Python基本語法

地表最狂最有趣的基本語法

Scan me

周凡剛(Elwing)著

1 環境設置

1.1 電腦是什麼

對於我來說，電腦就是有很多『開關』組合起來的結合體，也就是由無數個『開』(大家說的 0) 或者『關』(大家說的 1) 組合起來的機器

所以大家會說電腦是個只懂 0 和 1 的機器

- 一個0或者1我們也會稱為一個bit
- 我們通常會用8個bits為一組，稱作1個byte
- 更多單位(使用二進位): 1KB = 1024bytes, 1MB = 1024KB
- 但在買硬碟的時候，硬碟廠商用的是1K = 1000，所以作業系統會看到比較少

1.2 程式是什麼

程式語言根據他的演化歷史大概分成三個階段，由機器語言 -> 組合語言 -> 高階語言

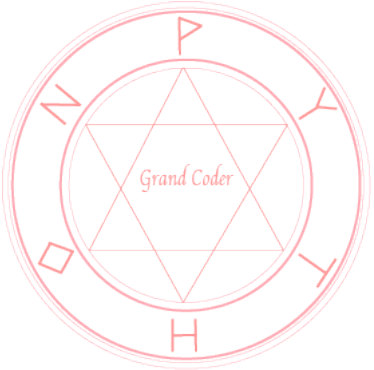
1.2.1 機器語言

機器語言是最初的的程式語言，就直接使用 0 和 1 來撰寫程式

1.2.2 組合語言

大家發現根本不可能直接用 0 和 1 來寫程式，所以 CPU 廠商制定了一套指令集，這些指令集由 CPU 負責『翻譯』成機器語言

- 所有的程式指令都是通過CPU來執行
- 組合語言例子: MOV R5 R6(把R6暫存器的值複製到R5)



1.2.3 高階語言

組合語言還是太基本，不適合人類直接撰寫，所以我們開發出了高階語言
你今天所有聽到的 C, Python, JAVA, Swift 都屬於高階語言的範疇
所以在執行程式的時候一定要做的就是要有個翻譯器把『高階語言』翻譯成『組合語言』

- 有各種的翻譯方式，直譯(執行時在翻譯)或者編譯(預先翻譯好)
- 不管什麼程式語言都要安裝對應的翻譯器(C++: GCC, Python: Python2/3)

1.3 需要安裝

不管我們在練習何種程式語言，我們都一定要安裝下面幾個軟體

1.4 翻譯器

不管寫什麼程式，我們一定需要把翻譯器安裝起來，Python 的翻譯器分成兩個系列

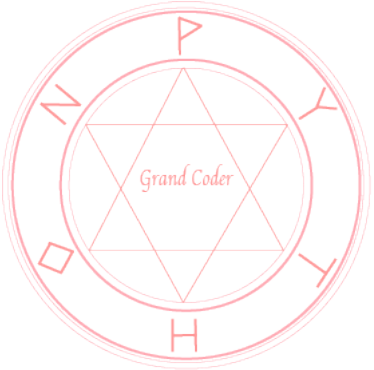
1. Python 2 系列: 已經停止版本更新，只會修正 bug，最後一個版本是 2.7

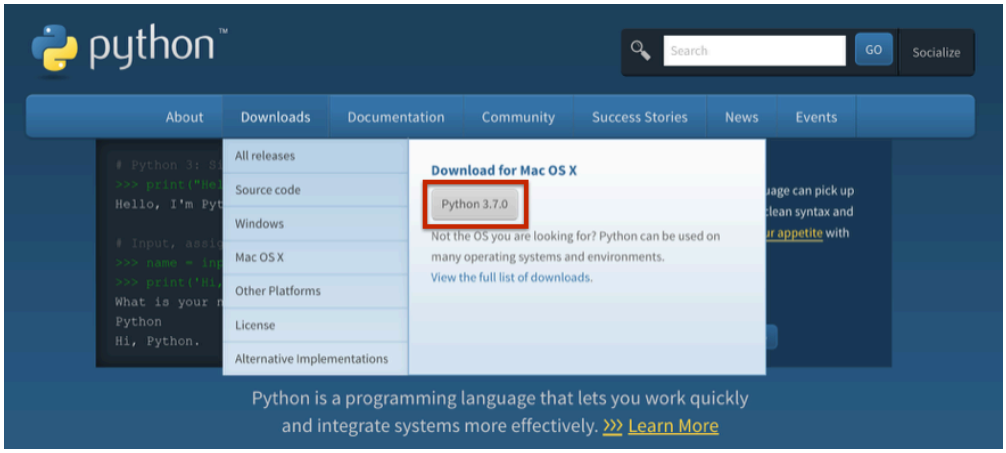
2. Python 3 系列: 跟 Python 語法有些許的不相容，目前還在更新中

那我們到底要安裝哪個版本呢?
這個市面上眾說紛紜的問題，我認為正確的答案是:『你想使用的函式庫 (工具) 最高支援哪一版本，你就只能用到那個版本，因為 Python 是個函式庫 (工具) 為主的語言』
所以當你遇到你覺得你的程式碼無論如何都沒錯，但卻一直執行錯誤的情況，我會建議你把你的 Python 翻譯器版本換成較為舊的版本
如果你不知道有什麼比較穩定，幾乎支援所有函式庫的版本，這裡推薦兩個版本

1. Python 2 穩定版本: 2.7. 最新版本
2. Python 3 穩定版本: 3.4.3

通常如果沒遇到上述的狀況，我們就可以安裝最新版本
請到 https://www.python.org/ 滑鼠移到 download 上，就有最新版本的下載點





下載完請一路安裝到完

1.4.1 翻譯器 32bit v.s. 64bit (Windows 必看)

事實上我們的作業系統有分成舊型 32bit(你也會看到 x86 這種稱呼) 和 64bit(你會看到 x64 這種稱呼)

那軟體就也有分 32bit 和 64bit 適用的

那你就會有個疑問了，我們的翻譯器是 32bit 還是 64bit 呢?

