Microsoft microsoft.com 2,909,797 ★★★★★ (28)

Language pack extension for Chinese (Traditional)

Install

DETAILS FEATURES CHANGELOG

VS Code 的中文 (繁體) 語言套件

中文 (繁體) 語言套件，讓 VS Code 提供本地化的使用者介面。

使用方式

您可以使用「設定顯示語言」命令，以明確的方式設定 VS Code 顯示語言，以覆寫預設的 UI 語言。按下「Ctrl+Shift+P」以顯示「命令選擇區」，然後開始鍵入「display」以篩選及顯示「設定顯示語言」命令。按下「確定」後會顯示已安裝的語言清單 (依據地區設定)，目前的地區設定會有醒目提示。選取其他「地區設定」以切換 UI 語言。詳細步驟請參考[文件](#)。

參與貢獻

若要對翻譯提出改善意見反應，請在 `vscode-loc` 存放庫中建立問題。翻譯字串會在 Microsoft 當地語系化平台中維護。只有在 Microsoft 當地語系化平台中才能進行變更，並匯出至 `vscode-loc` 存放庫。因此，`vscode-loc` 存放庫中不接受提取要求。

授權

原始碼以及文字屬於 MIT 授權。

感謝

中文 (繁體) 的語言套件是「從社群，到社群」的本地化社群努力的結果。

特別感謝社群的貢獻者們讓本地化變得可能。

Categories: Language Packs



Resources: Marketplace, Issues, Repository, Microsoft

More Info: Published: 2018-05-09, 07:07:20; Last released: 2024-06-19, 17:27:07; Identifier: ms-ceintl1.vscodelanguage-pack-zh-hant

Codeium: {...}

你可能需要設定一些東西，左下角齒輪圖案開啟設定

Auto Save，設成afterDelay，自動儲存

 設定 

auto save

使用者 工作區

經常使用的 (1)

▼ 文字編輯器 (5)

檔案 (5)

▼ 延伸模組 (2)

▼ C/C++ (1)

Code Analysis (1)


Python (1)

Files: Auto Save
控制具有未儲存變更之編輯器的 [自動儲存](#)。

afterDelay ▼

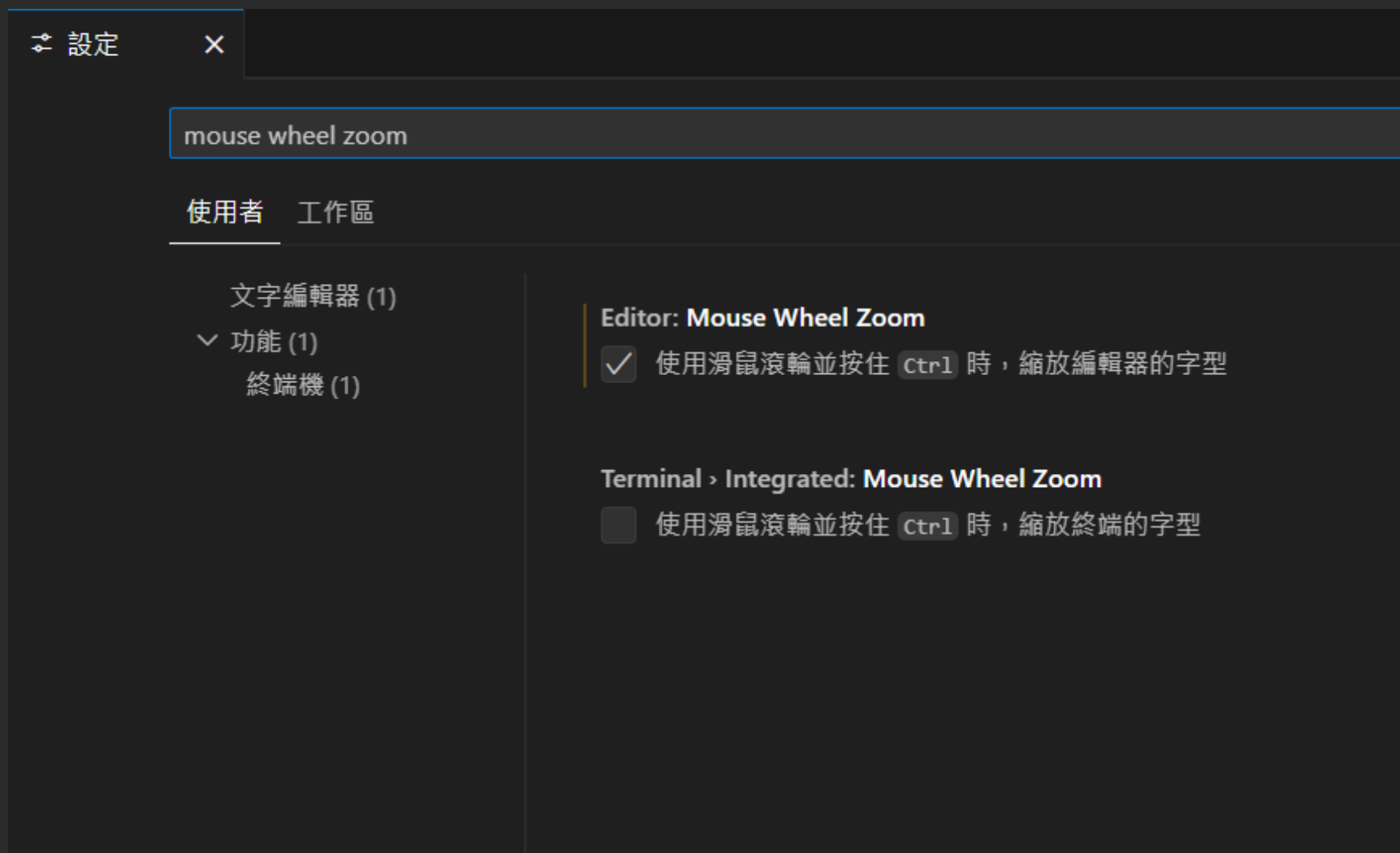
Files: Auto Save Delay
控制具有未儲存變更之編輯器自動儲存後的延遲 (毫秒)。只有在 [Files: Auto Save](#) 設定為 `afterDelay` 時才適用。

