

# 零基礎上手! Python 實戰班



黃熠程 Miller

- 2020 中央大學大氣物理學博士候選人
- 2018-2021 安吉氣象資訊 – 專案經理
- 2021-2022 和碩集團 – 軟體工程師
- 2022-now 陽明海運 – 網站全端工程師

## 程式語言

- C/C++, C#, Fortran, Python, MATLAB, JavaScript, HTML5/CSS

## 專長

- 數值模擬, 深度學習, 軟體開發, 網站開發

# 課程簡介

Python 簡介

變數

如何開始寫程式

如何寫程式

如何用程式

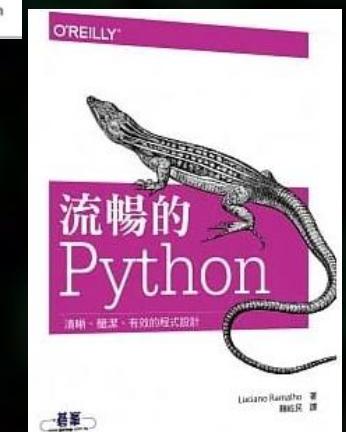
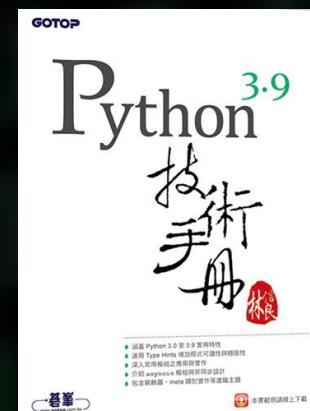
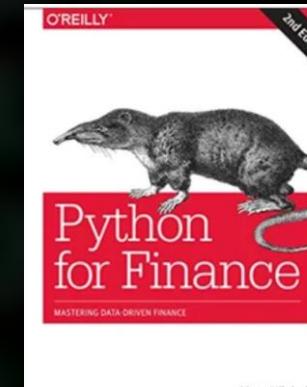
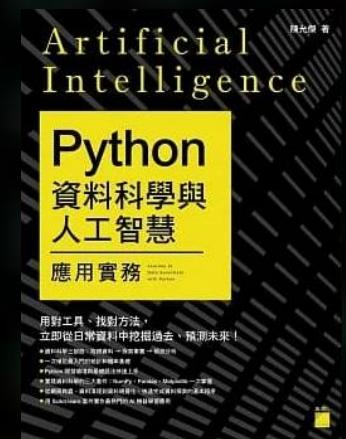
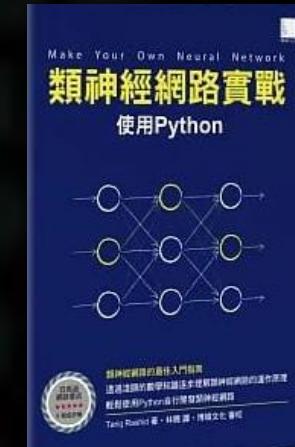
如何寫好程式



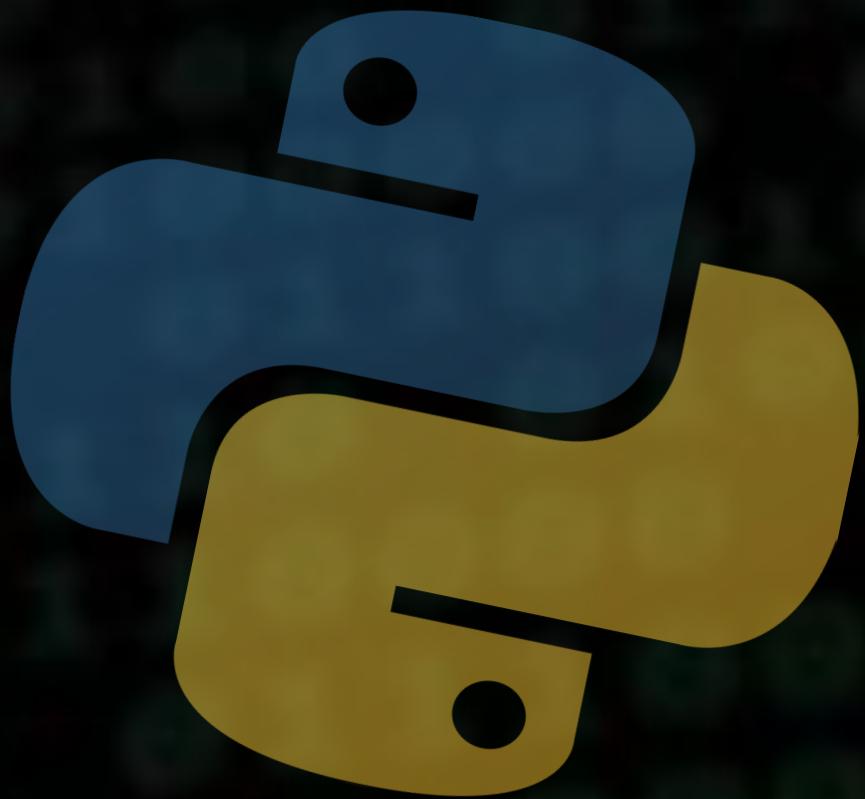
邏輯處理

# 參考與延伸閱讀

- Python 資料科學與人工智慧 應用實務
- 類神經網路實戰：使用Python
- [Python For Finance](#)
- [Python 3.9技術手冊](#)
- 流暢的 Python：清晰、簡潔、有效的程式設計

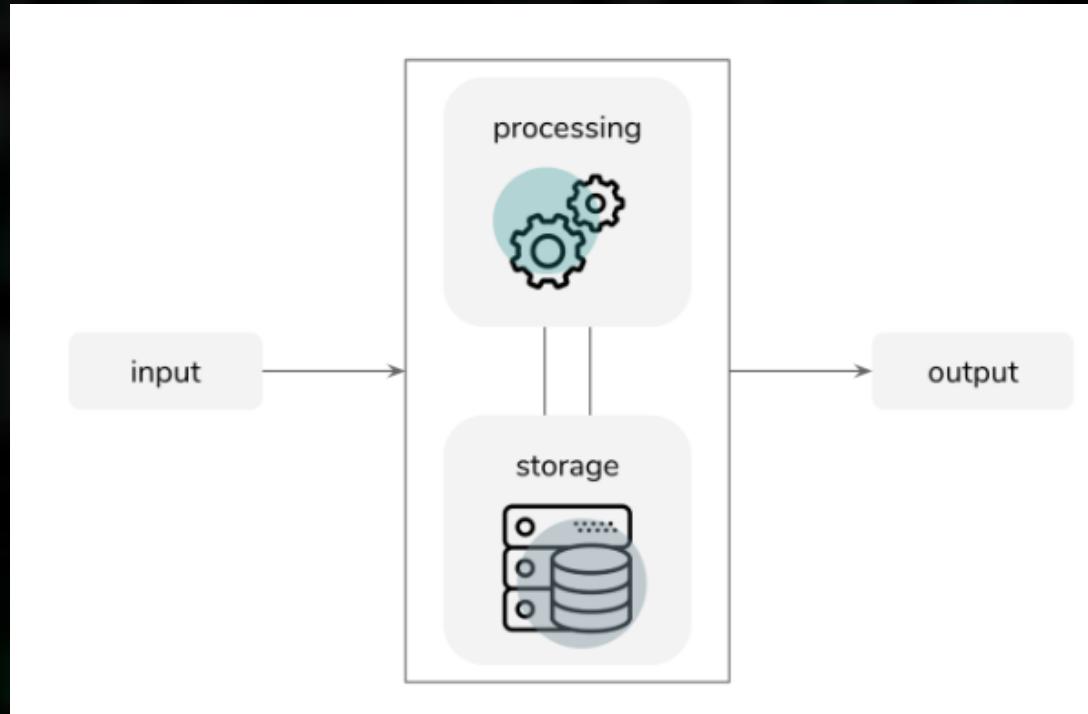


# 程式設計



# 程式是什麼

Ans: 用來控制電腦的指令



從我們生活的玩樂到商業只要有用到電腦運算的叫做 程式

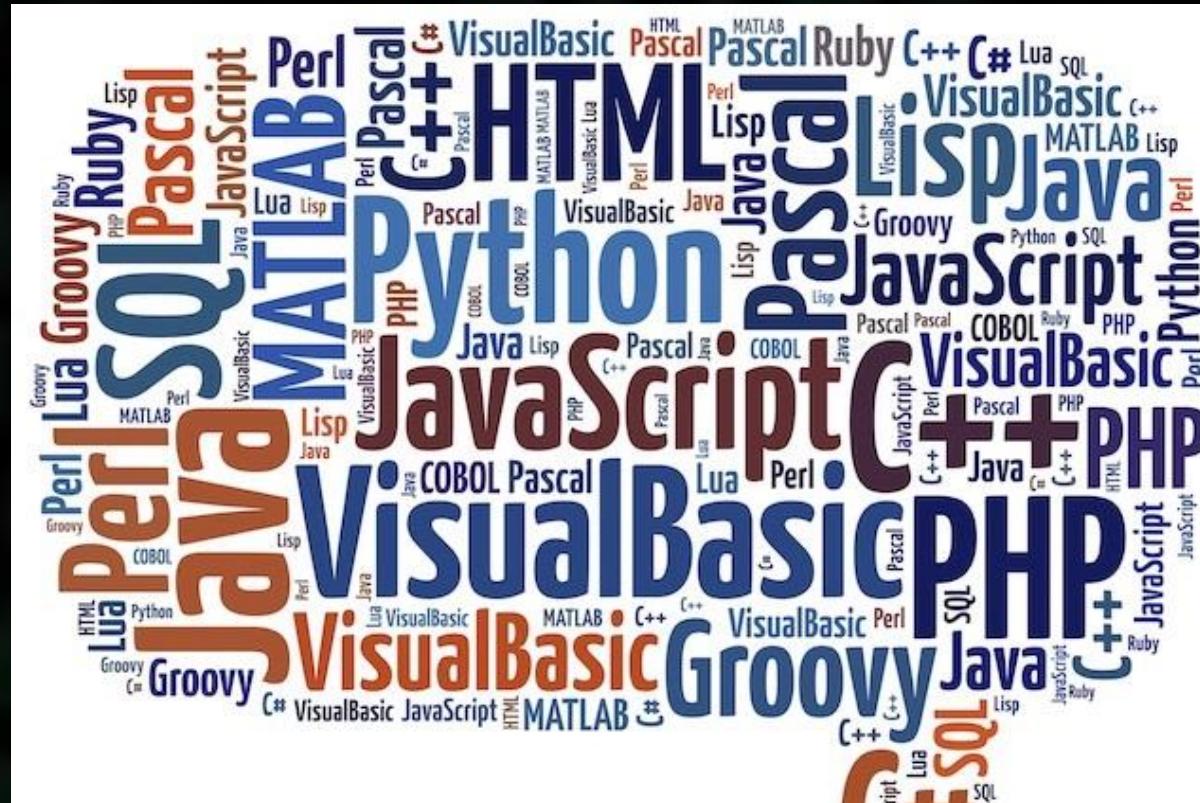
# 為什麼需要程式

Ans: 計算量(重複性)超出人腦負荷時，需要依靠電腦

1. 計算全台灣指考考生的物理平均
2. 計算本校這學期的平均遲到人數
3. 設計定時紅綠燈標誌
4. ...

# “程式語言” 是什麼？

Ans: 人類用來跟機器溝通的工具



# “程式語言”怎麼選?

Ans: 看你的需求

目標	應用	語言
系統程式設計	作業系統	C
網路後端應用程式	資料庫、網站架構	PHP、Ruby、Python
網路前端應用程式	所有網頁的視覺編排	JavaScript、HTML、CSS
Android 應用程式	Android App	Java、Kotlin
Apple iOS 應用程式	iOS App	Swift、Objective-C
資料分析	大數據、文本分析	R、Python

Jun 2022	Jun 2021	Change	Programming Language	Ratings	Change
1	2	▲	Python	12.20%	+0.35%
2	1	▼	C	11.91%	-0.64%
3	3		Java	10.47%	-1.07%
4	4		C++	9.63%	+2.26%
5	5		C#	6.12%	+1.79%
6	6		Visual Basic	5.42%	+1.40%
7	7		JavaScript	2.09%	-0.24%
8	10	▲	SQL	1.94%	+0.06%
9	9		Assembly language	1.85%	-0.21%
10	16	▲	Swift	1.55%	+0.44%

<https://tw.alphacamp.co/blog/programming-for-beginner>

<https://www.tiobe.com/tiobe-index/>

# Why Python

- Easy to learn
- Open Source & Free
- Cross Platform
- Many Lib

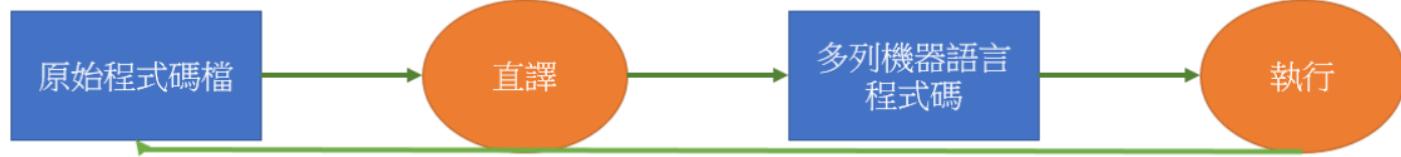
函式庫名稱	適用於
Pandas	數據操作與處理
Matplotlib	繪製圖表
OpenCV	電腦視覺
Plotly	視覺化互動圖表
Django、Weppy、Bottle、Flask	服務器端 Web 開發
Selenium	網頁爬蟲、自動化測試
BeautifulSoup	解析 HTML 和 XML
SciPy	工程應用、科學和數學
NumPy	科學計算
TensorFlow、Keras、Torch	深度學習
scikit-learn	機器學習

□ (補充：Python 2.7 為較舊的版本，會有部份功能跟語法和 Python 3 不同)

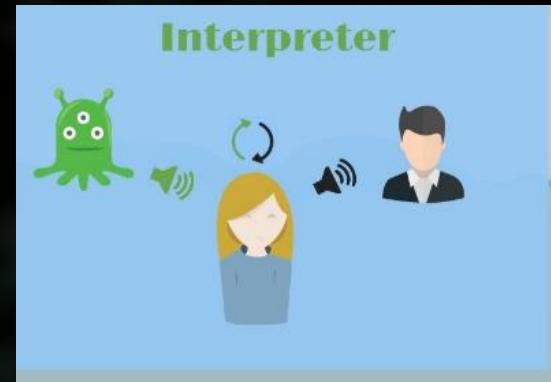
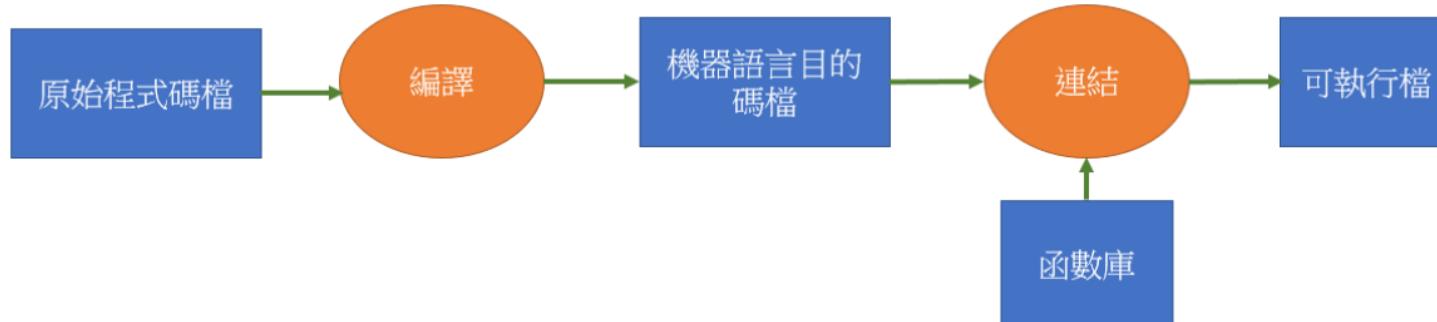
# Python 與 其他語言的差異