在回答這個問題前，我們先畫一個表來看 OS 和軟體的適配程度

軟體可以在作業系統上可以執行我們就打 O，否則打 X

能否執行	軟體 32bit	軟體 64bit
作業系統 32bit(通常 XP)	O	X
作業系統 64bit(通常 XP 以後)	O	O

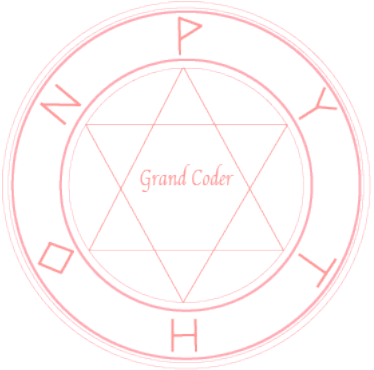
你會發現唯一不行的情況是新軟體配上舊的作業系統

那我們有時候做的功能會跟這有關係 (e.g. 打包成 exe)，你打包成的 exe 就跟你用的翻譯器是同一版本

所以如果你是 windows 的同學我會建議你先安裝 32-bit 版本 (事實上也是官網的預設值) 以防遇到 XP 的作業系統

那如果是其他作業系統那就不用想的選 64-bit 版本了

另外，如果在使用比較特別的函式庫 (e.g. TensorFlow, Keras) 也會被強迫使用 64-bit 的版本



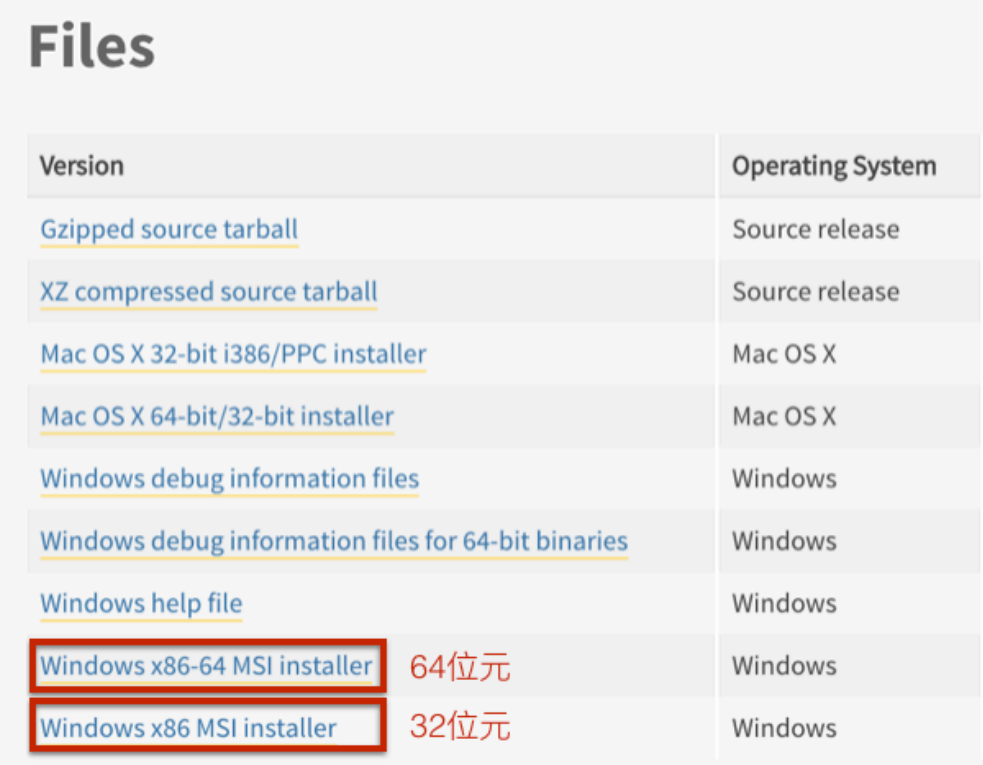
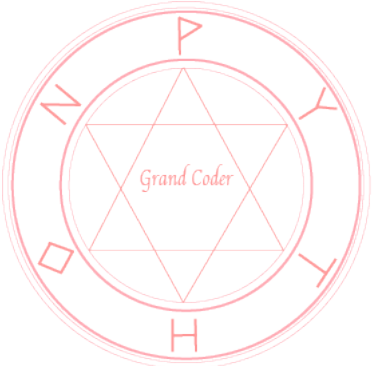


圖: 3.4.3 官網版本

1.5 輔助開發工具

有了翻譯器，其實我們已經可以開始寫程式了
不過我會建議安裝個輔助開發工具來幫你開發
這邊我選用 PyCharm 作為我們主要的輔助開發工具
因為 PyCharm 比較輕量級，而且可以好好地幫助你瞭解環境
首先先到 https://www.jetbrains.com/pycharm/ 官方網站下載 community 版本
接著安裝步驟如下