1000

 **Files: Auto Save When No Errors**
☐ 啟用時，會限制 [自動儲存](#) 編輯者在觸發自動儲存時沒有報告任何錯誤的檔案。只有在啟用 [Files: Auto Save](#) 時才適用。

Files: Auto Save Workspace Files Only

Mouse Wheel Zoom，開啟，按住Ctrl加滑鼠滾輪可以縮放字體大小



基礎語法

變數 Variable

- - 用來存資料 裡面的資料可以隨時改變
- - 可以自己命名 良好的命名習慣會對你有很大的幫助
- - 程式會透過變數名稱來找尋裡面的資料
- - 分為多種形態 在Python中主要有
Integer(整數)、float(浮點數)、string(字串)、Boolean(布林值)

資料型態

int

5

34

108

float

1.2

3.14

67.83

string

"Apple"

"cat123"

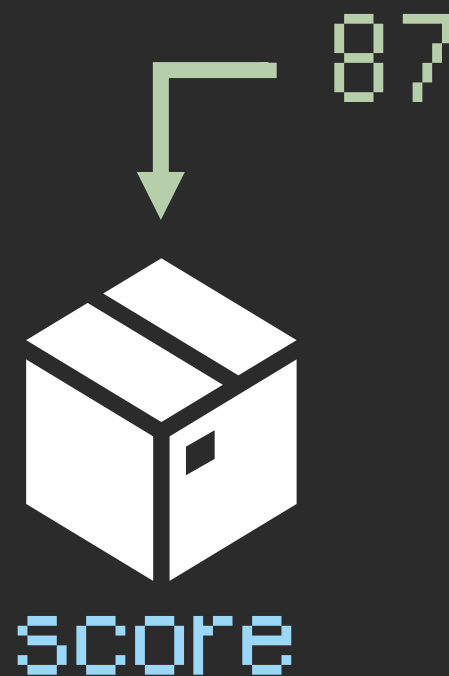
bool

True

False

變數名稱 = 變數值

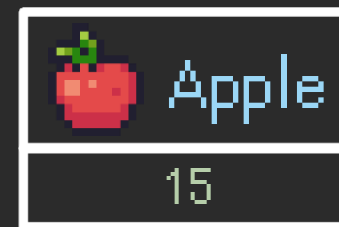
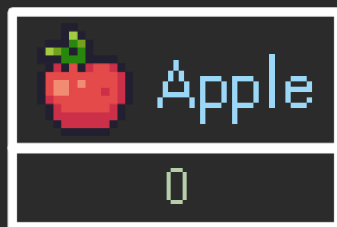
```
1 score = 87  
2 bmi = 20.5  
3 name = "Bob"  
4 isMale = True
```



