

# 程式語言

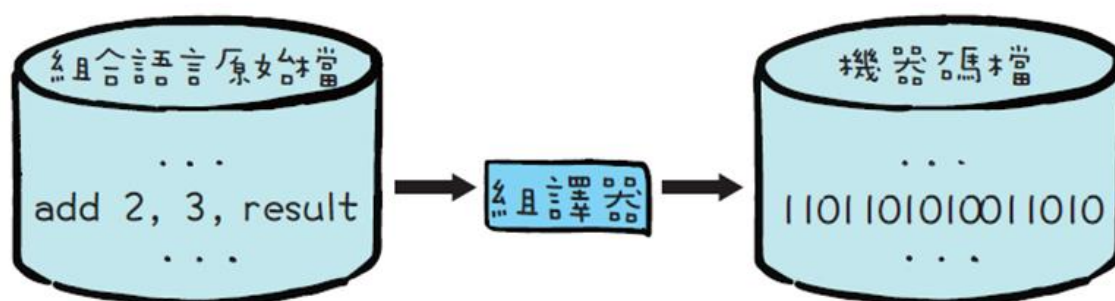
電腦程式又被稱做軟體，是一連串告知電腦所需執行任務指令（instruction）。

## 1. 機器語言

電腦的原始語言為機器語言（machine language），是一組內建的原始指令，會因為不同型態的電腦而有所不同。指令是以二進位編碼（binary code）的形式表示，因此，如要以原始語言下指令給電腦，就必須以二進位編碼的形式輸入指令。

## 2. 組合語言

組合語言（assembly language）是為了讓程式設計者更容易撰寫指令而開發的。由於電腦看不懂組合語言，因此需要組譯器（assembler）將以組合語言撰寫的程式翻譯成機器語言。



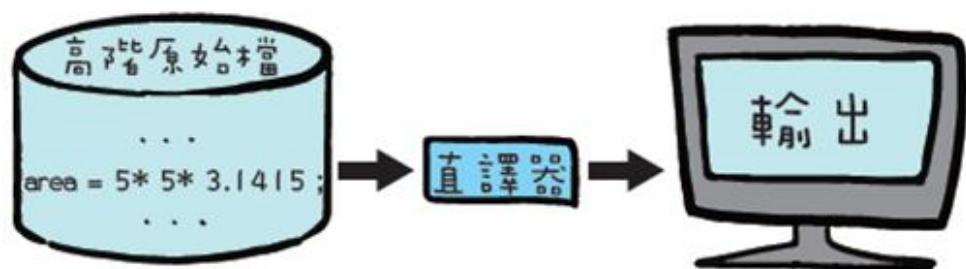
## 3. 高階語言

1950 年代，誕生了一種稱為高階語言（ high-level language ）的程式語言，跟英文很像，容易學習與使用。一般稱高階程式語言的指令為敘述（ statement ）。

使用高階語言撰寫的程式，被稱作原始程式（ source program ）或原始碼（ source code ）。由於電腦看不懂原始程式，因此需要藉由直譯器（ interpreter ）或編譯器（ compiler ）程式工具來翻譯成機器碼才能執行。

這兩款程式工具不是硬體（ hardware ）而是軟體（ software ）。

直譯器從原始碼讀取一個敘述，將其翻譯成機器碼或虛擬機器碼，接著馬上作執行的動作，程式語言有 BASIC 、 JavaScript 、 Python 、 PHP 、 LISP 、 Shell 、 Perl 、 Ruby ...等。



翻譯器會將整個原始碼翻譯成一份機器碼檔案，接著這份機器碼檔案會被執行，程式語言有 C、C++、C#、COBOL、FORTRAN、Java、Pascal、Visual BASIC ...等。



# Python 簡介

Python 是一通用、直譯，以及物件導向的程式語言。

Python 是由 Guido van Rossum 於 1990 年在荷蘭所創造的。由於它簡單、簡潔、直覺式的語法，以及龐大的函式庫，導致在工業和學術界廣泛地受到喜愛的程式語言。