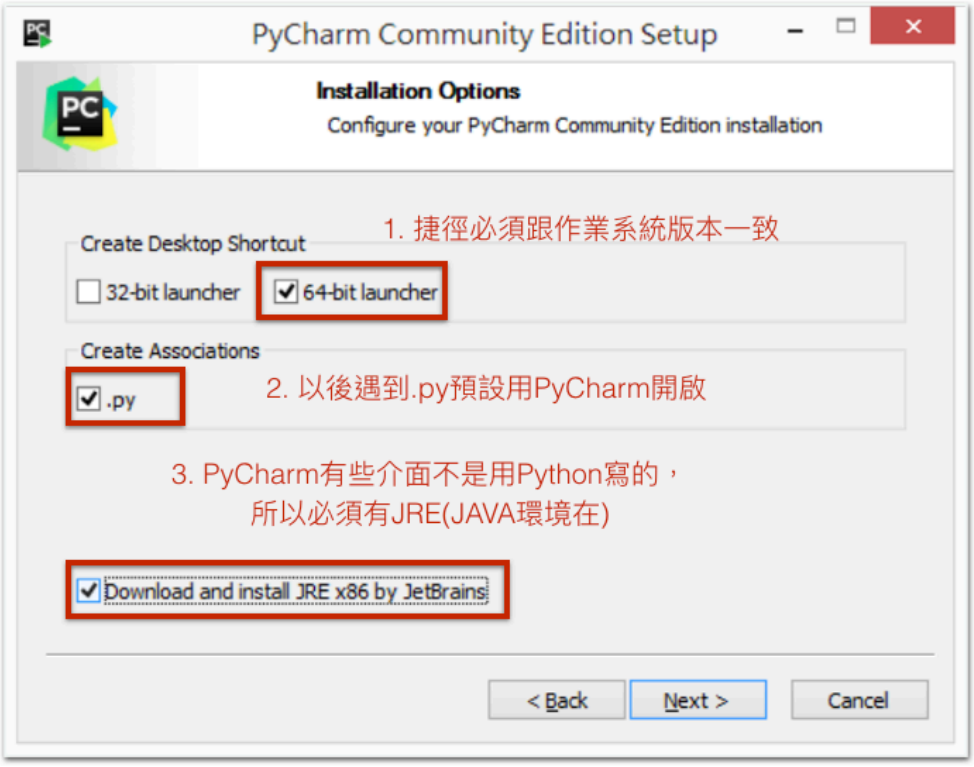
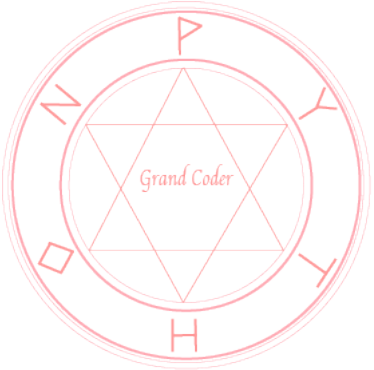