賦值

- - 你當然可以更改變數的值
- - 將資料存進變數裡面，這個行為稱為「賦值」

```
1 Apple = 0  
2 Apple = 15
```

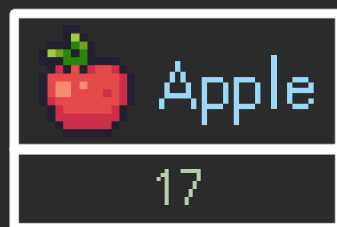
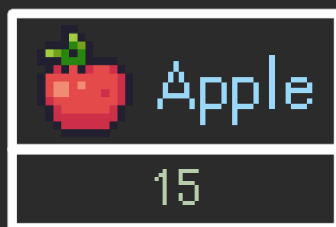


運算式

- - 你可以對資料做數學運算

```
1 Apple = 15  
2 Apple = Apple + 2
```

先把「Apple」中原本的資料取出來，加2之後再存進「Apple」中。



· 基本的運算有這幾種

運算子	意義	範例	範例結果
+	兩個數字相加	$15 + 1$	16
-	兩個數字相減	$15 - 2$	13
*	兩個數字相乘	$15 * 2$	30
/	兩個數字相除	$32 / 5$	6.4
%	取得餘數	$32 \% 5$	2
//	取得整除的商數	$32 // 5$	6
**	次方	$7 ** 2$	49

複合指定運算子

如果進行運算的對象就是該變數本身，則還有另一種寫法可以直接完成運算與賦值。

```
Apple += 11  #等同於 Apple = Apple + 11
```

```
Apple -= 23  #等同於 Apple = Apple - 23
```

```
Apple *= 2   #等同於 Apple = Apple * 2
```

```
Apple /= 10  #等同於 Apple = Apple / 10
```

```
Apple %= 2   #等同於 Apple = Apple % 2
```


字串的加法

- 字串型別也可以進行加法，將兩個字串給串接起來。

```
1 NamePart1 = "Sponge"  
2 NamePart2 = "Bob"  
3 Name = NamePart1 + NamePart2
```



輸出與輸入

- - 我們可以透過使用輸入函數來把外部資料存進變數。
- - 也可以透過使用輸出函數來將結果顯示在螢幕上。

- - 使用「print」函數可以把資料輸出。

```
print("Hello, world!")
```

```
Hello, world!
```

- - 可以使用逗點把資料在同一行依序輸出，會用一個空白(" ")做分隔。

```
Name = "Bob"
```

```
Age = 12
```

```
print("我的名字是", Name, "，我今年", Age, "歲。")
```

```
我的名字是 Bob ，我今年 12 歲。
```

- - print 會在輸出資料後輸出結尾字串，預設是換行
- - 可以用 end 來設定結尾要輸出的字串

```
1 print(100, "多吃水果", 60)  
2 print(100, "多吃水果", 60, end = ".")
```

```
100 多吃水果 60  
100 多吃水果 60.
```

- - 使用「input」函數可以把使用者輸入的一行資料讀取進來。

```
Name = input("請輸入您的大名：")  
print("你好，", Name, "！")
```

```
請輸入您的大名：Dio  
你好， Dio！
```

- - 輸入的資料是字串，需要轉換成數值資料才可以做數學運算

```
Age = int(input("請輸入您的年紀："))  
print("明年你就", Age + 1, "歲了!")
```

```
請輸入您的年紀：39  
明年你就 40 歲了！
```

- 以下是基本的幾個轉換函數

函數名稱	意義	範例用法	範例結果
int	將資料轉換成整數型態	int("29")	29
float	將資料型態轉換成浮點數	float("37.7")	37.7
bool	將資料型態轉換成布林值	bool(0)	False
str	將資料型態轉換成字串	str(1123)	"1123"

練習時間

比較運算

- – 比較運算子可以讓資料進行比對，得知相對大小關係。
- – 比較結果正確就回傳True，錯誤就回傳False

```
print("5.7 > 2.3 is", 5.7 > 2.3)  
print("37 < 12 is", 37 < 12)
```

```
5.7 > 2.3 is True  
37 < 12 is False
```