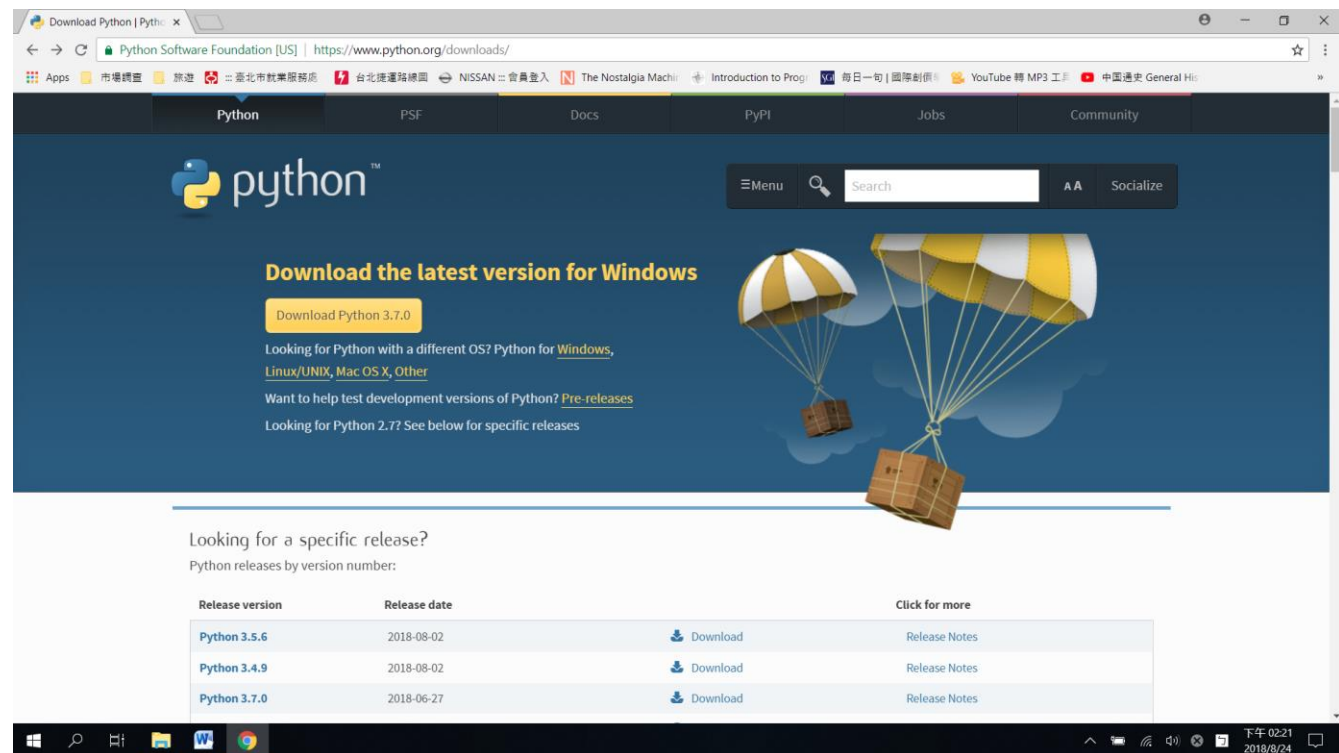
各種主要的作業系統都支援 Python。Python 程式常常不需要修改，便可以同時在 Windows、Linux、Mac OS X 平台上執行。