Python作為一種直譯語言，並不需要編譯器，因此不會輸出可執行的檔案，直譯式與編譯語言運行流程如下：

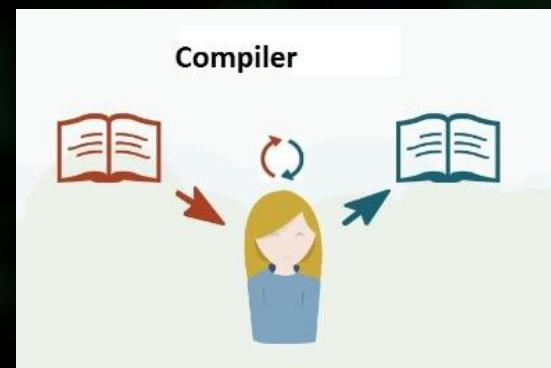
- 直譯式語言



- 編譯式語言



直譯器就像一個口譯者  
所見即所得



編譯器就像一個翻譯者  
換成另一個語言

# Quiz

下列敘述何者正確？

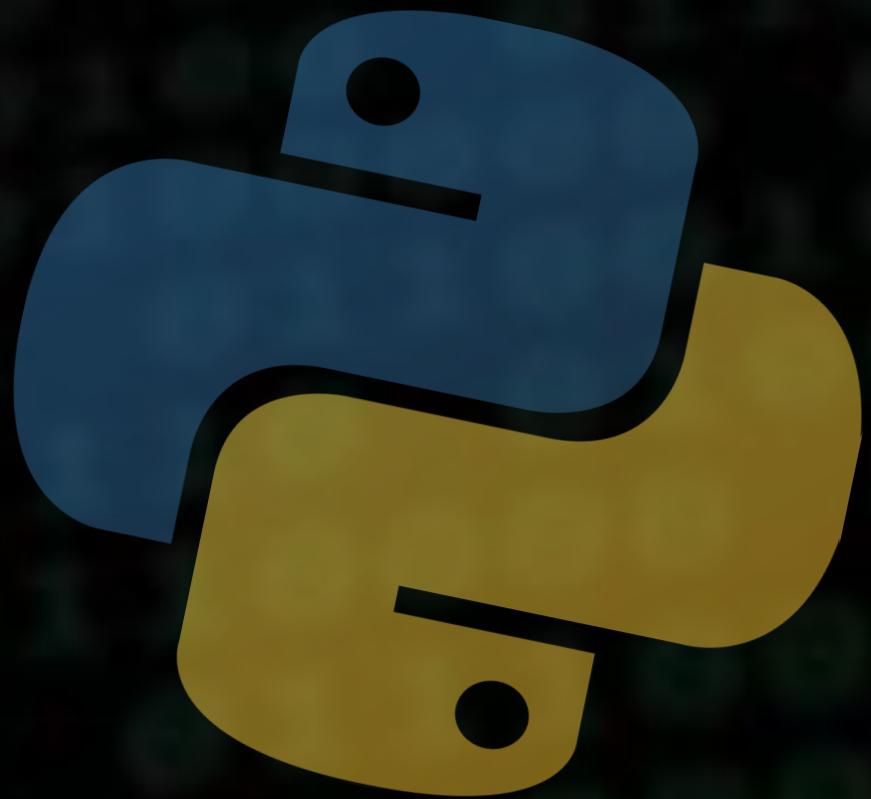
- A. Python 是一種程式語言，雖然適合初學者學習，但不太實用
- B. 具有「特定規則」的事情適合寫程式來讓電腦完成
- C. 只有電腦裡面會有程式
- D. 傳統電腦裡面沒有程式

# Ans

下列敘述何者正確？

- A. Python 是一種程式語言，雖然適合初學者學習，但不太實用
- B. 具有「特定規則」的事情適合寫程式來讓電腦完成
- C. 只有電腦裡面會有程式
- D. 傳統電腦裡面沒有程式

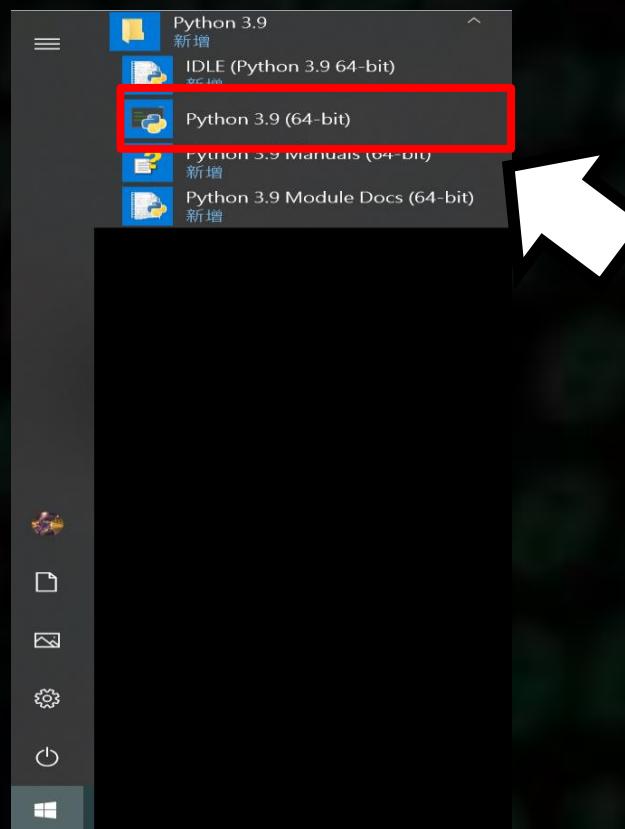
# 開發環境



# 開發環境設定

- 安裝Python

<https://www.python.org>



The top half of the image shows the Python.org homepage. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The 'Downloads' button is highlighted with a red box. Below the navigation is a banner with code snippets and a 'Functions Defined' section. The bottom half shows a screenshot of a Windows Command Line window titled 'Python 3.9 (64-bit)'. It displays the Python interpreter prompt and a Fibonacci sequence example. Below the window, the text 'Command Line' is written.

# 開發環境比較

## □ 線上開發器: Colab

★ 快速、方便、免安裝

✗ 不適合正式開發

## □ 整合開發環境: PyCharm, Jupyter, Spyder ...

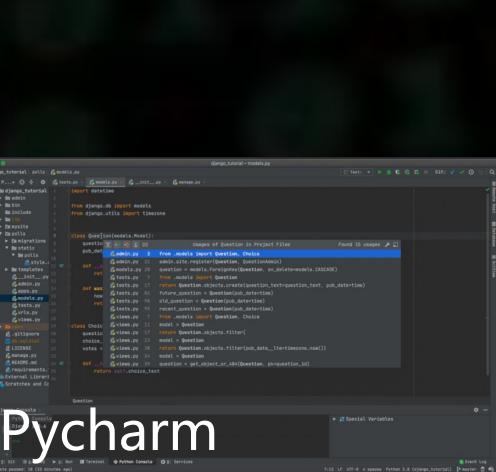
★ 整合除錯, 程式碼提示...等多功能

✗ 品牌眾多

## □ 多元整合開發平台: Anaconda, conda

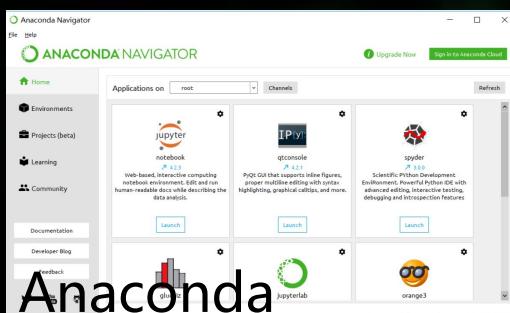
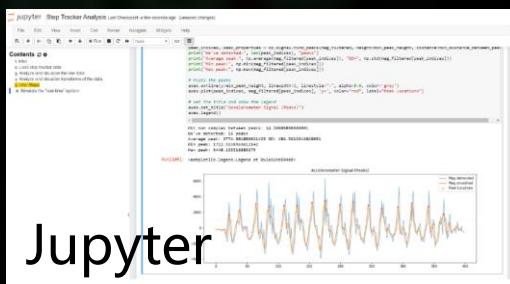
★ 整合多種開發環境, 整合套件與環境管理系統

✗ 檔案龐大



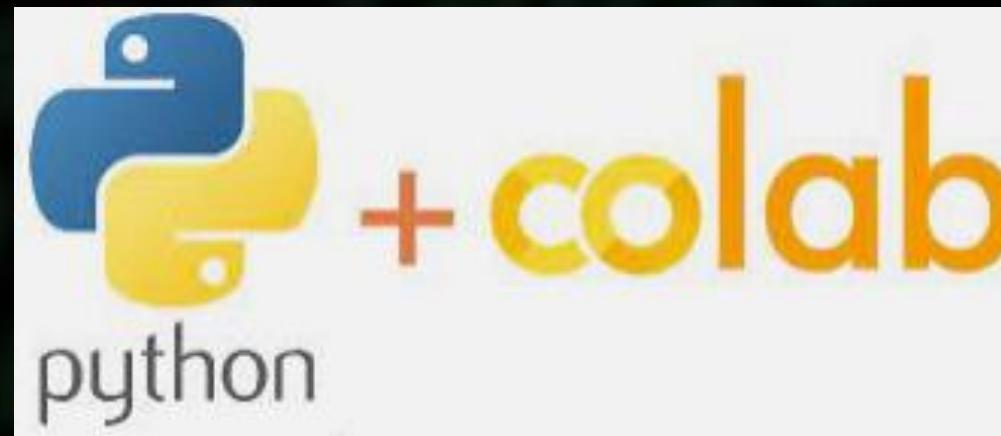
```
[1]: 1 pip install opencv-contrib-python
2 Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.6/dist-packages (from opencv-contrib-python)
3 Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from opencv-contrib-python)
```

Colab

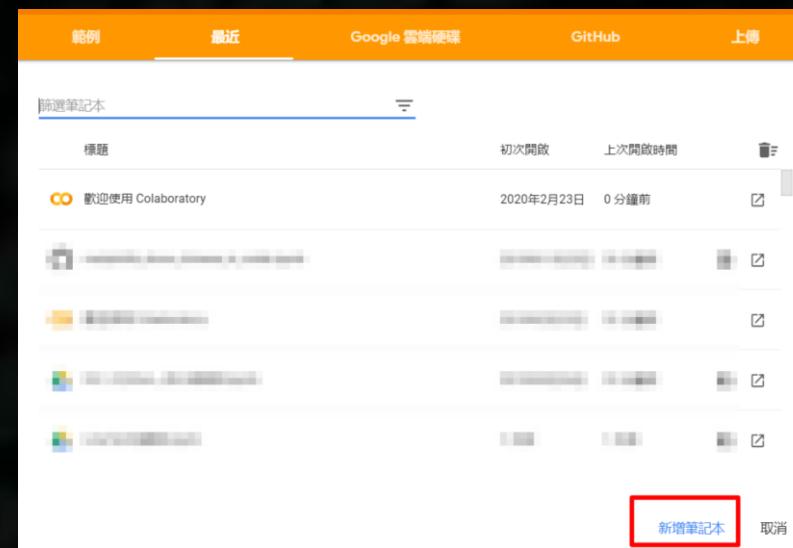


# 好用的Colab

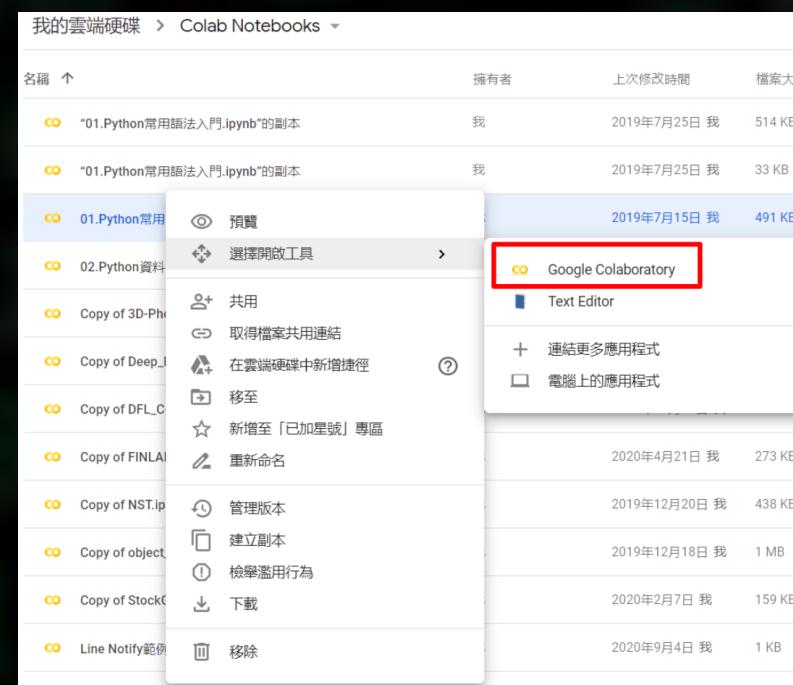
- Colab (全名為「Colaboratory」) 可讓你在瀏覽器中編寫及執行 Python 程式碼
- 不必進行任何設定
- 只需要google 帳號
- 免費 的 GPU & CPU



# 從Google搜尋Colab新增或開啟檔案



# 從 Google 雲端資料夾開啟 Colab



# 更改檔名



變數視窗

檔案視窗

程式編輯區塊

執行程式

# 寫下第一行程式碼

流傳許久的都市傳說...