圖: 這裡要記得做出選擇

選好以後啟動程式



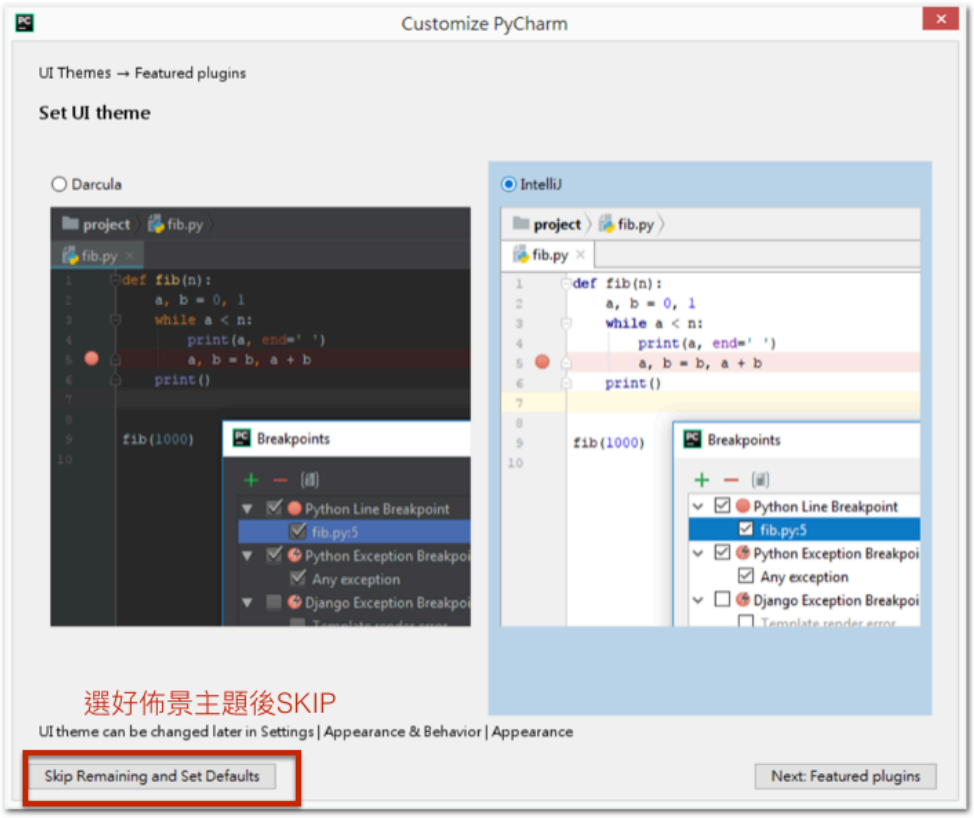


圖: 選完佈景主題 skip

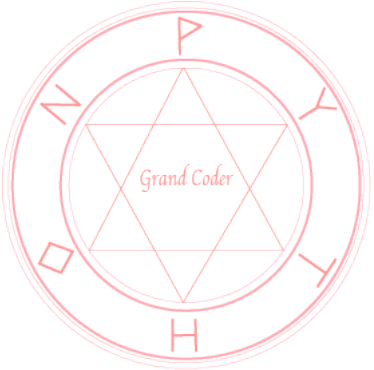
1.6 (同場加映) 線上環境

如果你是在有防火牆或者出外不方便的環境，我會建議你可以參考一下以下這個線上環境，但平常練習還是以線下環境為主

https://repl.it/repls

他有幾點好處

1. 所有東西寫完就存在雲端
2. 不用費神安裝環境
3. 你可以直接把網址列上的網址分享給別人



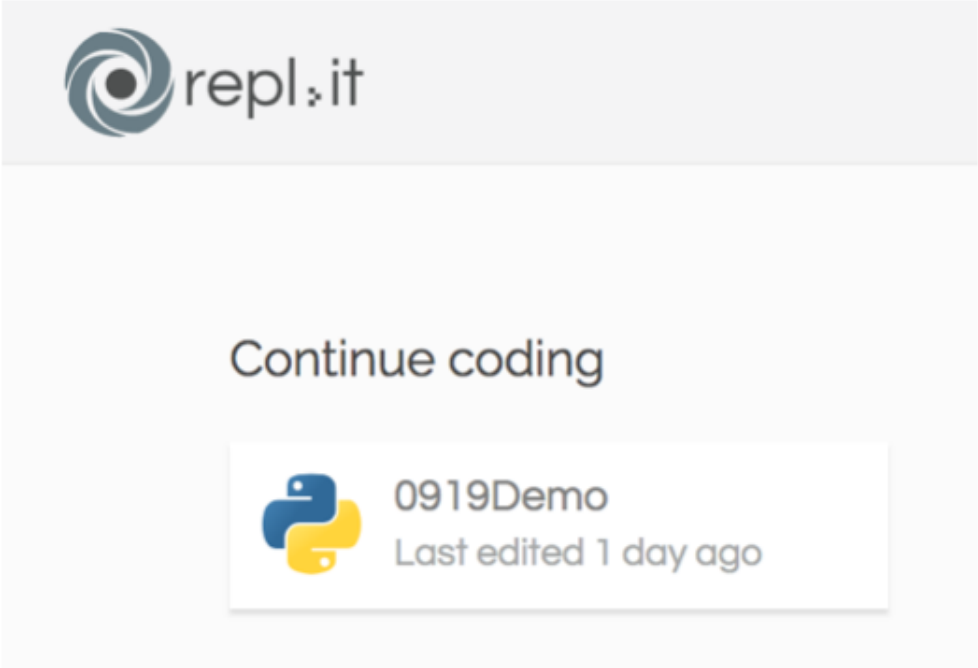


圖: repl 網站

1.7 第一個程式

1.7.1 開啟新 Project

首先在你開起來的 PyCharm 上選擇 Create New Project(第二次以後在 File)

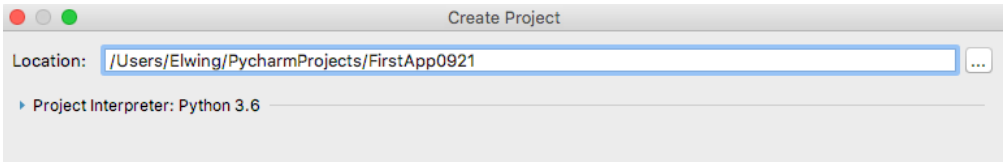


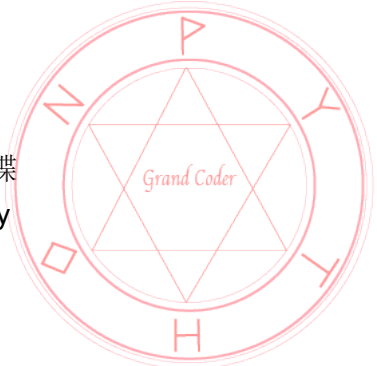
圖: 把你的 APP(Project) 開起來

在名稱那欄位改成你想要的名字，這裡有幾個『不要』

1. 不要中文
2. 不要空白鍵 (在命令列的時候容易被當成分隔符號)
3. 不要全形

1.7.2 開啟 Python 執行檔

Python 的構造很像我們一個遊戲光碟，APP(Project) 就是這光碟裡面可以有很多執行檔，不過這些執行檔不叫.exe，而是叫做.py



對 Project 點擊右鍵 -> New -> Python File

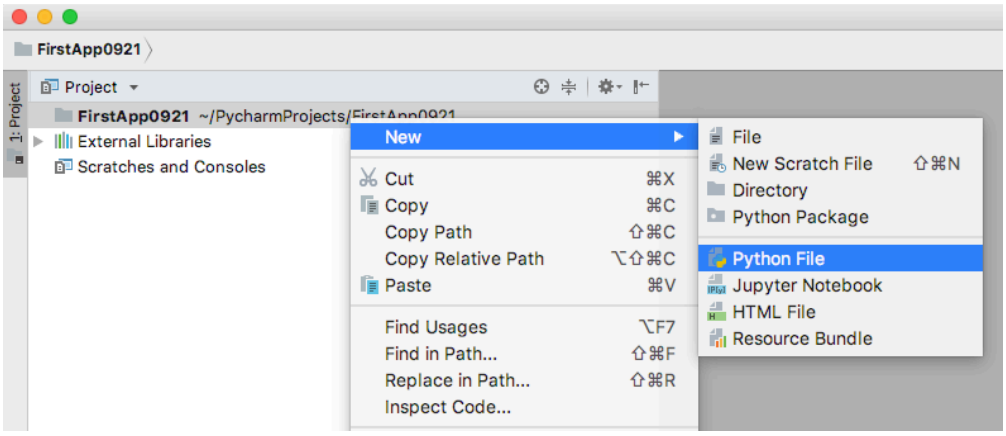


圖: 開啟方式

那我們通常第一支執行檔會命名為 main.py(主要的執行檔)

1.7.3 撰寫第一支 Python 程式

請在你的 main.py 區域幫我打上兩行

執行結果如下

```
[程式]: print(3 + 2)
        print("hello")

5
hello
```

1.7.4 基本概念

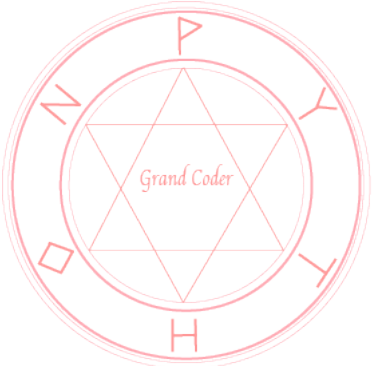
首先，我們要先知道程式的世界其實很簡單

他向我們的現實世界一樣，一樣有些基本的構成粒子!