- 以下是幾種基本的比較運算子

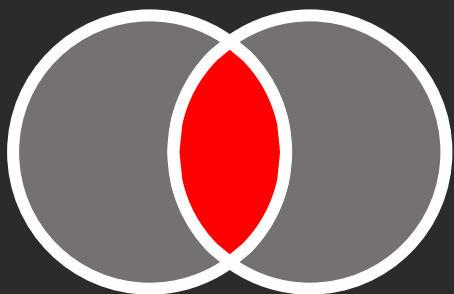
比較運算子	意義	範例	範例結果
>	判斷左邊的資料是否大於右邊的資料	3 > 2	True
<	判斷左邊的資料是否小於右邊的資料	25 < 25	False
>=	判斷左邊的資料是否大於或等於右邊的資料	78 >= 77	True
<=	判斷左邊的資料是否小於或等於右邊的資料	66 <= 66	True
==	判斷左邊的資料是否等於右邊的資料	97.1 == 97.2	False
!=	判斷左邊的資料是否不等於右邊的資料	"YEE" != "YEE"	False

邏輯運算

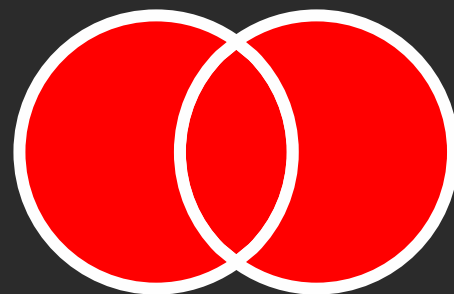
- - 利用邏輯運算子可以將多種不同的條件合併成一個結果，用於複雜的比較條件

運算子	意義	範例	範例結果
not	傳回相反的結果 True轉成False，False轉成True	not(True) not(False)	False True

運算子	意義	範例	範例結果
and	兩個條件都是True時才傳回True， 其餘情況都傳回False	$(5 > 3) \text{ and } (9 > 6)$ $(5 < 3) \text{ and } (9 > 6)$ $(5 < 3) \text{ and } (9 < 6)$	True False False
or	兩個條件都是False時才傳回False， 其餘情況都傳回True	$(5 > 3) \text{ or } (9 > 6)$ $(5 < 3) \text{ or } (9 > 6)$ $(5 < 3) \text{ or } (9 < 6)$	True True False



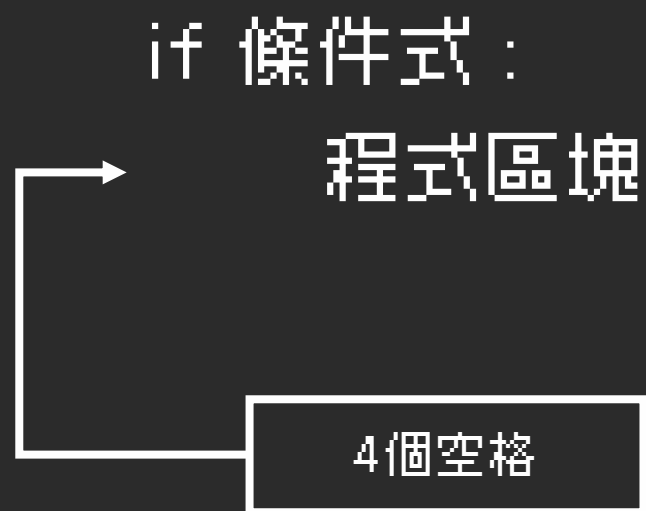
and



or

判斷式

- - 在 if 判斷式中，當給定的條件成立時，便會執行指定的程式碼。
- - Python 會根據縮排來判斷 if 的有效範圍，所以縮排必須要正確。



- – if 判斷式也可以有多個分支，也就是「if – elif – else」判斷式。
- – 當上一個判斷式沒有觸發就會去下一個檢查。

```
if 條件式：  
    程式區塊  
elif 條件式：  
    程式區塊  
    ⋮  
else：  
    程式區塊
```