在學新的程式語言的時候  
一定要先寫 “Hello World!”  
不然就會學不好



由 布萊恩·柯林漢 撰寫的「Hello, world」程式 (1978年)

[https://zh.m.wikipedia.org/zh-tw/Hello\\_World](https://zh.m.wikipedia.org/zh-tw/Hello_World)

```
[1] print("Hello world!")
[1] > Hello world!
```



Brian Kernighan

# Colab 優點與缺點

- 優點
  - 快速、方便
  - 免事先安裝任何東西，只需要有適合的瀏覽器
  - 適合在手邊沒有安裝好開發環境的電腦，卻有急迫的需求
  
- 缺點
  - 需要另外把寫好的程式碼下載下來
  - 不適合正式開發，有長期需求還是建議安裝開發環境
  - 需要網路，沒有網路就不能使用正式的

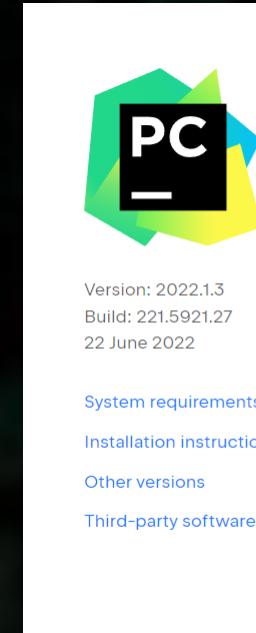
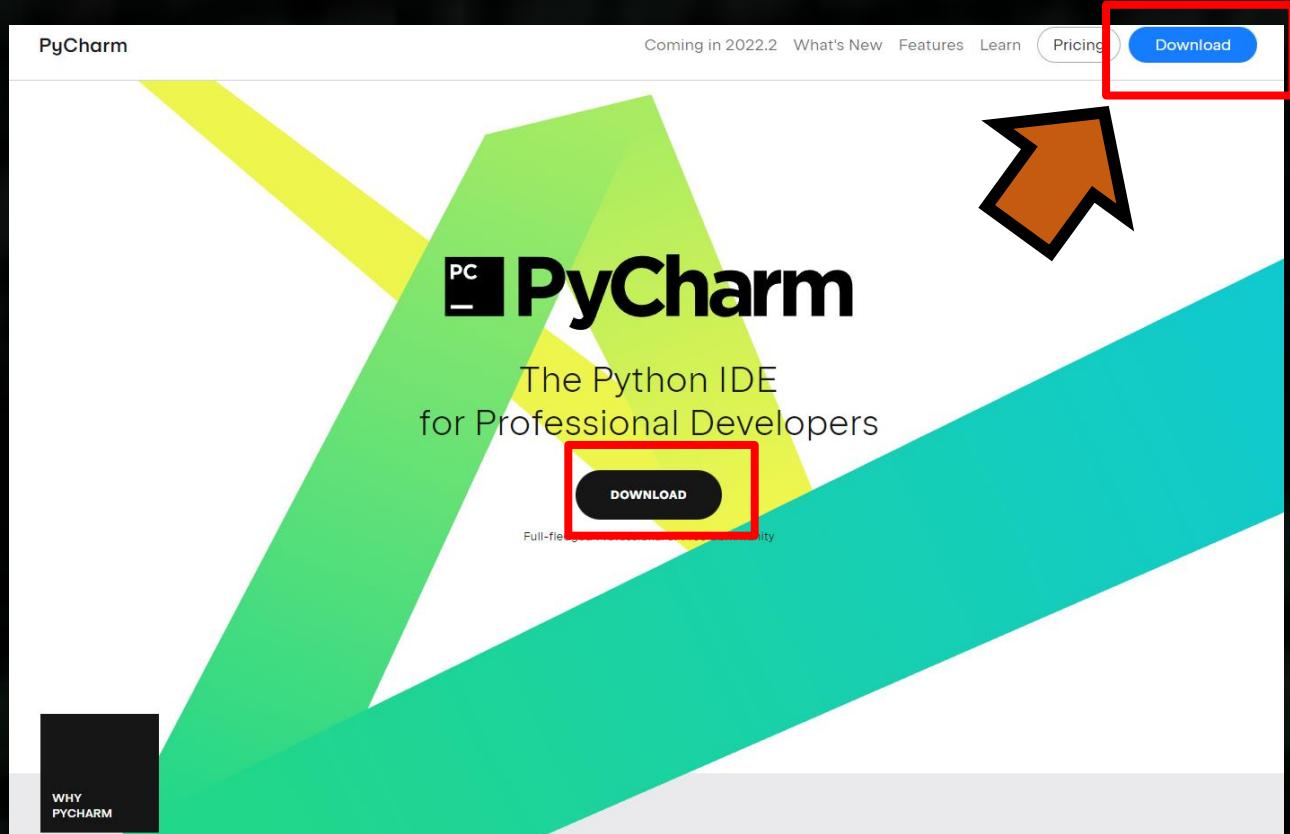
# 專業感十足的IDE - PyCharm

A screenshot of the PyCharm IDE interface. The main window shows a code editor with Python code for a Django application. A search results panel is open, showing 15 usages of the class 'Question'. The project structure on the left includes files like 'models.py', 'tests.py', and 'admin.py'. At the bottom, there are toolbars for Git, Terminal, Python Console, and Services, along with an Event Log.

Pycharm

- 整合多種套件
- 語法提示
- 整合多元視窗與檔案總管
- 除錯功能
- 執行斷點 (break point)

# PyCharm



## Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

### Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free 30-day trial available

### Community

For pure Python development

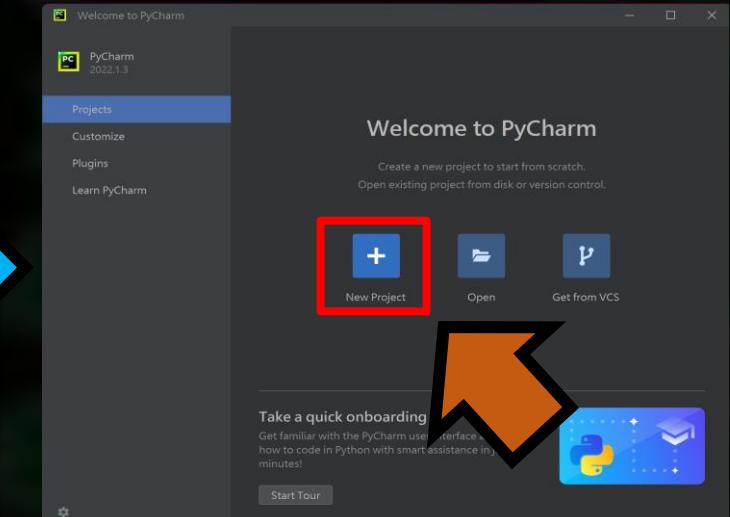
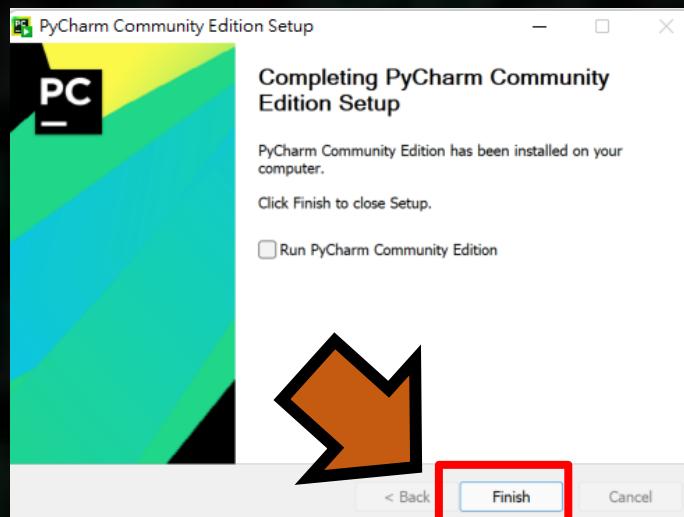
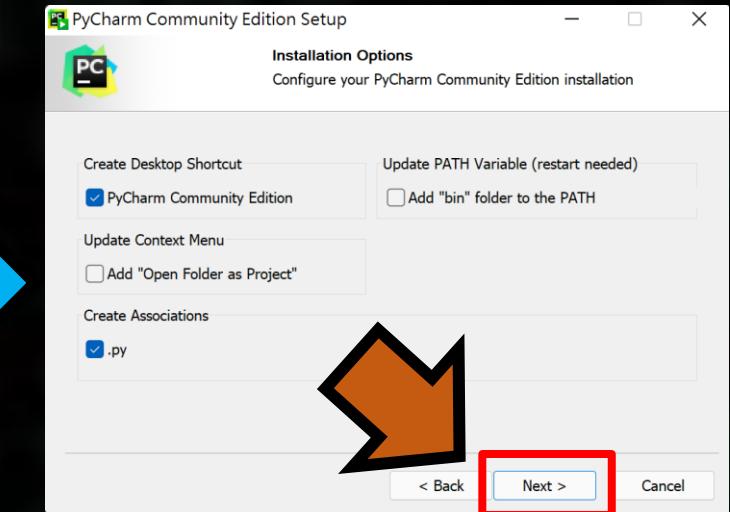
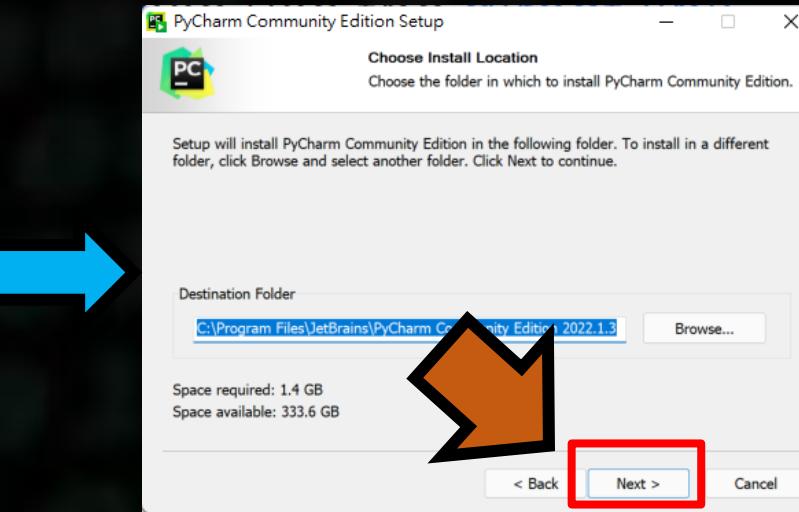
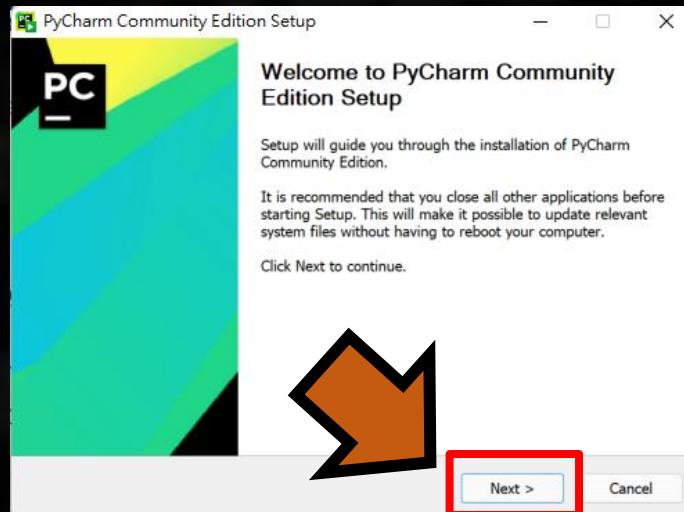
[Download](#)

Free, built on open-source

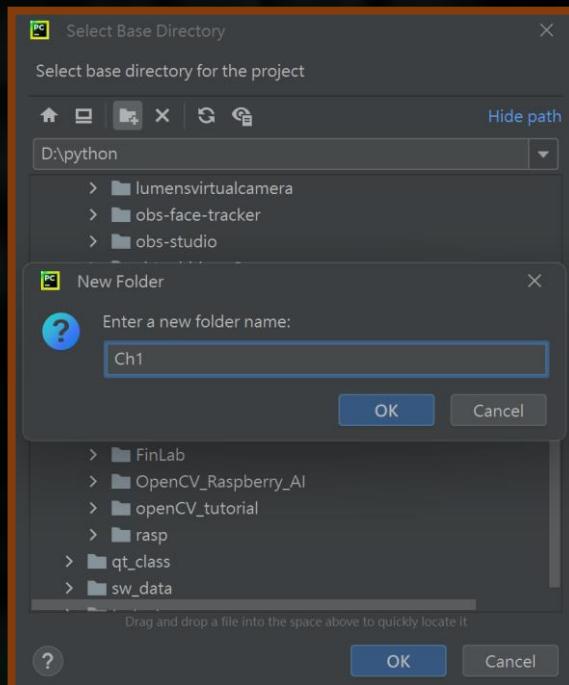
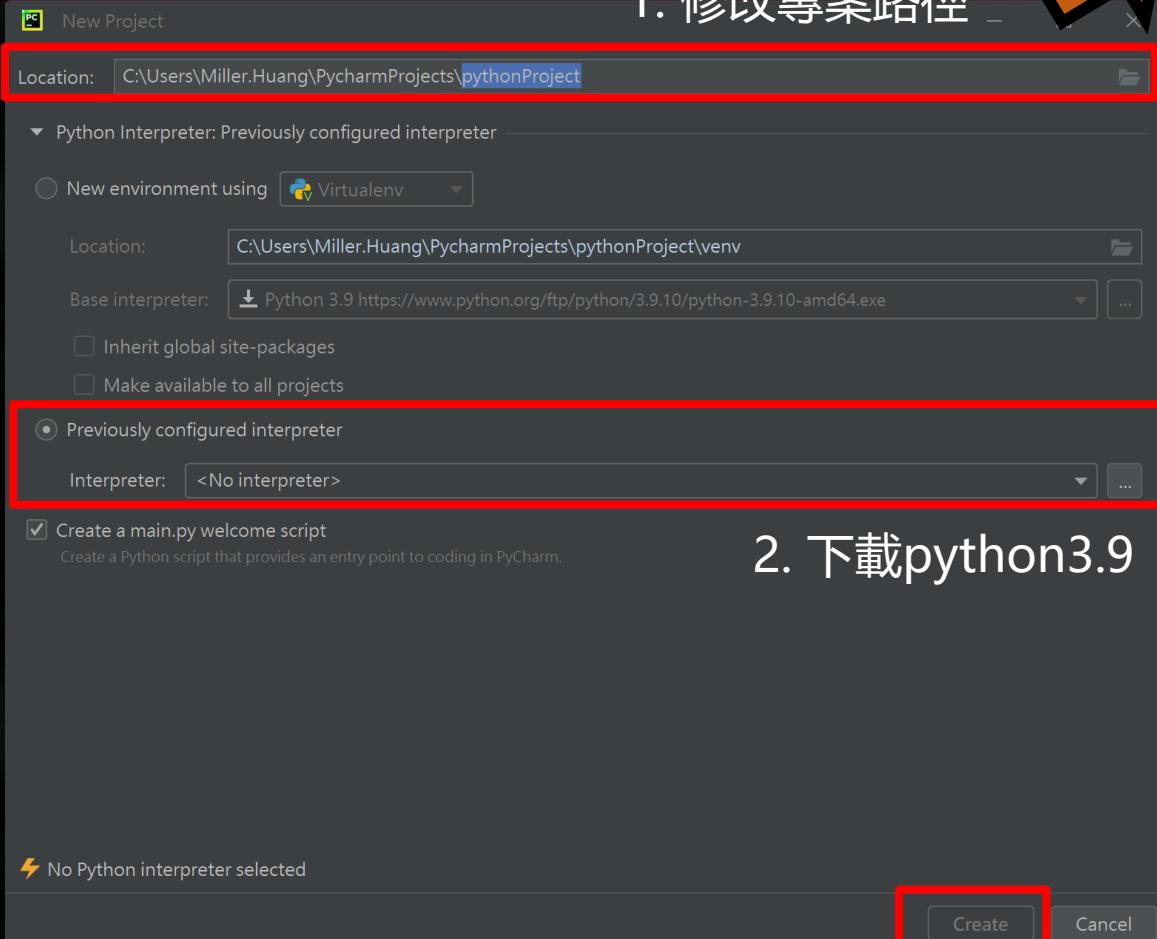