這裡我們看到三種基本粒子中的兩種，數字 (直接寫就可以)，和文字 (要用雙引號或者單引號括號起來)

我們看到了兩種對於基本資料的操作，運算 (+)，和功能 (print 就是列印功能，我們把我們基本資料丟給他請他操作)

- 學程式的時候要盡量把程式的符號用中文去對照
- print後面的()在中文就是來承接丟進來資料的意思



1.7.5 註解

『程式碼是寫給別人，寫給未來的自己看的，而不是只是寫給電腦看的』
所以建議可以在一大段程式碼前面加個說明，讓自己回頭看的時候能看懂
只要在一行前面加個 **#**，這行就不會被當成程式處理，而是當成一個說明處理

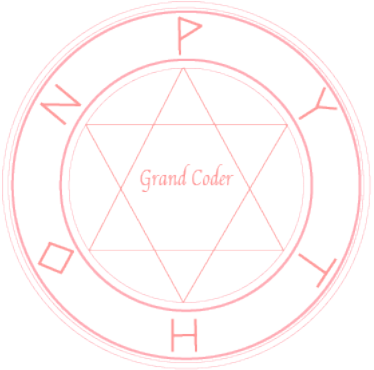
```
[程式]: # 這是註解: 練習一下基本資料
        print(3 + 2)
        print("hello")
```

```
5
hello
```

1.7.6 程式碼風格

最後在結束環境架設前，要跟各位討論一個很重要的事
『寫程式碼就像在寫文章，你的排版良好，大家一定會更喜歡看你的文章』
那可以建議初學者在一開始可以學習我的風格就可以了
有些基本風格會在後面的程式碼看到!

- 我在運算符號(e.g. +)左右兩邊會空出一格 -> 才不會太擁擠
- 我在逗號後也會空一格 -> 如同我們在寫英文一樣
- 兩大段的程式碼之間我會放下一個空行 -> 如同我們平常的分段



2 基本資料

這章節我們要就基本的資料來個全面的介紹

2.1 等號

在講數字之前，我們要先來看一個特別的符號 (=)

在現實的世界，= 這符號有兩個意思，1) 給名字: x = 3 2) 比較相等: 5 = 3 + 2

但在程式的世界，我們不能全部都要，於是我們在程式裡，= 只有『給名字的意思』

❤ 語法: 名字 = 資料

❤ 注意: 名字一定放在 = 左邊，資料一定放在 = 右邊

❤ 時機: 當你一個資料要用很多次的時候，給個名字就能重複使用

2.2 數字

數字比較需要注意的地方是其實還是分成小數和整數，Python 會依照你寫的型態自動幫你歸類

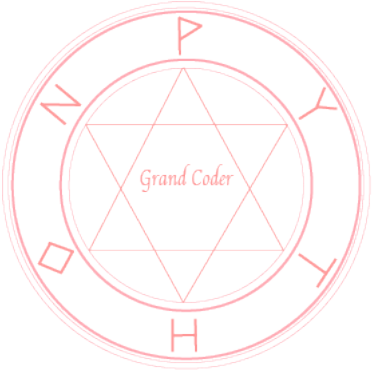
這裡比較要注意的是小數由於是使用科學記號法 (e.g. 十進位的科學記號法 $a \times 10^b$)，不過電腦是用二進位的科學記號法

由於科學記號法就會有無窮小數被『截斷』的問題，所以我們要知道小數是會有誤差的，這我們是覺得 OK 的 (e.g.50% 和 50.000001% 在普通的時候根本沒差)

初學者一定要在這裡習慣這件事! (如果需要精確運算，通常是計算機的時候，我們會使用 Decimal)

2.2.1 基本運算

這裡先介紹一點基本的操作



符號	用途
+	加法
-	減法
*	乘法
/	除法
%	做完除法取餘數
//	做完除法取整數
**	次方

[程式]:
```python
# = 應用: 算出數字, 給名字 a
a = 3 + 3.14
print(a)

# 重複利用 a, 並把結果叫做 b
b = a * 2
print(b)

# 進階! 非常重要!
# 把結果重新設定到 a 名字背後
# 那舊的自然就不見了, 而是換成新設定的東西
a = a * 3
print(a)
```

```
6.140000000000001
12.280000000000001
18.42
```

2.2.2 運算的禁忌

事實上在程式語言的基本，我們是不能把不同類型的東西拿來做運算的
譬如你寫下

```python
print("hello" + 3)
```

你就會看到

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-166b09aa1854> in <module>()
----> 1 print("hello" + 3)

TypeError: must be str, not int
```

那大家會有兩個問題

Q1. 為什麼其他程式語言可以呢?

A1. 因為那些程式語言偷偷先幫你轉換成同樣的類型，再開始做運算，Python 本質上是一個很嚴格的語言，所以他絕對不會幫你做偷偷轉換這件事!

Q2. 那如果我想印得漂亮一點，該怎麼做呢?

A2. print 這個技能可以帶入多個參數 (為何可以我們後面再說)，請看下面

```
[程式]: # 絕對不行, 不同類型
        # print(" 數字:" + 3.14)
        # 帶入多個參數
        print("數字:", 3.14)
```

```
數字: 3.14
```

2.2.3 基本技能

介紹一點基本的技能操作

abs(x)	絕對值
ceil(x)	比 x 大的最小整數
floor(x)	比 x 小的最大整數
pow(x, y)	x 的 y 次方
round(x [,n])	四捨五入，如果有帶入第二個參數，則會四捨五入到那個位數
sqrt(x)	x 的平方根

這裡要順便教各位一下什麼是技能

技能 (函數) 由三個部分構成:

1. 技能名字: 像 abs 這個技能的名字就是 abs

2. 參數 (技能需要的資料): 像絕對值這種技能是不可能單獨執行的，因為沒傳入東西不知道對誰做絕對值，用 () 來丟入需要資料

3. 回傳值 (回傳答案): 像 abs 做完就會有一個答案，但是像 print 這種技能是沒回傳答案的 (就像辦公室的印表機，一直印一直印，你不會停下來等答案)

```
[程式]: # abs 有回傳值, 我們可以直接用舊名字 a 來接他
        a = abs(-2 * a)
        print("絕對值:", a)

        # pow 需要兩個參數, 兩個參數之間我們用逗號隔開
        a = pow(a, 2)
        print("次方運算:", a)

        # bmi: (體重) / (身高公尺) ^ 2
        # 其實也可以使用 ** 來做次方
        bmi = 75 / pow(1.75, 2)
        print("BMI:", bmi)

        # 四捨五入到小數第二位
        bmi = 75 / (1.75 ** 2)
        print("BMI(四捨五入到小數第二位):", round(bmi, 2))
```

```
絕對值: 36.84
次方運算: 1357.1856000000002
BMI: 24.489795918367346
BMI(四捨五入到小數第二位): 24.49
```

2.2.4 (進階) 精確運算

這裡初學者不用練習，當要使用精確運算的話，我們會使用 Decimal
但事實上，使用時機非常非常少，大概只有計算機 APP 需要