練習時間

串列 List

- - 串列是一種能儲存多個資料的結構。
- - 串列在賦值時可以用中括號將所有內容物(元素)裝進去。

串列名稱 = [元素1, 元素2,]

```
Fruit = ["Apple", "Banana", "Cherry", "Orange"]    # 擁有的水果  
Vegetable = []                                     # 擁有的蔬菜
```

Fruit
"Apple"
"Banana"
"Cherry"
"Orange"

Vegetable

- - 可以透過在中括號內放入「索引值」來存取串列中的元素。
- - 第一個元素的索引值是 0；第二個是 1；第三個是 2，依此類推。

串列名稱 [索引值]

```
Fruit = ["Apple", "Banana", "Cherry", "Orange"] # 擁有的水果  
Fruit[1] = "Kiwi" # 把Fruit中索引值為1的元素改成奇異果
```

Fruit	
0	"Apple"
1	"Banana"
2	"Cherry"
3	"Orange"



Fruit	
0	"Apple"
1	"Kiwi"
2	"Cherry"
3	"Orange"

- - 索引值可以是負數，代表要取「倒數第幾個元素」。

```
Fruit = ["Apple", "Banana", "Cherry", "Orange"]    # 擁有的水果
Fruit[1] = "Kiwi"                                # 把Fruit中索引值為1的元素改成奇異果
Fruit[-2] = "Banana"                             # 把Fruit中倒數第二項元素改成香蕉
```

Fruit	
0	"Apple"
1	"Banana"
2	"Cherry"
3	"Orange"



Fruit	
0	"Apple"
1	"Kiwi"
2	"Cherry"
3	"Orange"



Fruit	
0	"Apple"
1	"Kiwi"
2	"Banana"
3	"Orange"

- - 使用「len」函數可以計算串列的長度（裡面有幾個元素）。

```
Fruit = ["Apple", "Banana", "Cherry", "Orange"]    # 擁有的水果  
Length = len(Fruit)                                # 計算水果清單的長度並存在變數裡
```

Fruit	
0	"Apple"
1	"Banana"
2	"Cherry"
3	"Orange"

 Length	
4	

- - 可以使用「insert」函數來將新元素插入到串列中。
- - 也可以使用「append」函數來將新元素加到串列最後一項。

```
Fruit = ["Apple", "Banana"]    # 擁有的水果  
Fruit.append("Orange")        # 新增橘子到最後一項  
Fruit.insert(2, "Cherry")     # 將櫻桃插入到索引值為2的位置
```

Fruit	
0	"Apple"
1	"Banana"



Fruit	
0	"Apple"
1	"Banana"
2	"Orange"



Fruit	
0	"Apple"
1	"Banana"
2	"Cherry"
3	"Orange"

- - 可以使用「remove」函數來移除特定的元素。
- - 也可以使用「del」移除指定的元素。

```
Fruit = ["Apple", "Banana", "Cherry", "Orange"] # 擁有的水果  
Fruit.remove("Cherry") # 移除櫻桃  
del(Fruit[2]) # 移除索引值為2的元素
```

Fruit	
0	"Apple"
1	"Banana"
2	"Cherry"
3	"Orange"



Fruit	
0	"Apple"
1	"Banana"
2	"Orange"



Fruit	
0	"Apple"
1	"Banana"

- - 使用「sort」函數可以將串列中的元素由小到大排序。
- - 使用「reverse」函數可以將串列中的元素排序顛倒過來。

```
NumberList = [3, 11, 9, 7]  
NumberList.sort()  
NumberList.reverse()
```