Python 應用範圍：系統程式設計、圖形處理、數學處理、文字處理、資料庫程式設計、網路程式設計、Web 程式設計、多媒體運用。

使用 Python 的企業：Google、NASA、YouTube、Minecraft、Dropbox、豆瓣網 ...等。

# Python 程式下載與安裝

前往 Python 官方網站 <https://python.org/downloads>



點選適合電腦 OS 的版本



## Python Releases for Windows

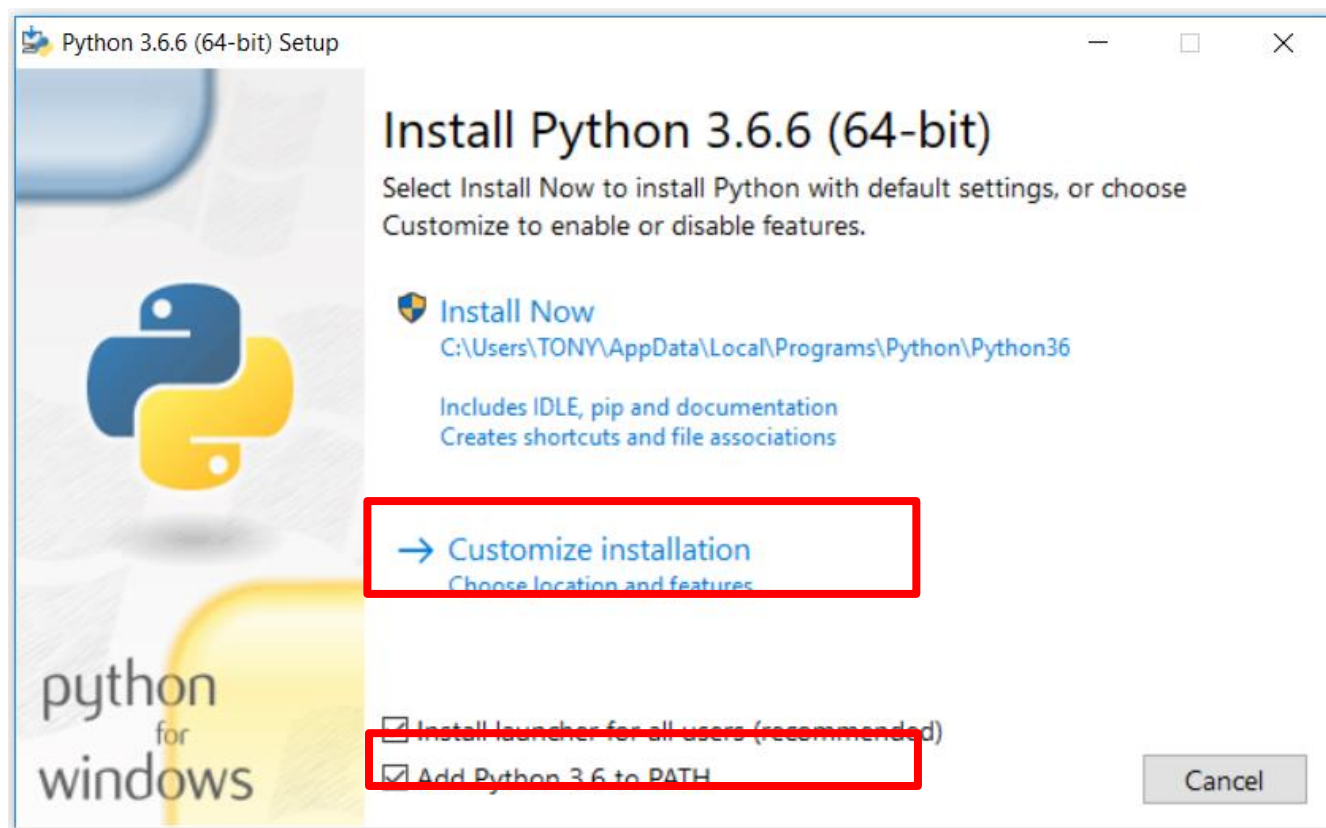
- [Python 3.6.6 - 2018-06-27](#)
  - Download [Windows x86 web-based installer](#)
  - Download [Windows x86 executable installer](#)
  - Download [Windows x86 embeddable zip file](#)
  - Download [Windows x86-64 web-based installer](#)
  - Download [Windows x86-64 executable installer](#)
  - Download [Windows x86-64 embeddable zip file](#)
  - Download [Windows help file](#)