<https://www.jetbrains.com/pycharm/>

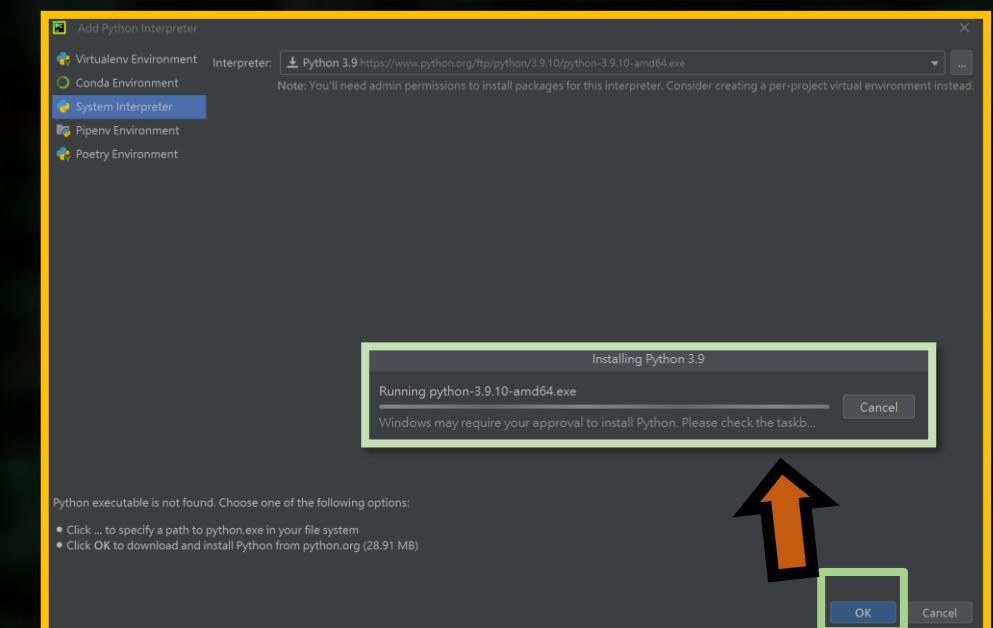
# PyCharm



# PyCharm



## 3. 創建專案



The screenshot shows the PyCharm IDE interface with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Ch1 - main.py. The left sidebar has a Project view showing a Ch1 folder containing main.py, External Libraries, and Scratches and Consoles. The main editor window displays the following Python code:

```
# This is a sample Python script.  
# Press Shift+F10 to execute it or replace it with your code.  
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.  
  
def print_hi(name):  
    # Use a breakpoint in the code line below to debug your script.  
    print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.  
  
# Press the green button in the gutter to run the script.  
if __name__ == '__main__':  
    print_hi('PyCharm')  
  
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

A yellow 'Try it !!!' watermark is overlaid on the bottom right of the screen. A notification at the bottom right corner says "Localized PyCharm 2022.1.3 is available" with a "Switch and restart" button.



# PyCharm

- 優點
  - 完整的功能介面
  - 適合正式的專案開發(個人 or 團隊)
  
- 缺點
  - 安裝與操作複雜, 不易上手
  - 需要網路, 沒有網路就不能寫程式碼

# Anaconda



- Anaconda
  - 完整的 Python 開發環境  
(含 Python 安裝與常用套件)
  - 將「常用的函式庫」都納入其中
  - 跨平台支援
- <https://www.anaconda.com/>

# Anaconda

The screenshot shows the Anaconda website's distribution page. At the top, there's a navigation bar with links for Products, Pricing, Solutions, Resources, Partners, Blog, Company, and Contact Sales. Below the navigation, a green banner states "Individual Edition is now ANACONDA DISTRIBUTION". The main heading "ANACONDA DISTRIBUTION" is in large green capital letters. Below it, the text "The world's most popular open-source Python distribution platform" is displayed. On the right side of the page, there's a call-to-action section for the "Anaconda Distribution" with a "Download" button (highlighted with a red box) and icons for Windows, Mac, and Linux. An orange arrow points towards the "Download" button.

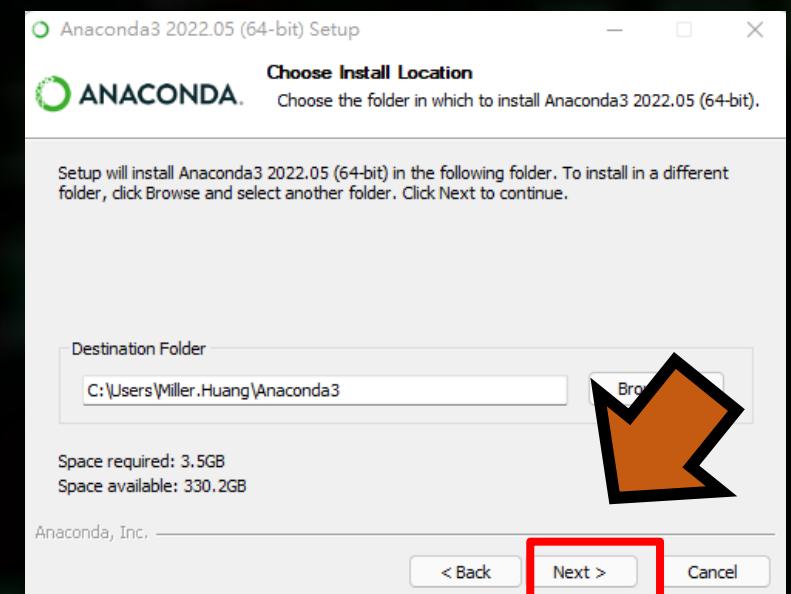
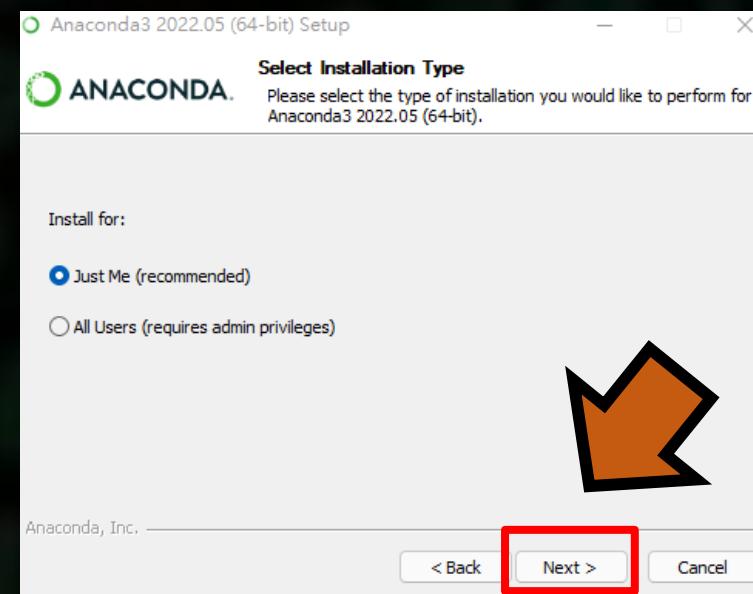
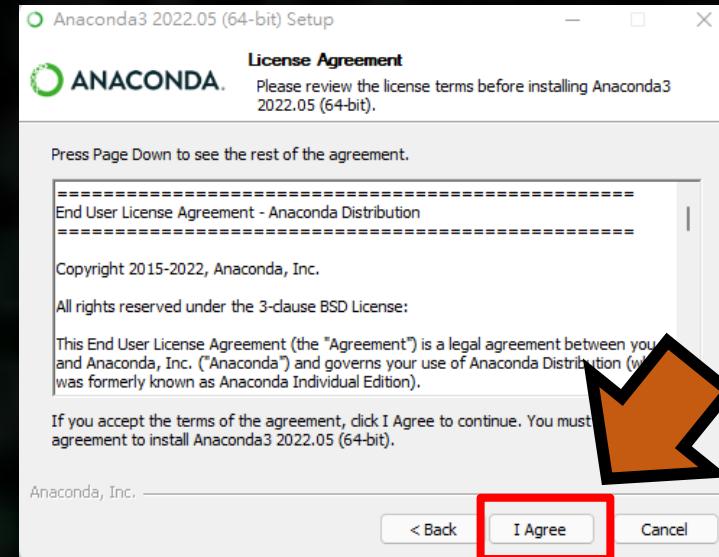
<https://www.anaconda.com/products/distribution>

The screenshot shows the Anaconda website's installers page. The title "Anaconda Installers" is at the top. Below it, there are three sections: "Windows" (with icons for Windows and a blue link), "MacOS" (with icons for Mac and a blue link), and "Linux" (with icons for Linux and a blue link). Each section lists various installer options for Python 3.9, including 64-Bit Graphical Installers, 32-Bit Graphical Installers, and Command Line Installers for both x86 and Power architectures.

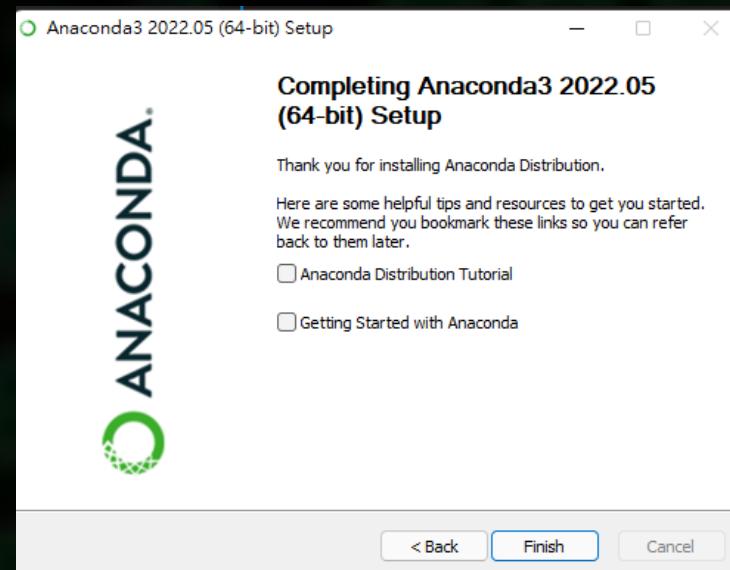
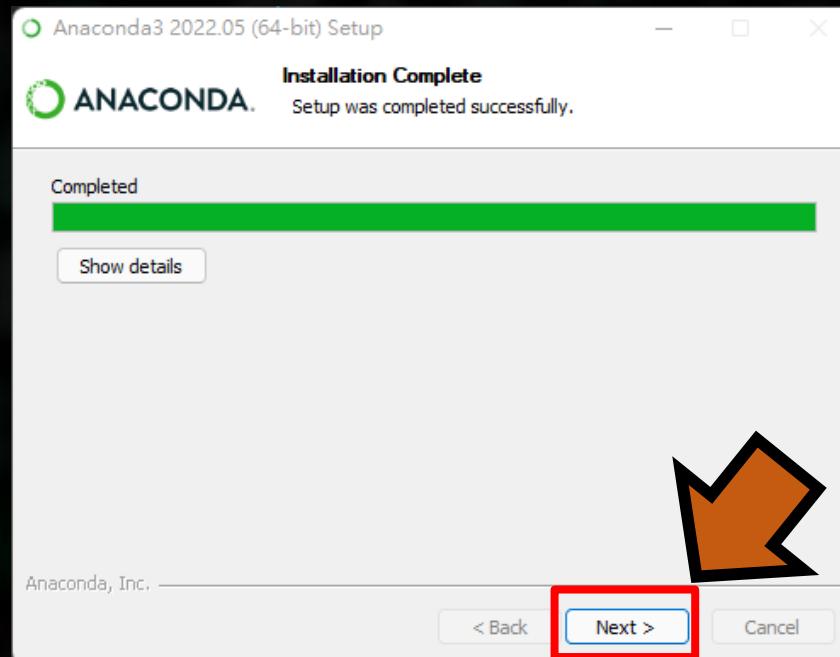
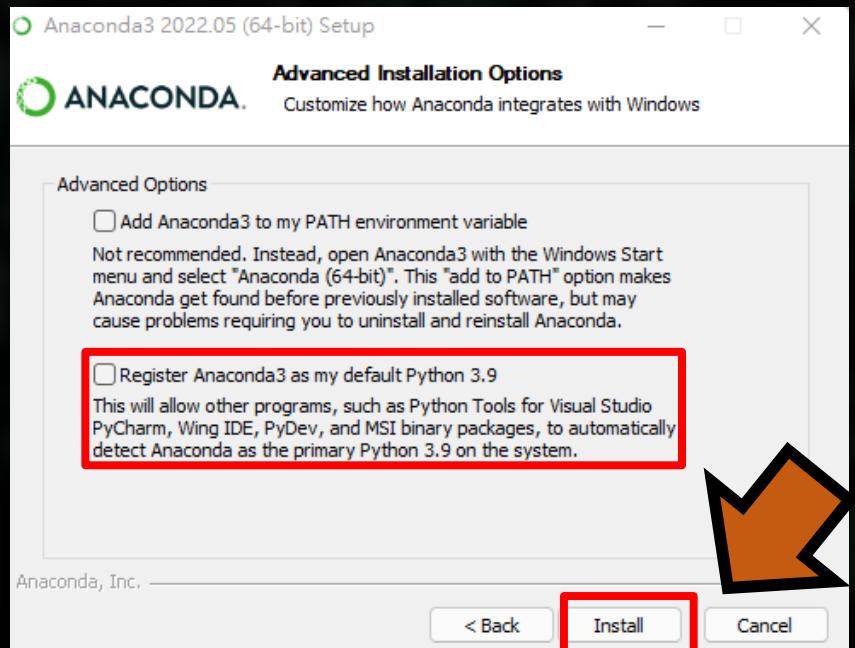
Platform	Type	Version	File Size
Windows	Graphical Installer	Python 3.9	(594 MB)
	Graphical Installer	Python 3.9	(488 MB)
MacOS	Graphical Installer	Python 3.9	(591 MB)
	Command Line Installer	Python 3.9	(584 MB)
Linux	Graphical Installer	Python 3.9	(659 MB)
	Command Line Installer	Python 3.9	(367 MB)

Or Download others install file

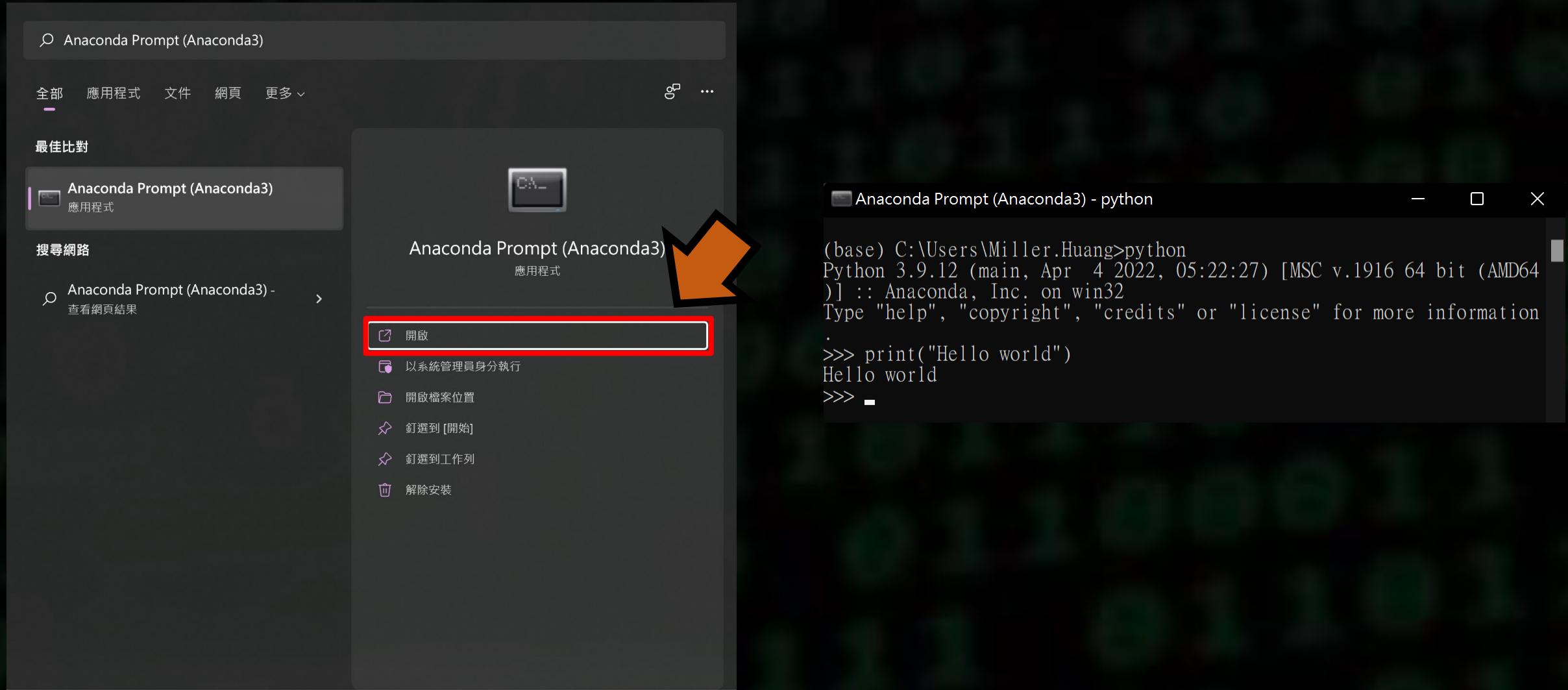
# Anaconda



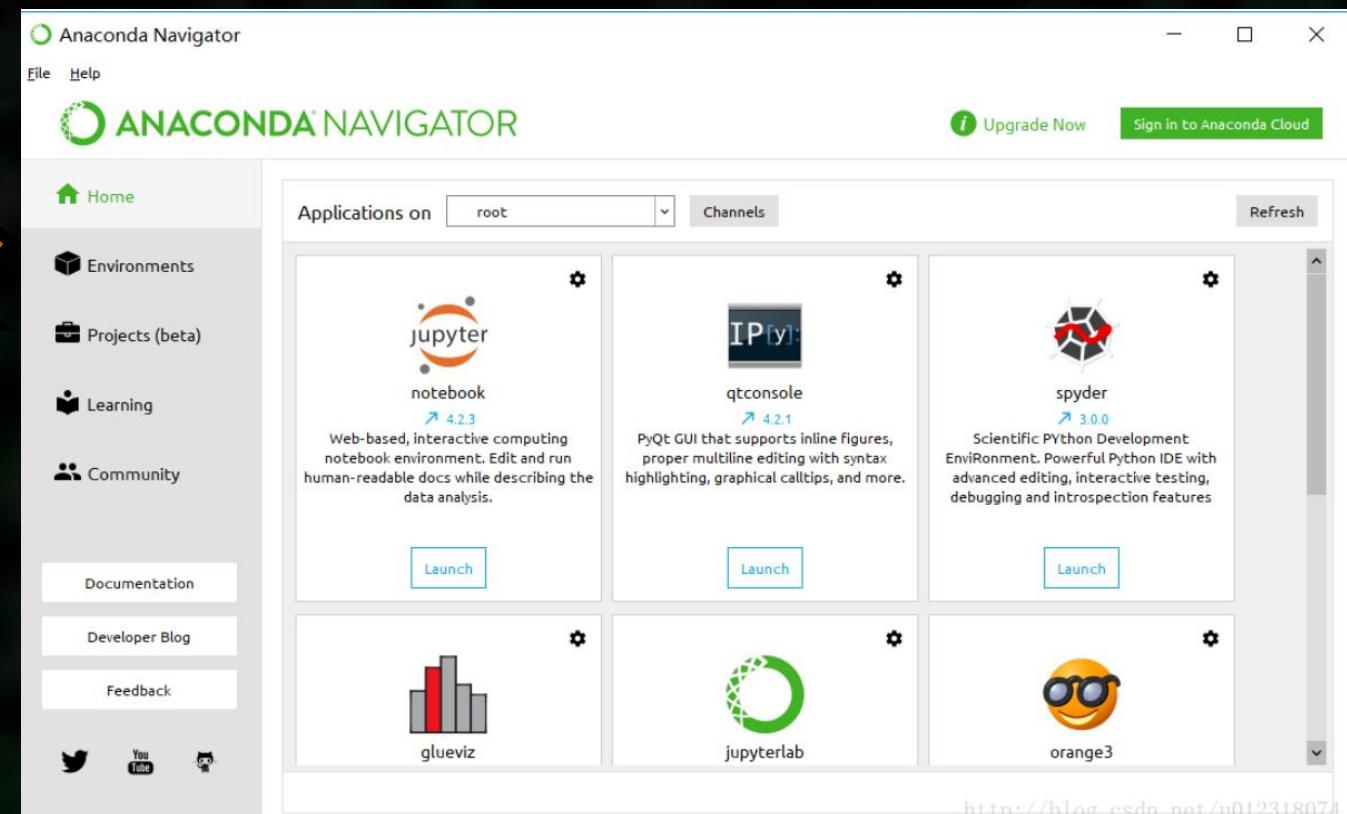
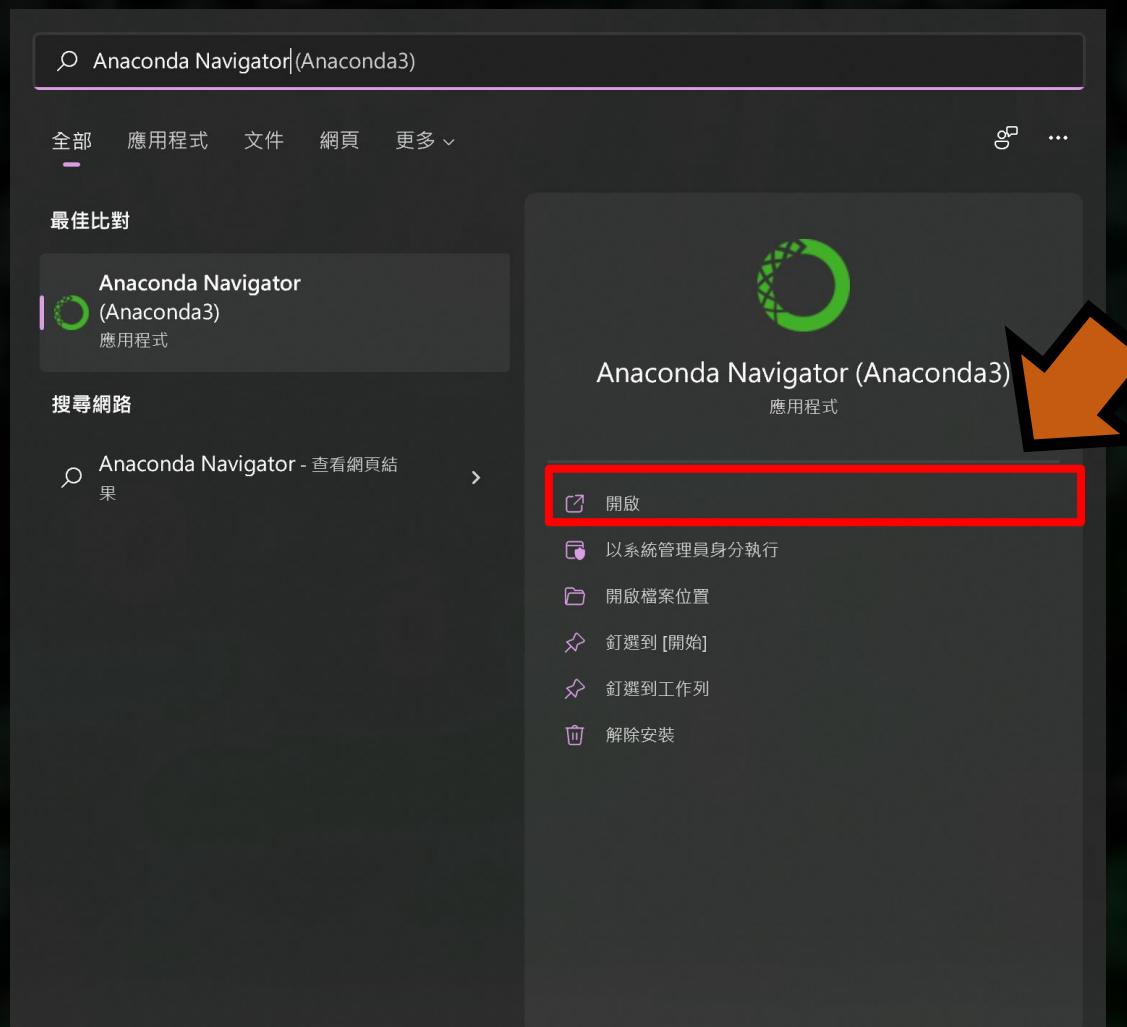
# Anaconda



# Anaconda



# Anaconda



# Anaconda



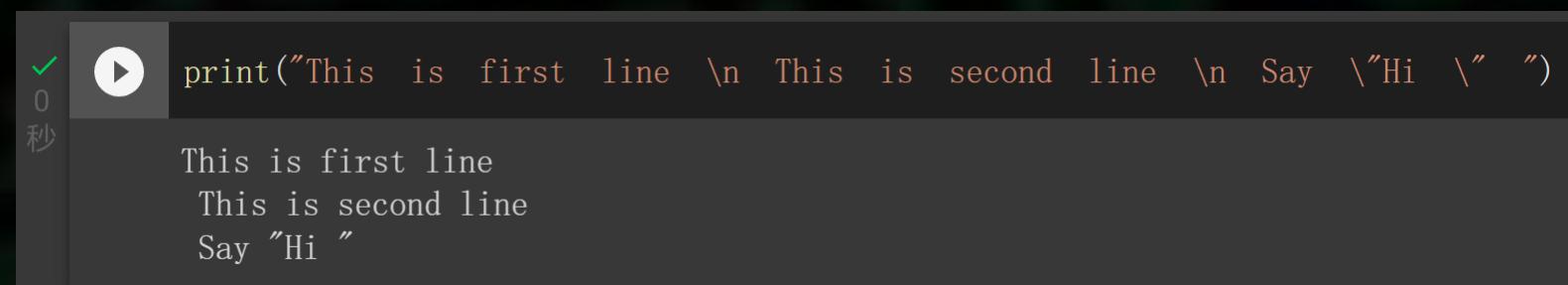
- 優點
  - 整合多種IDE
  - 具有套件管理系統
  - 輕鬆切換不同Python版本環境
  
- 缺點
  - 檔案太大
  - 需要網路，沒有網路就無法運行程式碼

# Python 語言基礎

# 好用的print()

- 可以將括號內的東西輸出到螢幕上
- 如果要輸出字元&字串，需要用成對的單引號（‘ ’）或雙引號（“ ”）將字元串括起來
- 如果要輸出其他東西，如：數字、變數則不需要單引號或雙引號
- 如果要輸出單引號或雙引號時就不能直接使用，需要依靠跳脫字元 “\”  
跳脫字元是字元的一種特殊情況，也標示著跳脫序列開始的那個字元。

單引號	\'
雙引號	\"
換行符號	\n
Tab (跳格)	\t



A screenshot of a terminal window showing the execution of a Python script. The terminal interface includes a progress bar at the top, a play button icon, and a timer indicating 0 seconds. The code is:

```
print("This is first line \n This is second line \n Say \"Hi \"")
```

The output below the code shows three lines of text: "This is first line", "This is second line", and "Say "Hi "".

# 好用的註解

- 註解是專門「寫給人看的」，電腦在執行程式碼的時候會直接跳過註解的部分
- 主要是用來「提醒自己」或「告訴他人」這一段程式碼寫了些什麼東西

優點:

- 可以用註解來標示整份程式碼的架構
- 讓很久之後地自己也能看懂現在的自己在寫什麼
- 讓別人可以快速了解這一段程式碼的用途

缺點:

- 麻煩、花時間

The screenshot shows a Jupyter Notebook interface with two code cells and their outputs.

- In [1]:** `# This is a comment`
- In [2]:** `# Print "Hello World!"`  
`print("Hello World!")`

The output for In [2] is "Hello World!". A green rectangular box highlights the input field for In [3].

# Quiz

請根據下方程式碼，選擇一個正確的描述

```
1 # average_cal 函數計算平均值
2 # x 是數據1
3 # y 是數據2
4 # 返回 x 與 y 的平均值
5 def average_cal(x, y):
6     comment = '# 輸出值'
7     return (x + y)/2 # x 與 y 的平均值
```

01到04行在語法檢查時將被忽略。

02和03行中的井字符號(#)非必填。

06行中的字串將被解釋為註釋。

07行中包含內嵌註釋。

07行中可用“//”代替“#”作為註釋標記。

# Ans

請根據下方程式碼，選擇正確的描述

```
1 # average_cal 函數計算平均值
2 # x 是數據1
3 # y 是數據2
4 # 返回 x 與 y 的平均值
5 def average_cal(x, y):
6     comment = '# 輸出值'
7     return (x + y)/2 # x 與 y 的平均值
```

Y 01到04行在語法檢查時將被忽略。

02和03行中的井字符號(#)非必填。

06行中的字串將被解釋為註釋。

Y 07行中包含內嵌註釋。

07行中可用“//”代替“#”作為註釋標記。

變數宣告



想像一下，你手上有一個設計藍圖，你可以根據這個藍圖，蓋出任意外觀的房子





型別  
type



物件  
object

型別(type)



物件(object)



物件(object)



...

Python 內建多種**基礎**型別Type

數值型態(Numeric type) - int, float, bool

字串型態(String type)

容器型態(Container type) - list, set, tuple

型別(type)



物件(object)



描述 : value

500 坪



100 坪

型別(type)      物件(object)/變數(variable)      描述 : value

integer

integerObj

10

宣告

指派

C++  
C#  
Java

```
Integer integerObj = 10
```

靜態宣告

javaScript

```
var integerObj = 10
```

動態宣告

python

```
integerObj = 10
```

動態宣告

```
integer,parameter :: NX=200,NY=200,NZ=30
integer, parameter :: dx = 1000,dy = 1000, dz = 250
real, parameter :: rlatd = 22.25, rlonl = 119.75
real, parameter :: lon_l = 118.29151, lat_l = 20.900108
real, parameter :: lon_g = 9.72325634E-03, lat_g = 8.99927784E-03
```

!~background

```
real :: u0(NX,NY,NZ),v0(NX,NY,NZ),qv0(NX,NY,NZ),t0(NX,NY,NZ)
```

!~ascat

```
integer,parameter :: sea_info=358
```

```
real :: sea_site(3,sea_info)
```

```
integer :: sea_site_id(sea_info)
```

```
character*13 :: infile_sea(sea_info)
```

!

```
INTEGER :: ID,YY,MM,DD,HH,NN,itime
```

```
REAL :: pre,temp,td,RH,WIND,WINDIR
```

```
real :: alt(sea_info),lon(sea_info),lat(sea_info)
```

```
real :: spd(sea_info),dir(sea_info)
```

!