NumberList	
0	3
1	11
2	9
3	7



NumberList	
0	3
1	7
2	9
3	11

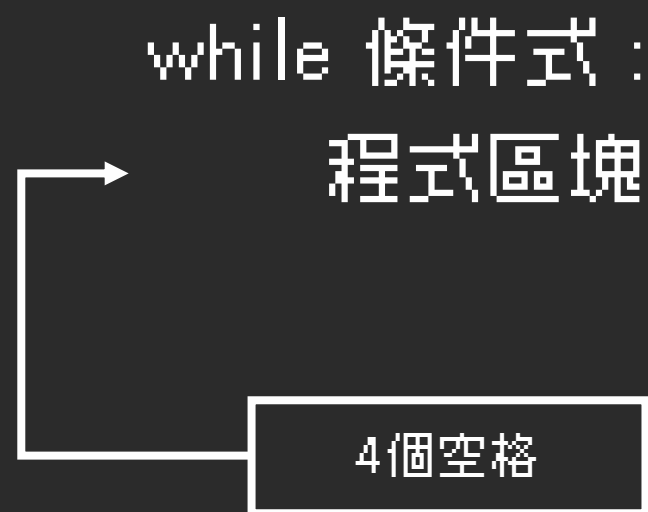


NumberList	
0	11
1	9
2	7
3	3

練習時間

迴圈

- - while 迴圈會不斷執行指定程式碼，直到給定的條件不成立為止。
- - Python 會根據縮排來判斷 while 的有效範圍，所以縮排必須要正確。



- - for 迴圈可以直接遍歷串列，針對裡面每一個元素執行一次指定程式碼。
- - 執行for迴圈時，系統會將串列的元素依序作為變數的值，設定完後執行程式區塊一次
- - Python 會根據縮排來判斷 for 的有效範圍，所以縮排必須要正確。

```
for 變數 in 串列：  
    程式區塊
```

- – 利用「range」函數，可以產生一個可迭代的整數循序數列。
- – for 迴圈也可以搭配「range」函數來遍歷某個範圍的數字，常用於索引值遍歷。

range (整數值)

range (起始值, 終止值)

range (起始值, 終止值, 間隔值)

```
range(5)           # 0, 1, 2, 3, 4
range(2, 5)        # 2, 3, 4
range(1, 10, 2)     # 1, 3, 5, 7, 9
range(10, 0, -2)    # 10, 8, 6, 4, 2
```

Break 命令

- 迴圈執行時，如果要在中途結束迴圈執行，可以用 break 命令強制離開迴圈

```
1 for i in range(10):  
2     if (i == 6):  
3         break  
4     print(i, end = ", ")
```

0, 1, 2, 3, 4, 5,

continue 命令

- - continue 命令是在迴圈執行中途暫時停住不往下執行，而跳到迴圈起始處繼續執行

```
1 for i in range(10):  
2     if (i == 6):  
3         continue  
4     print(i, end = ", ")
```

0, 1, 2, 3, 4, 5, 7, 8, 9,

練習時間

自定義函數 (Function)

- - 我們可以把一套程式碼打包、定義成函數，方便後續的使用。

```
# Height: 身高
# Weight: 體重
def ComputeBMI(Height, Weight):
    H_Square = (Height ** 2)          # 計算身高的平方
    BMI_Value = Weight / H_Square     # BMI值是 體重÷(身高×身高)

    return BMI_Value                 # 將計算結果回傳

H = float(input("請輸入您的身高(公尺): "))
W = float(input("請輸入您的體重(公斤): "))

print("您的BMI值為 :", ComputeBMI(H, W))
```

```
請輸入您的身高(公尺): 1.72
請輸入您的體重(公斤): 69
您的BMI值為 : 23.323418063818284
```

- - 函數不一定需要回傳值。
- - 無論這個函數需不需要額外的資料(參數)，都必須要有小括號(參數區)。

```
def Print_A_To_F():  
    AlphaList = ['A', 'B', 'C', 'D', 'E', 'F']  
  
    for Alp in AlphaList:  
        print("字母", Alp)  
  
Print_A_To_F()
```

```
字母 A  
字母 B  
字母 C  
字母 D  
字母 E  
字母 F
```

字典 Dictionary

- - 元素以「鍵 - 值」的方式儲存，利用「鍵」來取得「值」
- - 鍵不能重複，值可以重複

價格表		
"香蕉"	→	20
"蘋果"	→	50
"橘子"	→	30

字典名稱 = { 鍵1 : 值1, 鍵2 : 值2, }

```
1 fruitPrice = {"香蕉" : 20, "蘋果" : 50, "橘子" : 30}  
2 print(fruitPrice["蘋果"])
```

50

修改字典

```
1 fruitPrice = {"香蕉" : 20, "蘋果" : 50, "橘子" : 30}
2 fruitPrice["香蕉"] = 100 #修改
3 fruitPrice["鳳梨"] = 60  #新增
4
5 print("香蕉價格:", fruitPrice["香蕉"])
6 print("鳳梨價格:", fruitPrice["鳳梨"])
```

```
香蕉價格: 100
鳳梨價格: 60
```