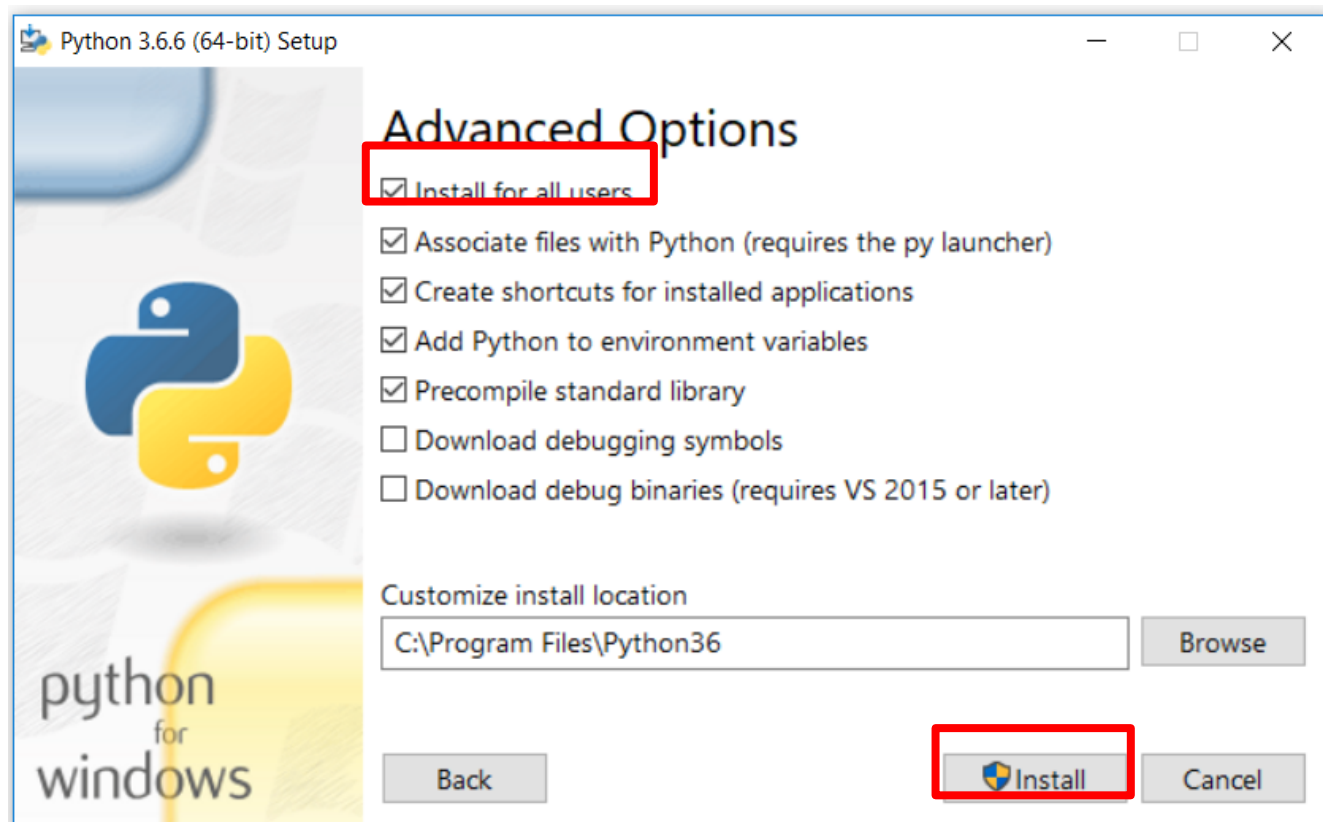
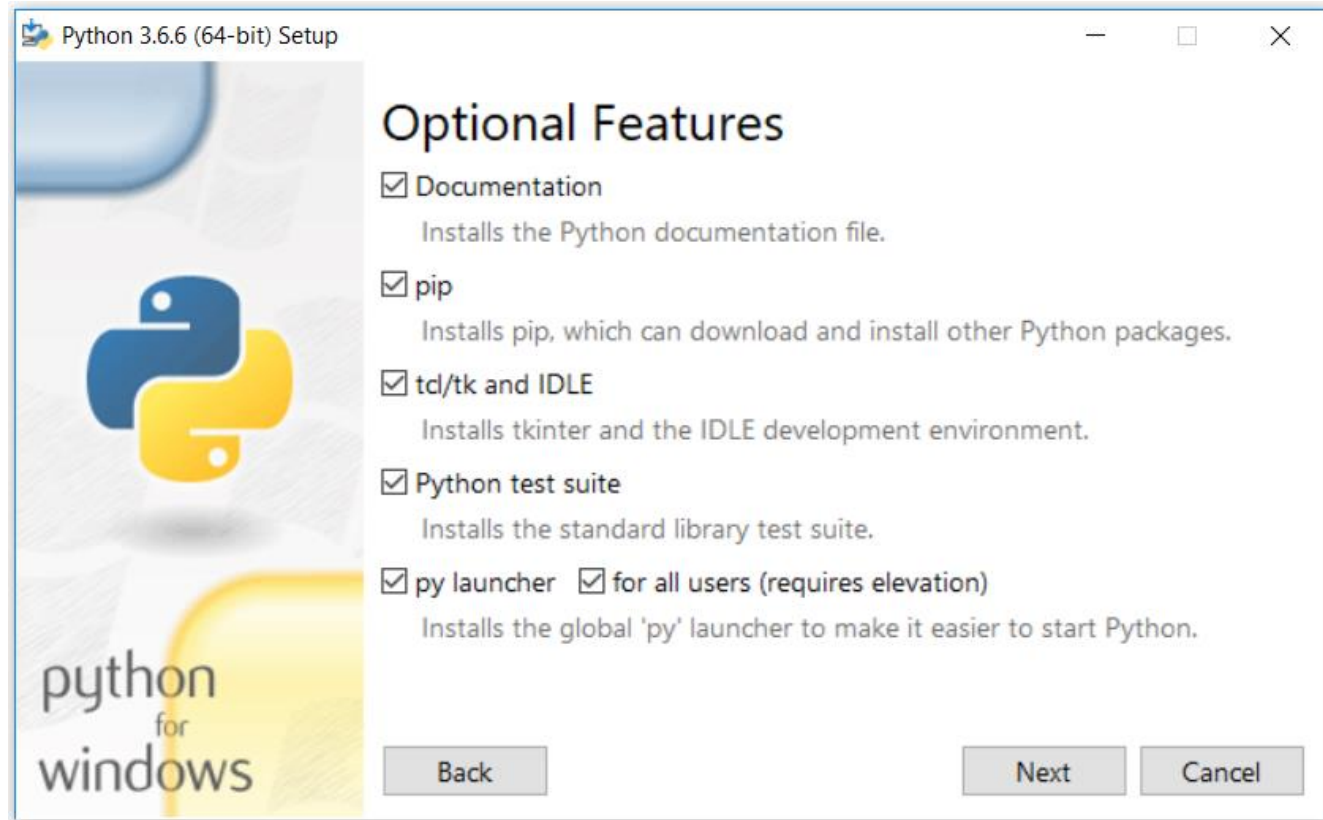
← 32 位元