```
real,parameter :: badpt = -999.98999
```

```
real :: ua(sea_info),va(sea_info)
```

```
real,parameter :: u_hfactor=-7.7368259E-02
```

```
real,parameter :: v_hfactor=0.612216
```

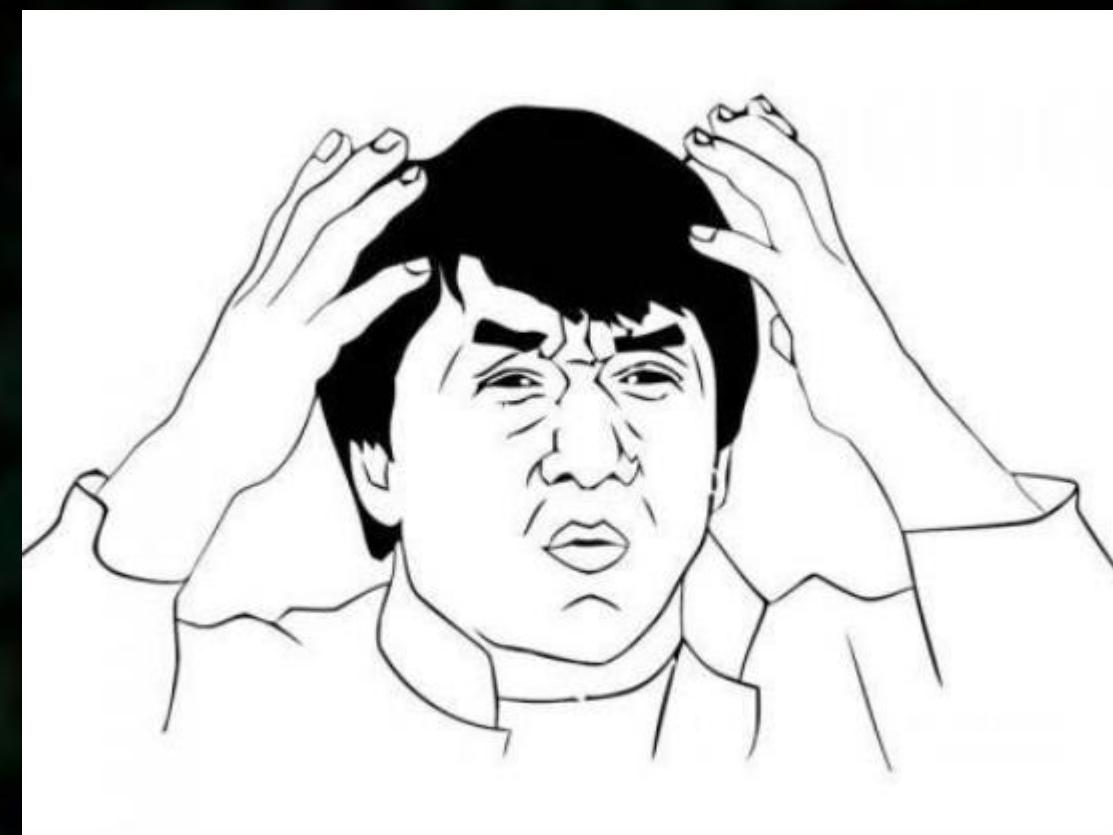
```
real,parameter :: u_mfactor=-0.6058777
```

```
real,parameter :: v_mfactor=-0.1411584
```

```
real,parameter :: u_lfactor=6.8603404E-02
```

```
real,parameter :: v_lfactor=0.1272687
```

沒有動態宣告的狀況...





C++之父：Bjarne Stroustrup



Java之父：James Gosling



Python之父：Guido van Rossum  
1024.com



PHP之父：Rasmus Lerdorf

# Warning !!

靜態宣告 vs 動態宣告

型別(type)

integer

變數(variable)

integerObj

描述 : value

10

宣告

指派

靜態宣告

```
integer a = 10  
integer a = 20  
Integer b = 10
```

10

a

20

a

10

b

記憶體

Position 1

Position 1

Position 2

每宣告完一個物件(變數)，即為該物件開一個記憶體位置。

修改物件(變數)的value，即是對該記憶體位置的value做修改。

型別(type)

integer

變數(variable)

integerObj

描述 : value

10

宣告

指派

動態宣告

```
a = 10  
a = 20  
b = 10
```

記憶體

10  
a

Position 1

20  
a

Position 2

10  
b

Position 1

每宣告完一個物件(變數)，即為該值(value)建一個記憶體位置。  
物件名稱(變數名稱)只是依附在該值(value)的一個標籤。

## 使用 id() 來查看記憶體空間

```
a = 10  
print(id(a))
```

```
a = 20  
print(id(a))
```

```
b = 10  
print(id(b))
```

```
11256352  
11256672  
11256352
```

```
a = 10  
print(id(a))
```

```
a = 20  
print(id(a))
```

```
b = a  
print(id(b))
```

```
11256352  
11256672
```

思考一下變數b的記憶體位置應該在哪裡

# 變數命名

## 變數-python 保留字

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

- 變數名稱大小寫有別
- 變數名稱不得是系統內的**保留字**
- 變數名稱必須由英文、數字與底線（\_）組成  
第一個字元必須以英文或是底線開頭、中間不能有空白
- 變數名稱提供程式記憶體位置，用來儲存資料在記憶體(memory)內
- 所有變數使用前都必須事先宣告
- 變數名稱必須賦予**有意義**的命名

Try it !!!

# Quiz

下列敘述何者正確？

- A. 變數可以隨心所欲的命名
- B. 「aVariable」以及「a Variable」皆可作為變數使用
- C. 某些保留字不能作為變數使用
- D. 變數只能使用一次就沒有作用了

# Ans

下列敘述何者正確？

- A. 變數可以隨心所欲的命名
- B. 「aVariable」以及「a Variable」皆可作為變數使用
- C. 某些保留字不能作為變數使用
- D. 變數只能使用一次就沒有作用了

# Python 資料型態

# 什麼是「資料型態」

- 「資料型態」就是變數內容物的「類別」，又稱為「型態」
- 不同資料型態的變數所能做的操作也不同
- 有些資料型態之間可以跟彼此互動
- 在滿足特定的條件下，資料型態可以互相轉換
- 在 Python 3 裡面的多種基本資料型態

數值型態(Numeric type) - int, float, bool

字串型態(String type)

容器型態(Container type) - list, set, tuple

# bool(布林值)

- 只有兩種型態 – **True, False**
- 可以當作數字 1 和 0 使用

```
In [47]: aaa=True  
print(aaa)
```

```
True
```

# int(整數)

- 基本上可以當作沒有上限與下限
- 可以做各種數學運算
- Ex. 0, 10, 200, -300, 9223372036854775807

```
In [48]: aaa=50  
print(aaa)
```

50

# float (浮點數)

- 就是小數 (一定要有小數點)
- 可以做各種數學運算，並且可以跟 int 互動
- Ex. 0.123, 0.0, 123.456

```
In [49]: aaa=0.6  
print(aaa)
```

```
0.6
```

# String (字串)

- 必須以成對的單引號 (' string' ) 或是雙引號 (" string" ) 来表示一串字元是一個字元串
- 可以使用 + 將兩個 string 連接起来
- 或是用 \*，乘上一個 int 來重複字元串
- Ex. 'one', 'two', "ABCDEFG", "", ""

```
In [1]: aaa="this is a "
bbb='string'
print(aaa+bbb)
print(3*bbb)
```

this is a string  
stringstringstring

# 利用type來查看資料型態

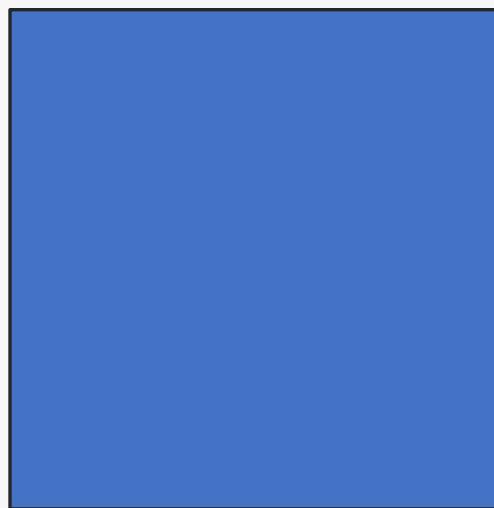
- type命令會取得該變數的資料型態，如果使用者不確定某些變數的資料型態，可以利用type來確認
- 語法: type(變數)
- 範例:

```
In [52]: aaa=False  
print(type(aaa))  
  
<class 'bool'>
```

# Quiz

動手試試看:分別宣告四種不同的資料型態的資料，並且利用type輸出來查看

In [53]:



```
<class 'bool'>
<class 'int'>
<class 'float'>
<class 'str'>
```

# Ans

動手試試看:分別宣告四種不同的資料型態的資料，並且利用type輸出來查看

```
In [53]: aaa=False  
print(type(aaa))  
aaa=5  
print(type(aaa))  
aaa=5.0  
print(type(aaa))  
aaa="5"  
print(type(aaa))
```

```
<class 'bool'>  
<class 'int'>  
<class 'float'>  
<class 'str'>
```

# Quiz

你正在編寫一個Python程式。該程式收集客戶資料並將其儲存在資料庫中。

這個程式處理各式各樣的資料。你需要確保程式能夠正確的處理資料，以便能夠正確的儲存在資料庫中。將資料類型與程式碼片段對應。回答時，請將適當的資料類型從左側的列拖動到右側的程式碼片段。每個運算式都可以使用一次、多次、或者不使用。

bool

float

int

str

age = 30

pps = True

name = "SUSSS"

strange = 777.2

pip = "9600"

# Ans

你正在編寫一個Python程式。該程式收集客戶資料並將其儲存在資料庫中。

這個程式處理各式各樣的資料。你需要確保程式能夠正確的處理資料，以便能夠正確的儲存在資料庫中。將資料類型與程式碼片段對應。回答時，請將適當的資料類型從左側的列拖動到右側的程式碼片段。每個運算式都可以使用一次、多次、或者不使用。

bool	<code>int</code>	<code>age = 30</code>
float	<code>bool</code>	<code>pps = True</code>
int	<code>str</code>	<code>name = "SUSSS"</code>
str	<code>float</code>	<code>strange = 777.2</code>
	<code>str</code>	<code>pip = "9600"</code>

# Python 算術運算子

# Python 數學計算

數學符號	功能
+	加法
-	減法
*	乘法
/	除法
//	只取得整數的除法
%	取餘數

- 所有數學運算符都可以混合使用
- 遵守四則運算規則

Try it !!!

```
>>> 17 / 3
```

```
5.666666666666667
```

```
>>> 17 // 3 # 去除小數點後的數值
```

```
5
```

```
>>> 17 % 3 # 算餘數
```

```
2
```

```
>>> 2*2*2*2*2
```

```
32
```

```
>>> 2**5 # 2 的 5 次方
```

```
32
```

# 另一種使用數學運算子的方法

- 在程式中，除了正常的數學運算子使用方法以外  
還可以透過以下特別的數學運算子達到相同的計算結果
- 語法

- `a += b` #  $a = a + b$
- `a -= b` #  $a = a - b$
- `a *= b` #  $a = a * b$
- `a /= b` #  $a = a / b$
- `a //= b` #  $a = a // b$
- `a %= b` #  $a = a \% b$
- `a **= b` #  $a = a ** b$

The screenshot shows a Jupyter Notebook interface with two code cells. Cell [1] contains assignments: `a = 123` and `b = 456`. Cell [2] contains the following code:

```
In [2]: print(a)
print("====")
a += b
print(a)
```

The output of Cell [2] is:

```
123
=====
579
```

A new input cell, In [ ], is visible at the bottom.

# Quiz

請觀察右圖中的程式碼。

請問在 In [4] 的區塊中，`print(a + b)` 的結果應該為？

- A. 4
- B. 6
- C. 7
- D. 9

The screenshot shows a Jupyter Notebook interface with three code cells:

- In [1]: `a = 1`  
`b = 3`
- In [2]: `print(a + b)`  
4
- In [3]: `a = a + 3`  
? ? ?

The final cell, In [3], has three question marks indicating an unexpected or undefined result.

# Ans

請觀察右圖中的程式碼。

請問在 In [4] 的區塊中，`print(a + b)` 的結果應該為？

- A. 4
- B. 6
- C. 7
- D. 9

The screenshot shows a Jupyter Notebook interface with four code cells:

- In [1]: `a = 1`  
`b = 3`
- In [2]: `print(a + b)`  
4
- In [3]: `a = a + 3`
- In [4]: `print(a + b)`  
7

The cell In [4] is currently active, indicated by a blue vertical bar on the left.

# Quiz

你編寫了一個程式碼進行數學運算

```
a = 22  
b = 8  
  
print(a/b)  
print(a//b)  
print(a%b)
```

請寫下該程式碼運行的結果

# Ans

你編寫了一個程式碼進行數學運算

```
a = 22  
b = 8  
  
print(a/b)  
print(a//b)  
print(a%b)
```

請寫下該程式碼運行的結果

2.75

2

6

# 轉換資料型態

- 在符合特定條件下， 資料型態之間可以互相轉換
- 例如一個**string** 是 “123” ，  
因為都是由數字組成  
所以可以轉換成 **int** 或是 **float**

```
In [5]: numStr = "123"
```

```
In [6]: type(numStr)
```

```
Out[6]: str
```

```
In [7]: numInt = int(numStr)
numFloat = float(numStr)
```

```
In [8]: print(type(numStr))
print(type(numInt))
print(type(numFloat))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

# 轉換資料型態

但如果字元串中參雜了英文，如 “123a” 在轉換成 int 時就會出現錯誤

```
In [9]: numStr = "123a"
```

```
In [10]: numInt = int(numStr)
```

```
-----  
ValueError
```

```
Traceback (most recent call last)
```

```
<ipython-input-10-0beb548463ae> in <module>
```

```
----> 1 numInt = int(numStr)
```

```
ValueError: invalid literal for int() with base 10: '123a'
```

# Quiz

下列三種的結果分別為何？

- A. `print(3+7)`
- B. `print( "3" +7)`
- C. `print( '3' +' 7' )`

# Ans

下列三種的結果分別為何？

- A. 10
- B. TypeError: can only concatenate str (not "int") to str
- C. 37

# 格式化輸出 – print()

# 更豐富的輸出

- **print(...)** 可以將括號內的字元串、數字或變數等輸出到畫面上
- 但一個**print(...)** 所能顯示出的內容不僅限於單個字元串或單個數字
- 可以在一個 **print(...)** 中，同時顯示出多個東西

```
In [1]: print(7, "is lucky number.")  
7 is lucky number.  
  
In [2]: print("I am", 18, "years old.")  
I am 18 years old.  
  
In [3]: print("123 * 456 =", 123 * 456)  
123 * 456 = 56088  
  
In [4]: print("I have $", 100, ".")  
I have $ 100
```

# 更豐富的輸出

如果想要同時輸出多個東西，但不想以空白連接時該怎麼辦呢？

Ex: `print("I have $", 100, ".")`

```
In [4]: print("I have $", 100, ".")
I have $ 100 .

In [5]: print(''')
I have $100.

In [ ]:
```

# 更豐富的輸出

如果想要同時輸出多個東西，但不想以空白連接時該怎麼辦呢？

Ex: `print("I have $", 100, ".")`

最簡單的做法就是通通轉換成字元串 (`str()`) ,再連接起來 (+)

```
In [4]: print("I have $", 100, ".")
I have $ 100 .

In [5]: print("I have $" + str(100) + ".")
I have $100.

In [ ]:
```

# 格式化字串

客製化你的輸出格式

## ➤ %: 字串插值 (string interpolation)

舊式格式化, 與C語言類似

```
print("Hello %s %s" % (text1, text2))
```

- 可讀性低

## ➤ str.format: 字串函式 (string format)

新式格式化, 與Jinja(模板)類似

```
print("Hello {0} {1}".format(text1, text2))
```

- 當變數太多時, 程式碼會過長

## ➤ f-string: (Formatted String Literals)

新式格式化, 效能&可讀性最好

```
print(f"Hello {text1} {text2}")
```

使用者輸入

# 輸入

- 使用函式 `input()` 讓使用者可以用鍵盤輸入字元串，按下 [Enter] 送出
  - 透過 `input()` 輸入的字元串一律被視為「字串」
- 
- 語法：
    - `input()`
    - 變數 = `input()`

```
input()
test
'test'

a = input()
10

type(a)
str
```

Try it !!!