```
[程式]: from decimal import *
        print(Decimal("3.14") + Decimal("3"))
```

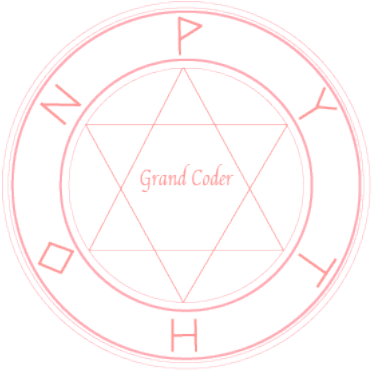
```
6.14
```

2.3 文字

文字就是一般人所說的『字串』，你需要使用『雙引號』或者『單引號』包起來
有時候你要特別注意：像 0912345678 這種電話不該是數字形態，而應該是文字型態 (1. 你不會去掉前面的 0 2. 你不會用幾萬幾千去唸他)

2.2.1 運算

文字支援 + 來做兩個文字的串連運算



2.3.2 技能

最常使用的技能就是 len 來算長度了

```
[程式]: a = "hello"
        print("原本:", a)
        a = a + "python"
        print("串連:", a)
        print("長度:", len(a))
```

```
原本: hello
串連: hellopython
長度: 11
```

2.3.3 特殊的操作

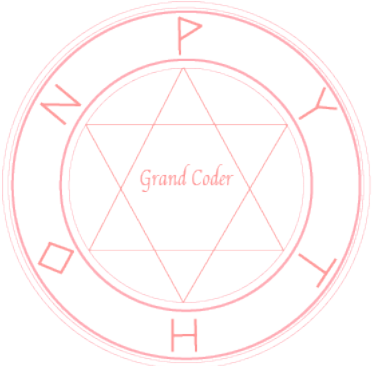
你可以透過做好來取得單一的字
假設我們創造了個 HELLO 這文字
你可以用正向座號或反向座號來取得對應的字
平常我們使用正向座號就可以了
這裡要注意一下：座號是從 0 開始，跟平常我們熟悉的 1 不太一樣

文字	H	E	L	L	O
正向	0	1	2	3	4
反向	-5	-4	-3	-2	-1

❤ 單一個字語法: 字串名 [座號]
❤ 多個字語法: 字串名 [想拿座號 1: 想拿座號 2+1] (第二個數字不包括，要多 +1)

```
[程式]: # 記得座號從 0 開始
        # [座號] -> 拿到背後的字
        a = "hello python"
        print("第三個字:", a[2])
        # 利用反向座號直接拿最後一個字, 就不用寫 a[len(a) - 1] 了
        print("最後一個字:", a[-1])
        # 也可以拿某一個字到某一個字
        # 要記得第二個數字要多 +1
        print("第三個字-第七個字:", a[2:7])
```

```
第三個字: l
最後一個字: n
第三個字-第七個字: llo p
```



2.3.4 (進階) 間隔拿取操作

我們可以在拿取子字串的操作中使用第二個: 來決定要間隔幾個字拿取一次

```
[程式]: a = "hello python"
        # 2->4->6->8
        print(a[2:9:2])
        # 如果前面什麼都不寫代表從頭, 後面什麼都不寫代表尾巴
        print(a[:])
        # 從頭到尾, 並且三個一拿
        print(a[::3])

lopt
hello python
hlph
```

2.3.5 逃脫字元

另外有一些單字是無法直接打出的，譬如: Enter 的換行 (被編輯器變成下一行程式)，TAB(被編輯器變成多個空白)

這時候我們會用 (逃脫字元) + 一個特殊的字來代表這個無法打出來的字

特殊符號	對應功能
\n	換行
\t	TAB
\b	BackSpace

```
[程式]: print("第一行\n第二行")

第一行
第二行
```

2.3.6 專屬技能

文字還有一個特別的使用方式，叫做專屬技能

白話文: 只有文字才能使用的技能

❤ 語法: 文字. 專屬技能 (參數)

❤ 注意: 這裡的. 就是我們中文『的』意思，也就是使用這個文字『的』一個專屬技能

❤ 參考: 請 Google “Python String method”，建議參考 https://www.tutorialspoint.com/python/python_strings.htm，最下面的 Built-in String Methods

我們先來一起練習其中的幾個

第一個: replace(“舊字串”, “新字串”, (選用) 要取代幾個)

```
[程式]: # 先準備一個字串
        a = "hellohellohello"
        # 使用 replace(取代的專屬技能)
        b = a.replace("hello", "goodbye")
        # 這裡要稍微注意, 你會發現 a 並沒有變動, 而是回傳一個新的答案
        print("原本的沒改變:", a)
        print("回傳新的答案:", b)
        # 所以如果你要 a 後面換成新的可以這樣寫 a = a.replace
        # replace 如果帶入第三個參數的話可以指定只取代幾個
        a = a.replace("hello", "goodbye", 2)
        print("設定回去:", a)

原本的沒改變: hellohellohello
回傳新的答案: goodbyegoodbyegoodbye
設定回去: goodbyegoodbyehello
```

再來練習兩個大小寫的轉換:

1. upper: 轉成大寫
2. lower: 轉成小寫

```
[程式]: # 這裡要注意, 由於他已經知道自己的所有東西, 所有不需參數
        print("轉成大寫", "hello".upper())
        print("轉成小寫", "HELLO".lower())