← 64 位元

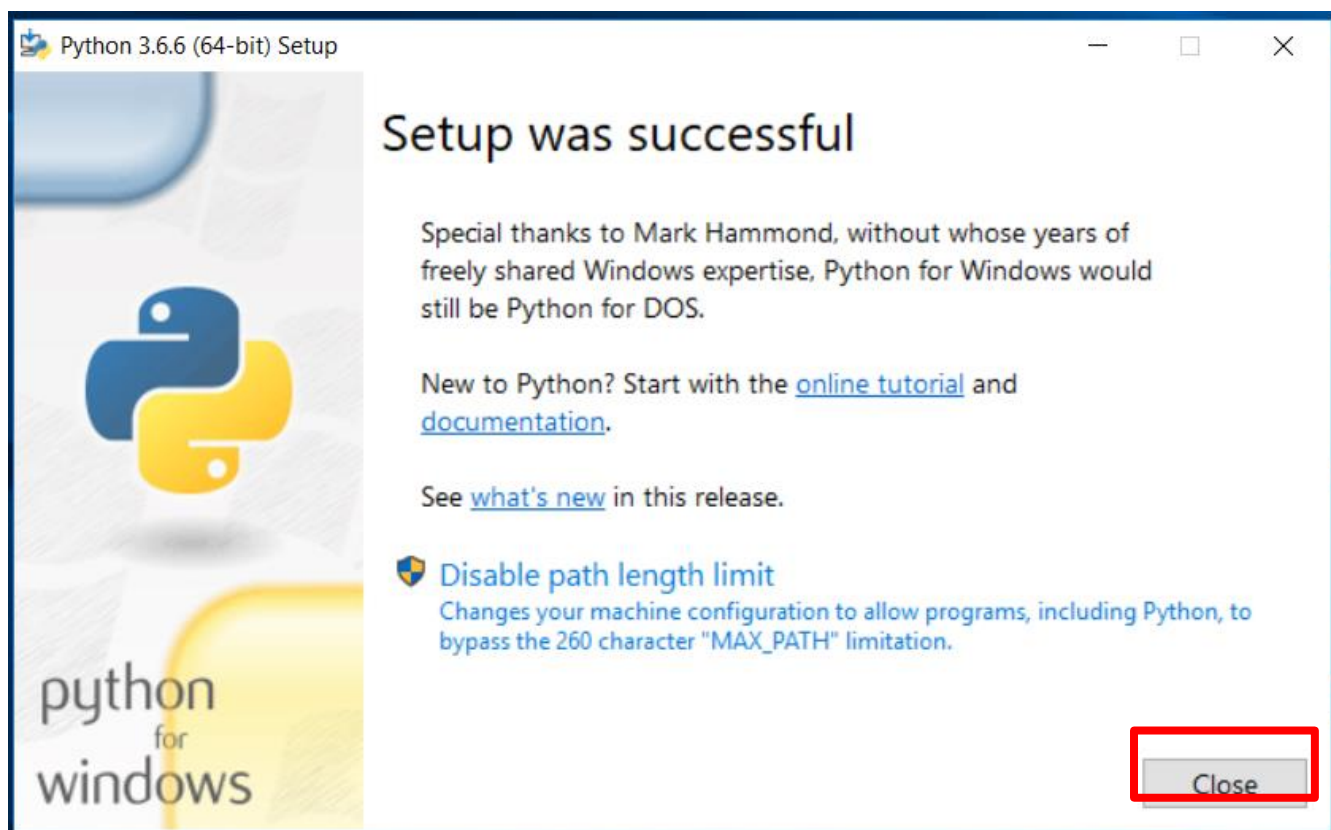