# Quiz

您正在創建一個商品動態清單，它接受使用者的輸入，並儲存資料

```
thing = input("輸入商品名稱:")  
number = input("輸入購買數量:")
```

輸出資料時，必須符合以下要求：

- 字串必須在雙引號內
- 數字不得使用引號或其他字元框起來
- 每個項目必須用逗號隔開

請選擇符合條件的程式碼片段

```
print(' "{0}" , {1}'.format(thing, number))  
print( '{0}, {1}'.format(thing, number))  
print(thing + ',' + number)  
print("%s, %s" % (thing, number))  
print(" " + thing + ", " + number)
```

# Ans

您正在創建一個商品動態清單，它接受使用者的輸入，並儲存資料

```
thing = input("輸入商品名稱:")  
number = input("輸入購買數量:")
```

輸出資料時，必須符合以下要求：

- 字串必須在雙引號內
- 數字不得使用引號或其他字元框起來
- 每個項目必須用逗號隔開

請選擇符合條件的程式碼片段

```
Y print(' "{0}", {1}'.format(thing, number))  
print(' {0}, {1}'.format(thing, number))  
print(thing + ', ' + number)  
Y print('%s, %s' % (thing, number))  
Y print("'" + thing + "', '" + number)
```

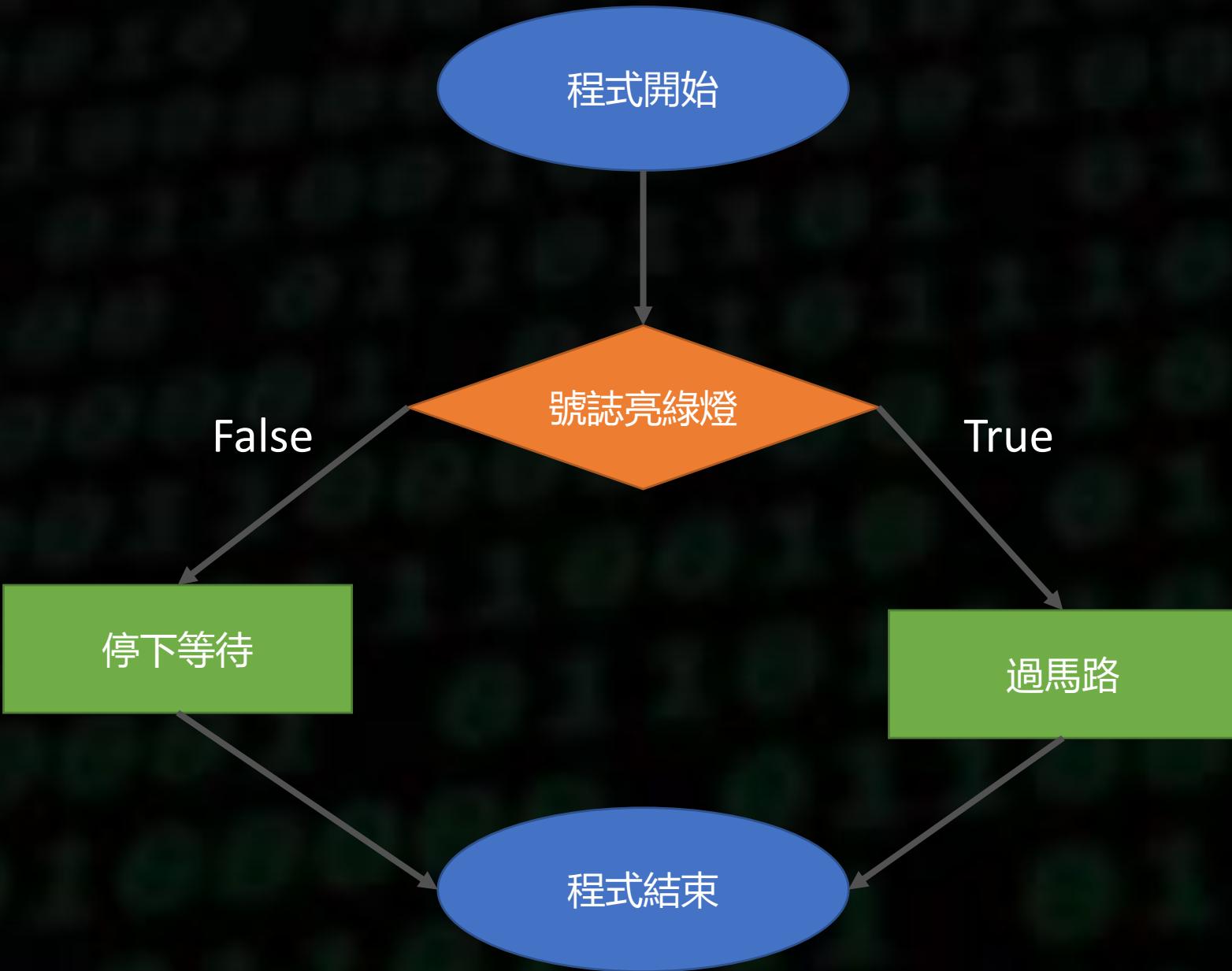
# 課後練習

已知氣溫換算公式如下：

$$\text{攝氏} = (5/9) * (\text{華氏溫度} - 32)$$

試設計一告系統，讓使用者輸入攝氏氣溫，可以得到華氏氣溫  
(備註：小數點只保留後2位)

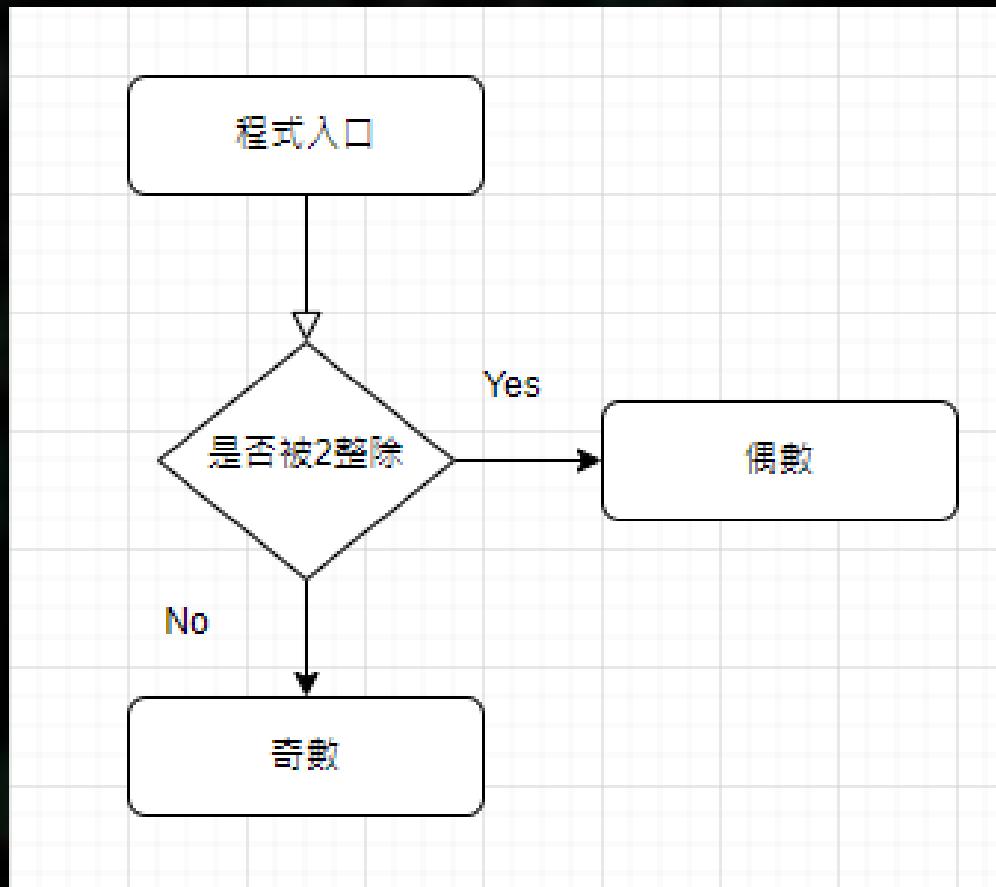
# 流程敘述



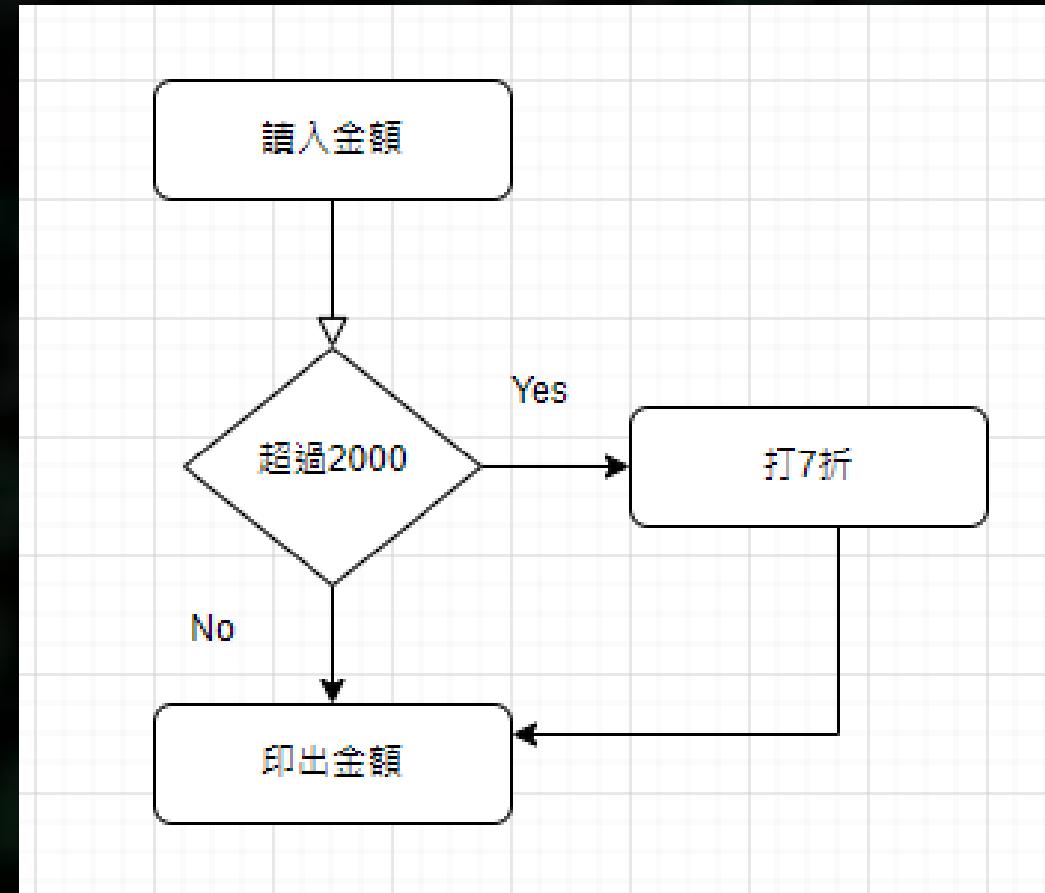
**if** (條件式):

  程式區塊

#若條件成立，則執行該程式區塊

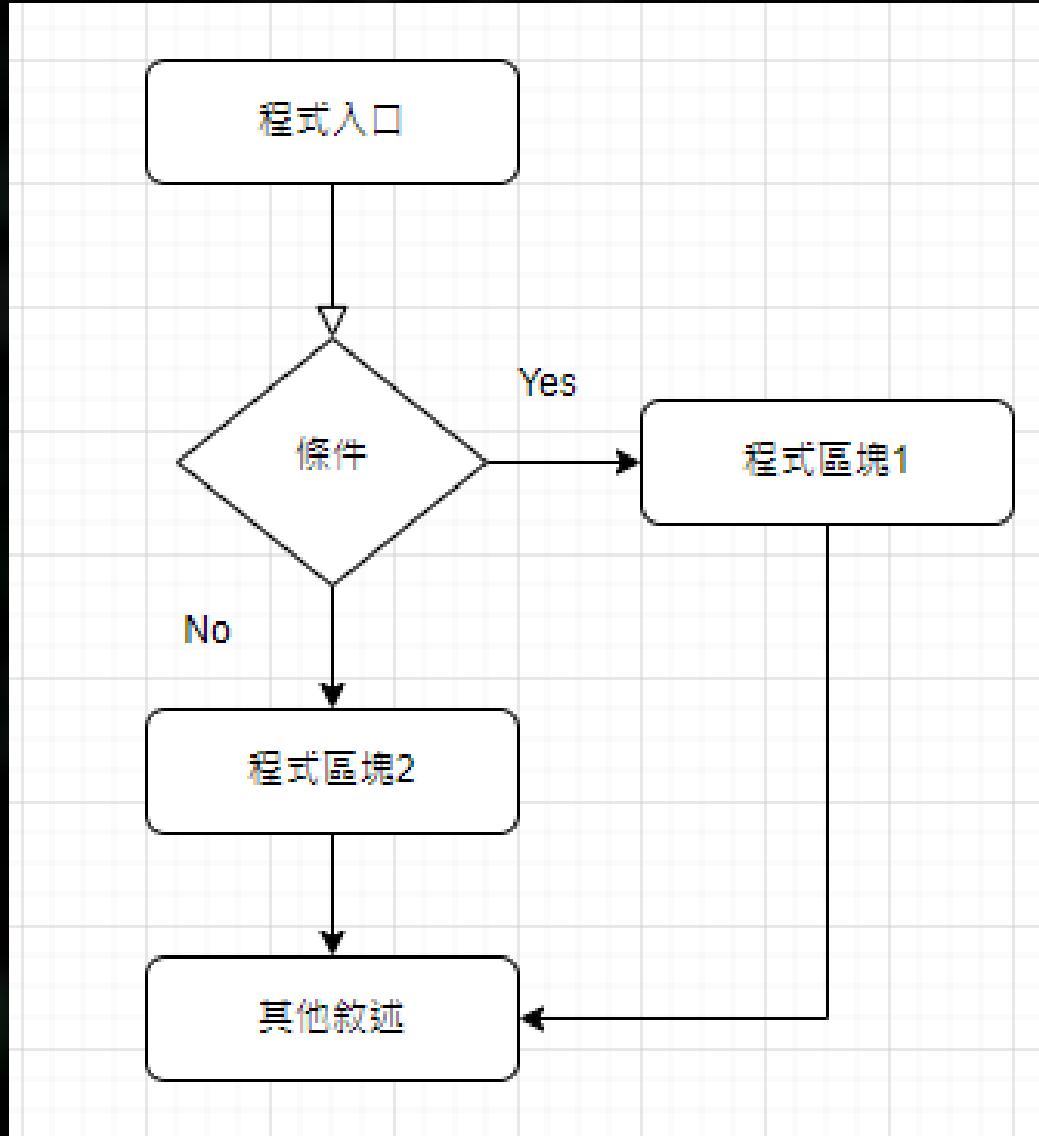


米勒百貨公司周年慶，  
公司決定在周年慶期間對消費超過2000元的顧客打7折，  
請設計一個收銀台程式，輸入顧客購買金額後，  
再計算實際要付的錢



```
if (條件式):  
    程式區塊 1  
else:  
    程式區塊 2
```