删除字典

```
1 fruitPrice = {"香蕉" : 20, "蘋果" : 50, "橘子" : 30}
2 del fruitPrice["橘子"] #删除特定元素
3 print(fruitPrice)
4 fruitPrice.clear()      #清空字典
5 print(fruitPrice)
```

```
{'香蕉': 20, '蘋果': 50}
{}
```

檢查鍵是否存在

鍵 in 字典名稱

```
1 fruitPrice = {"香蕉": 20, "蘋果": 50, "橘子": 30}
2 if ("香蕉" in fruitPrice):
3     print("有賣香蕉")
4 else:
5     print("沒賣香蕉")
```

有賣香蕉

```
1 fruitPrice = {"香蕉": 20, "蘋果": 50, "橘子": 30}
2 if ("草莓" in fruitPrice):
3     print("有賣草莓")
4 else:
5     print("沒賣草莓")
```

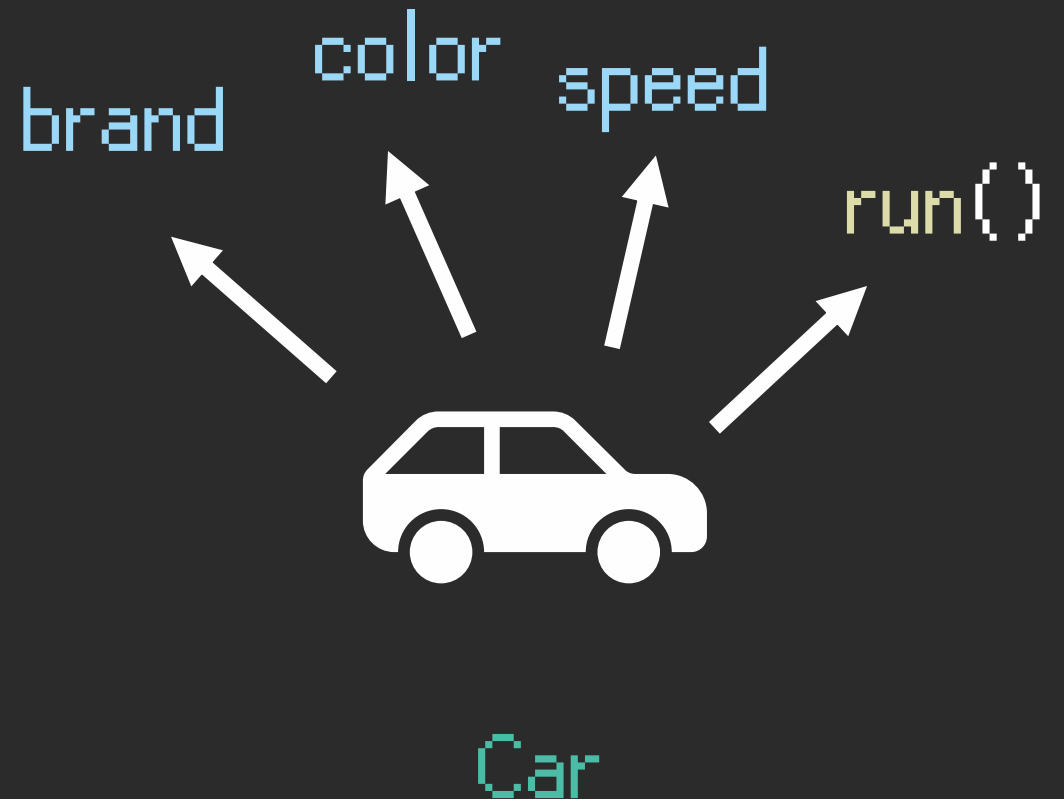
沒賣草莓

練習時間

類別 class

- - 自己定義的資料型別
- - 把很多變數和函式包在一起

```
1 class Car:
2     def __init__(self):
3         self.brand = "Toyota"
4         self.color = "White"
5         self.speed = 0
6     def run(self):
7         self.speed = 100
```



傳入參數給建構函式

```
1 class Car:
2     def __init__(self, brand, color, speed):
3         self.brand = brand
4         self.color = color
5         self.speed = speed
6     def run(self):
7         self.speed = 100
8
9 car = Car("BMW", "Black", 80)
10 print("牌子:", car.brand)
11 print("顏色:", car.color)
12 print("速度:", car.speed)
```

牌子: BMW
顏色: Black
速度: 80