轉成大寫 HELLO
轉成小寫 hello
```

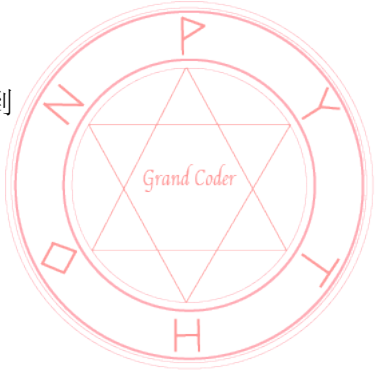
2.4 布林

有了文字，有了數字，我們已經可以表達大部分的資料了，但是如果今天我們要表示的『漂亮與否』的答案，文字和數字就不夠用了，因為這答案很明顯只有兩個!

一個叫『是』，一個叫『否』，在 Python 裡我們要寫『True』和『False』(沒錯！記得要第一個字大寫)

這就是我們的『是否資料』，也就是大家常說的『布林資料』

但是我們其實比較少直接寫 True 和 False，而是由別的運算得到



2.4.1 產生是否的運算

數字: **>**(大於), **<**(小於), **>=**(大於等於), **<=**(小於等於), **==**(相等, 因為一個 **=** 是給名字, 所以相等是 **==**)

文字: **==**(相等), in(檢查前面的字串是不是後面字串的一部份)

```
[程式]: # 直接使用 True False
        print("直接使用:", True)
        # 數字運算產生布林
        print("大於:", 5 > 3)
        print("相等:", 2 == 2)
        # 文字運算產生布林 (要記得大寫小寫是完全不同的)
        print("文字相等:", "hello" == "HELLO")
        print("文字包含:", "hel" in "hello")
```

```
直接使用: True
大於: True
相等: True
文字相等: False
文字包含: True
```

2.4.2 布林結合運算

有時候我們會把兩個布林運算結合在一起，其實就是我們中文說的『而且』和『或』，Python 寫成『and』和『or』

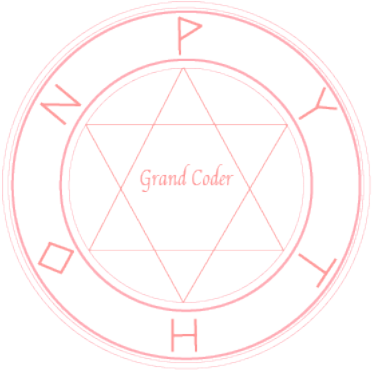
這裡其實非常好記，你只要想到一個例子：假設一個女生喜歡男生條件是『帥』『而且』『有才華』，看起來就是兩個都要有才行。但如果是『帥』『或者』『有才華』，看起來只要有一個可以就可以了!

我們詳細地列出表格

and	True	False
True	True	False
False	False	False

or	True	False
True	True	True
False	True	False

簡單的說，and 兩個都對才對，or 兩的都錯才錯



2.4.3 運算子順序

我們現在學過了很多種不同的運算，但是到底哪個運算先做呢? 這裡只告訴各位最重要的一句話!

千萬不要去背運算子順序!! 因為運算子順序其實是根據大部分人的直覺制定的

```
5 >= 3 + 2 and True
```

這裡我想你應該不會想先做 2 and True，你會覺得 and 兩邊應該要先做完，所以把 and 放後面點，而你也不會先做 5 >= 3，再把結果 + 2，所以實際上就是先算 3 + 2，再來 5 >= 5，再來 True and True

這裡就會很多人問，那不如就用 () 來確定誰先做，盡量也不要這樣做

```
((5 >= (3 + 2)) and True)
```

你覺得這樣寫和上面那樣寫那個比較漂亮呢?! 而且太多 () 會導致你不知道你少了幾個，也不知道下一個要加在哪裡

```
[程式]: print("and 結合:", True and False)
        print("or 結合:", True or False)
        # 請運用你對運算子的天然直覺
        print("複雜結合:", 5 >= 3 + 2 and True)
```

```
and 結合: False
or 結合: True
複雜結合: True
```

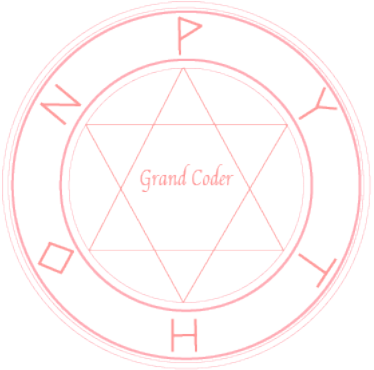
2.5 結論

現在你已經學會了程式語言基礎部分了!!!

我把他總結成一個讓你好記的口語

『三種基本資料』: 數字，文字，布林『二種基本操作』: 運算，技能『一個特殊符號』: =(給名字)

請好好記得這『三二一』!



3 if 語法

3.1 二選一

之前我們的程式都是從第一行執行到最後一行，但我們的程式理論上要有那種可以選擇的分支才對

其實就是我們平常中文說的:『如果』『否則』

中文是這樣說的

如果 XXX，我就做事情 1，否則我就做事情 2

換成 Python 語法

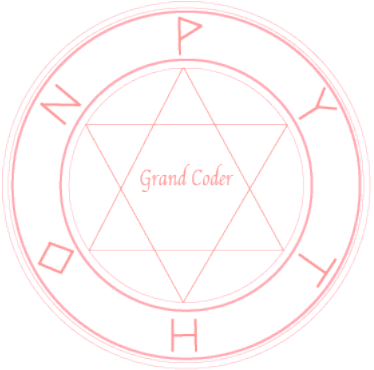
```
if 布林判斷:
    程式碼片段 1
else:
    程式碼片段 2
```

有幾個要注意的點

❤ if 和布林判斷間要加個空白鍵

❤ Python 當遇到這種有階層性架構，也就是『屬於』架構 (片段 1 屬於 if)，要加上縮排 (TAB) 和: 排版

```
[程式]: # 這裡我們學一個新的功能叫做 input, 可以讓使用者自行輸入
        # 但是這裡的輸入得到的答案一定是文字, 所以記得要轉換成你想要的
        # 我們用 float 轉換成數字的小數
        score = float(input("請輸入成績:"))
        if score > 60:
            print("PASS")
        else:
            print("FAIL")
```