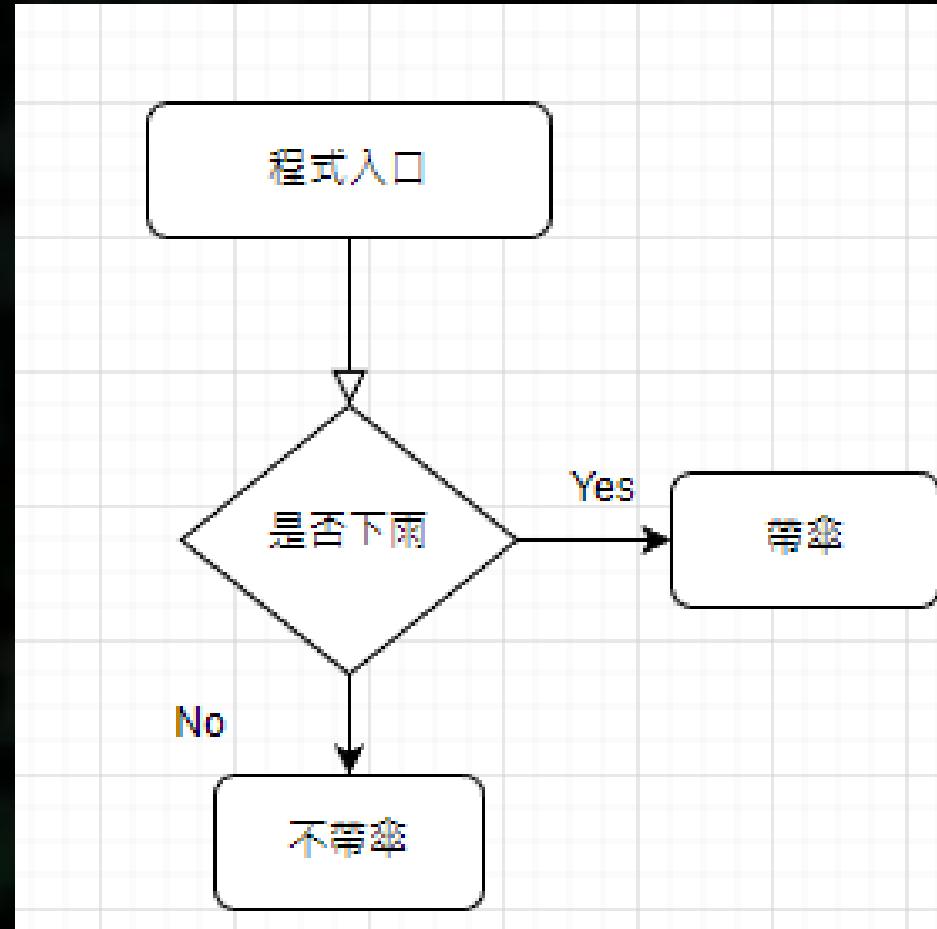
#若條件成立，則執行程式區塊1  
#反之，執行程式區塊2



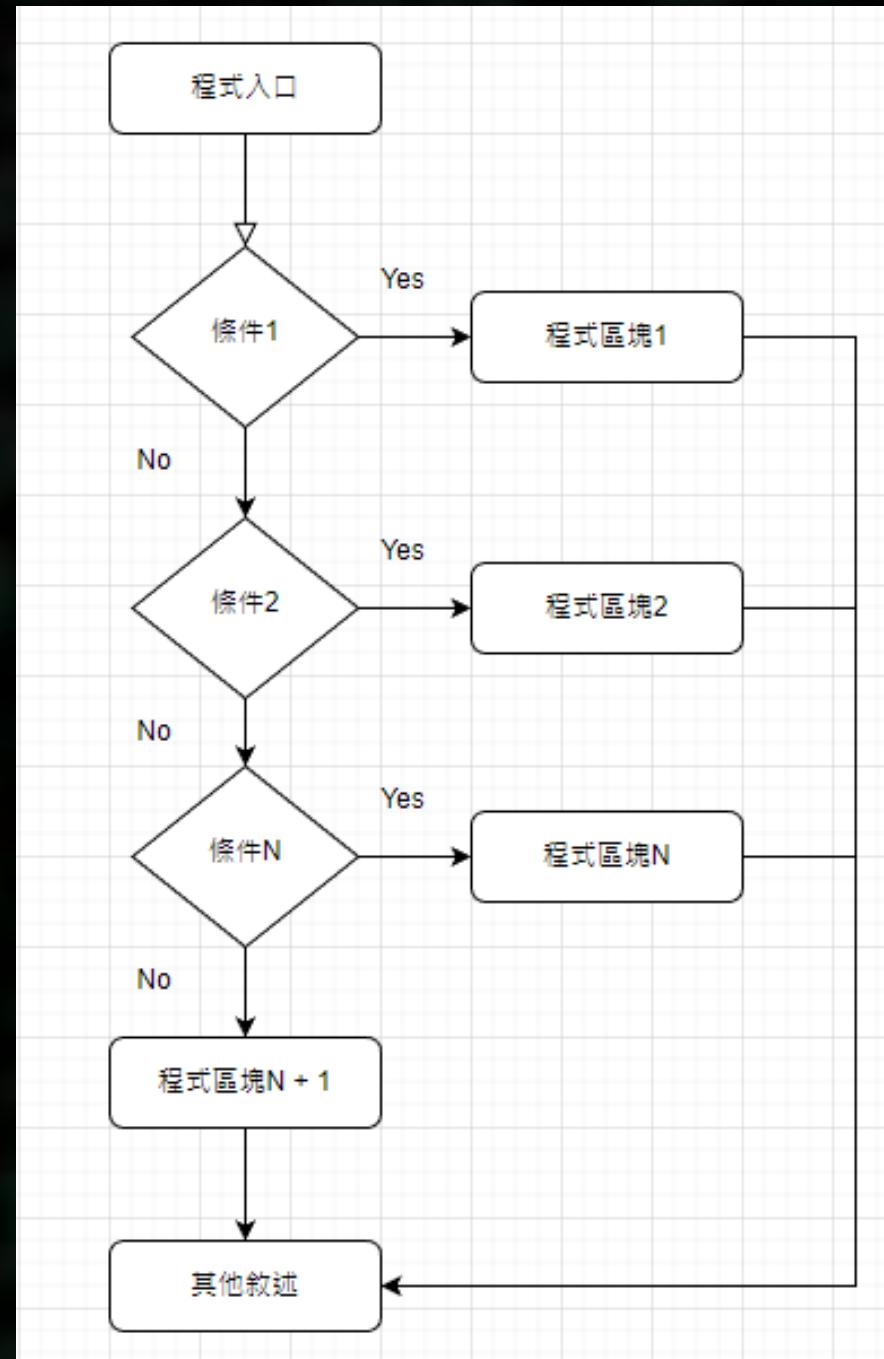
請寫出一個程式，

由使用者輸入今天是否下雨，

若下雨，則由程式提醒帶雨傘；反之則不帶傘



```
if condition 1:  
    #<若 condition 1 為真， 則執行此區塊>  
elif condition 2:  
    #<若 condition 2 為真， 則執行此區塊>  
...  
else:  
    #<若條件都不成立， 執行程式區塊N， 則執行此區塊>
```



請依據使用者輸入的成績，判定成績等第

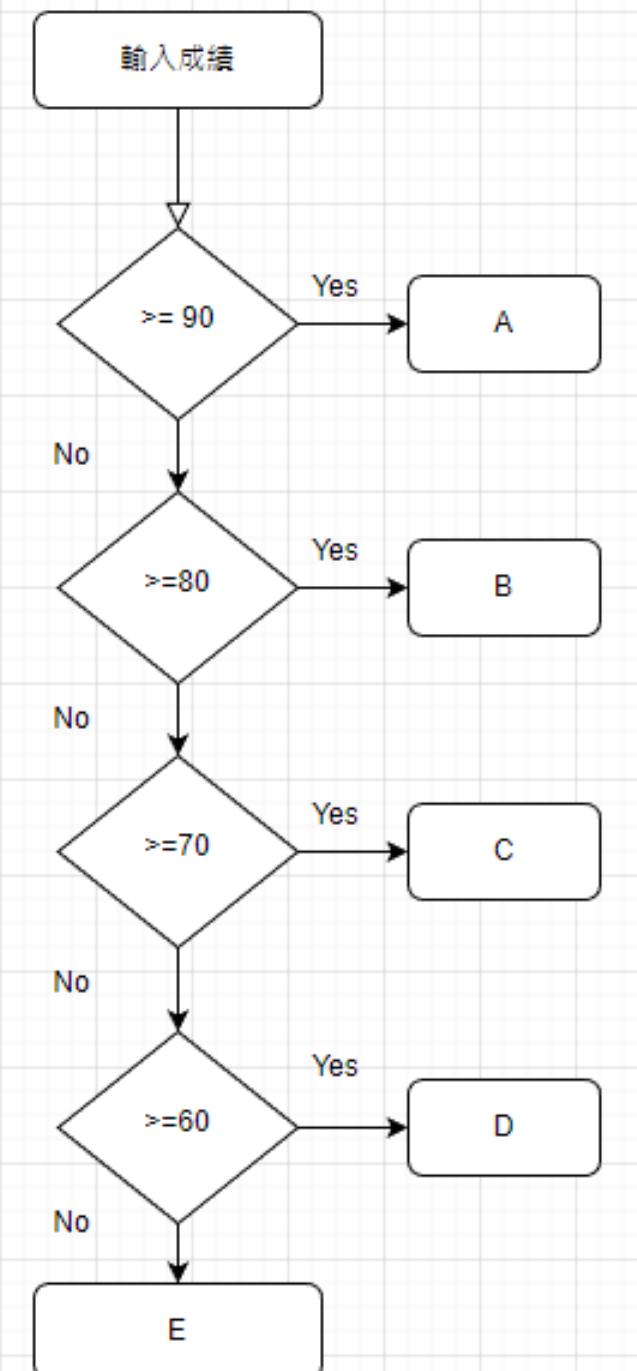
90以上: A

80 - 89: B

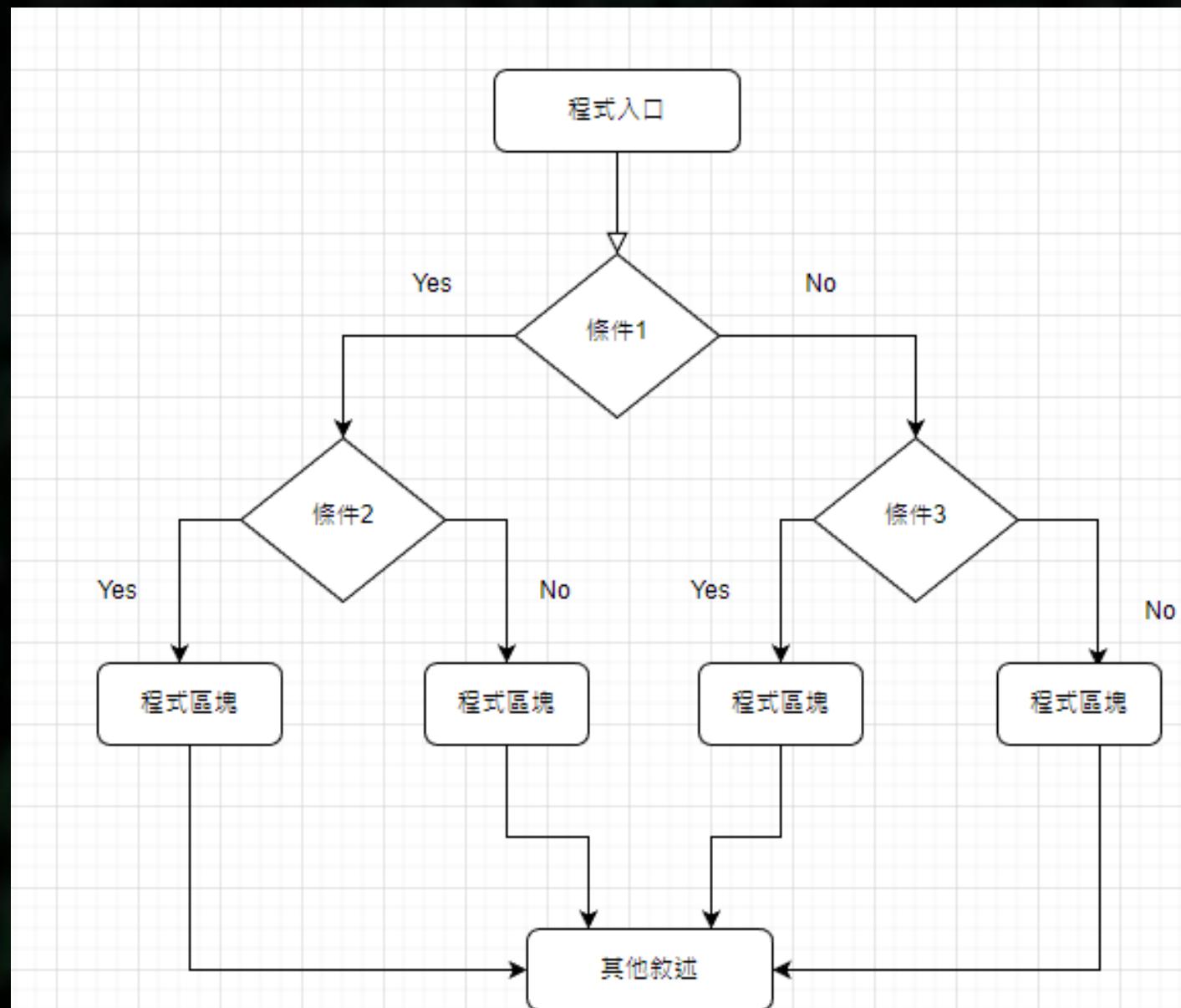
70 - 78: C

60 - 69: D

未滿60: E



```
if condition a1:  
    #<若 condition a1 為真，則執行此區塊>  
    if condition b:  
        #<若 condition b 為真，則執行此區塊>  
    else:  
        #<若 condition b 為假，則執行此區塊>  
  
elif condition a2:  
    #<若 condition a2 為真，則執行此區塊>  
    if condition c:  
        #<若 condition c 為真，則執行此區塊>  
    else:  
        #<若 condition c 為假，則執行此區塊>  
  
else:  
    #<若 a1 & a2 皆為假，則執行此區塊>
```



請寫出一個程式，讓使用者輸入西元年分，  
判斷該年是否為閏年  
判斷方式：四年一閏，逢百不閏，逢四百又閏

ex:

輸入 2000 : 顯示 Leap Year

輸入 2100 : 顯示 Not leap Year