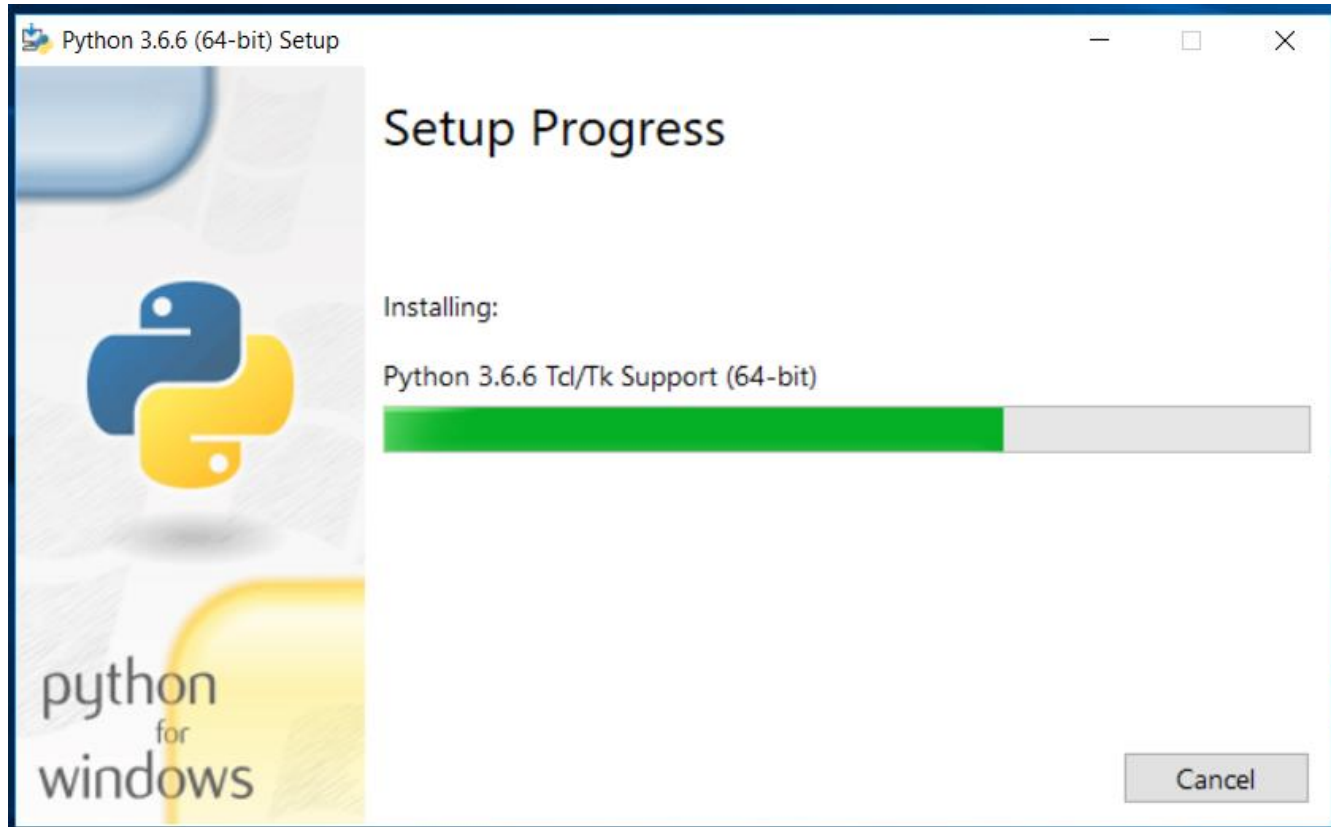
## Python Releases for Mac OS X