```
請輸入成績:87.2
PASS
```

3.2 多選一

多選一的時候，你可以用『從上而下的單選題』來記

語法是這樣的

```
if 選項 A:
    A 對的話做完這段程式碼離開
elif 選項 B:
    B 對的話做完這段程式碼離開
elif 選項 C:
    C 對的話做完這段程式碼離開
else:
    上面都不對直接做完這段程式碼離開
```

3.2.1 注意的點

但是如果多選一是有包含關係的話，記得比較嚴苛的條件要放上面

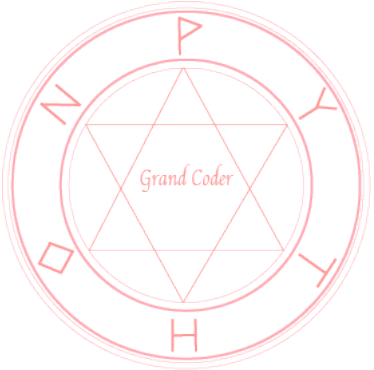
譬如下面三個選項

1. 成績 > 90
2. 成績 > 80
3. 成績 > 70

很明顯條件 1 是比較嚴苛的條件，就要放在最上面，做出第一次的過濾

```
[程式]: score = float(input("請輸入成績:"))
        if score > 90:
            print("RANK A")
        elif score > 80:
            print("RANK B")
        elif score > 70:
            print("RANK C")
        else:
            print("RANK D")
```

```
請輸入成績:78.2
RANK C
```



3.3 (實戰演練) 剪刀石頭布

剪刀石頭布是個看似簡單，但卻富有哲理的遊戲
如果你用最粗淺的想法來想，你會寫出 9 個 if-elif-else，因為你的 3 個狀況 x 對面的 3 個狀況
那我們就會想可不可以把輸，贏和平手真正的意義找出來

3.3.1 真正意義

其實剪刀石頭布是一個下一個拳一定贏前一個拳的遊戲
換句話說，如果換成選數字，就是下一個數字一定贏前一個數字的遊戲
❤ 重要概念: 當你發現其實東西的本質是有比大小，就像剪刀石頭布，請用數字來替代這些東西！因為會讓你比較大小很順利！

3.3.2 取餘數運算

這裡特別把取餘數運算拿出來特別說，因為取餘數運算其實是一個把值限制住的運算
a % b -> 小於 b 的數

3.3.3 (預先使用) 使用其他.py 的功能

當我們要使用其他的.py(就算是內建的)，我們一定要先引用她
你可以想像成你在使用別人文章的句子，你要先說我要引用 (import) 這本書的哪一句 (功能)
於是我們使用 import random 來引用內建的 random.py
再使用裡面的 randint 功能 (使用的時候要用.，我們之前說過的『的』)
語法

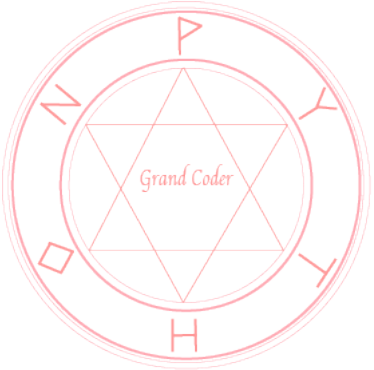
```
import (某 py 的名字)
(名字).(裡面你想用的功能)
```

3.3.4 (預先使用) 清單

因為我們希望可以把 0, 1, 2 翻譯成剪刀石頭布
所以我們準備了一個翻譯清單 0 -> 剪刀 1 -> 石頭 2 -> 布
Python 準備清單是使用 []，用座號做出轉換一樣是清單名字 [座號]

```
[程式]: # 引用內建的 random.py
        import random

        me = int(input("請出拳 [0] 剪刀 [1] 石頭 [2] 布"))
        com = random.randint(0, 2)
```



```
trans = ["剪刀", "石頭", "布"]
# 清單名字 [號碼牌]
print("你出的拳:", trans[me])
print("電腦的拳:", trans[com])

if me == com:
    print("平手")
# 針對 me= 剪刀, com= 布 的 case 處理
# +1 相當於往下一個拳前進, 但是不該有 3, 布 +1 應該是剪刀才對
# 於是我們對 3 取餘數, 讓他回到 0
elif me == (com + 1) % 3:
    print("我贏了")
else:
    print("我輸了")
```

```
請出拳 [0] 剪刀 [1] 石頭 [2] 布 0
你出的拳: 剪刀
電腦的拳: 布
我贏了
```

3.4 (進階練習) 棒打老虎雞吃蟲

棒打老虎雞吃蟲很明顯也是跟剪刀石頭布類似的遊戲

先不要看下面的答案，練習一下你是否也可以寫出類似的小遊戲呢?

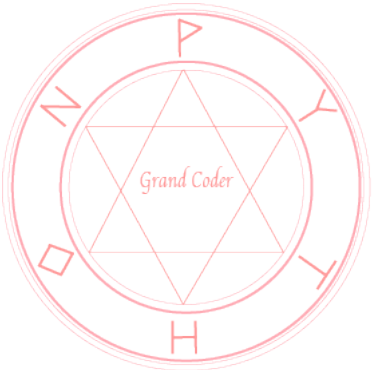
```
[程式]: import random

me = int(input("請出拳 [0] 蟲 [1] 雞 [2] 老虎 [3] 棒子"))
com = random.randint(0, 3)

trans = ["蟲", "雞", "老虎", "棒子"]

print("你出的拳:", trans[me])
print("電腦的拳:", trans[com])

# 跟剪刀石頭布不一樣的是平手條件反而最複雜
# 所以我們平手放在最後
if me == (com + 1) % 4:
    print("我贏了")
elif com == (me + 1) % 4:
    print("電腦贏了")
else:
    print("平手")
```



```
請出拳 [0] 蟲 [1] 雞 [2] 老虎 [3] 棒子 0
你出的拳: 蟲
電腦的拳: 雞
電腦贏了
```

3.5 結語

我們已經學會第一個重要的文法 if 了，還練習了一個剪刀石頭布遊戲

你應該慢慢發現寫程式其實只是把你用中文思考的邏輯用更清晰的方式轉換成為程式!

也發現，剪刀石頭布換個想法，竟然可以變得這麼簡單，這麼有擴展性，可以擴展到不管多少對多少的遊戲!