- [Latest Python 3 Release - Python 3.7.0](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.7.0 - 2018-06-27](#)
  - Download [macOS 64-bit installer](#)
  - Download [macOS 64-bit/32-bit installer](#)
- [Python 3.6.6 - 2018-06-27](#)
  - Download [macOS 64-bit installer](#)
  - Download [macOS 64-bit/32-bit installer](#)







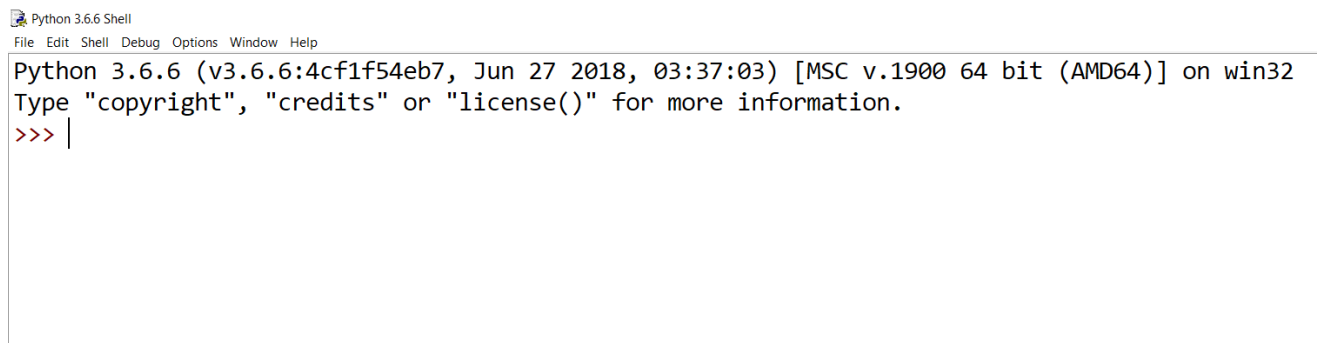


Mac 環境下 IDLE 不支持中文輸入的解決辦法：

1. IDLE 介面使用的 Tkinter 需要依賴 Tcl/Tk，而系統自帶的 Tcl/Tk 版本太低，造成不相容的問題，因此，下載安裝最新版的 Tcl/Tk，就能輸入中文了。
2. 按照 IDLE and tkinter with Tcl/Tk on MacOS 的說明，再依據系統環境選擇適合的 ActiveTcl 下載安裝。
3. 影片說明：<https://www.youtube.com/watch?v=jo9XAs6Vsuw>。

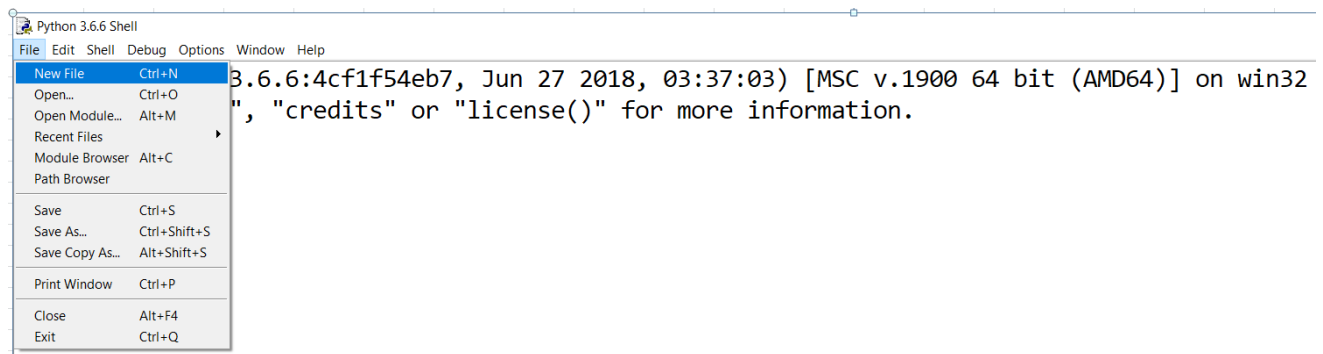
## 啟動 Python

交談式開發環境 ( Interactive Development Environment, IDLE ) 是一整合開發環境，可以來建立、打開、儲存、編輯，以及執行 Python 程式。

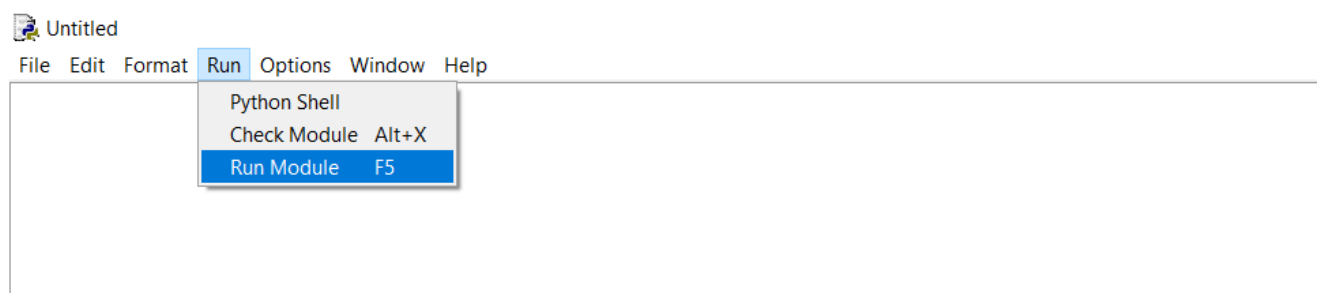


點擊「IDLE(Python 3.6 4-bit)」後，會看到 >>> 符號，>>> 是 Python 敘述的提示，此處可以鍵入敘述。

在 >>> 提示下輸入敘述是很方便，但無法儲存。為了能夠儲存以便日後使用，可以在 IDLE 選單選取 File → New File，用編輯器來建立原始碼（source code）檔案。



當程式碼建立好之後，選取選單 Run → Run Module（或按 F5）來執行原始碼。



執行程式的結果將會輸出於 IDLE 的視窗中。

```
#display two messages
print("Learning Python now!")
print("Python is fun")
```

上例程式的第一行為註解敘述，用以記錄此程式為何以及它的結構為何。註解敘述可幫助程式設計師彼此溝通與了解程式。因為它不是程

式敘述，所以直譯器不會理會註解敘述。Python 的註解敘述是在一行的最前面加上 # 符號，此稱為行註解敘述 ( line comment )。也可以使用'''與'''聯合撰寫多行的註解敘述。此稱為段落註解 ( paragraph comment )。

有關 Python 的縮排，每一敘述皆是從新行的第一欄位開始撰寫。若輸入以下的敘述，Python 將會產生錯誤的訊息。

```
#display two messages
    print("Learning Python now!")
print("Python is fun")
```

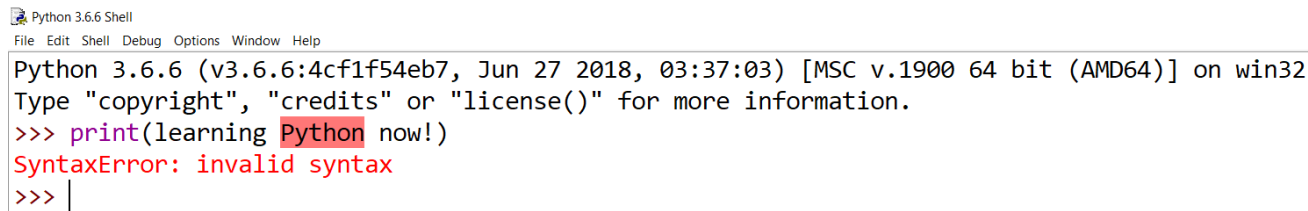
不要將任何標點符號，如分號，放在敘述的尾端。例如以下的程式碼，Python 也會產生錯誤的訊息。

```
#display two messages
print("Learning Python now!");
print("Python is fun");
```

Python 程式大、小寫字母是有差別的。若將程式中的 print 改為 Print 將會產生錯誤。

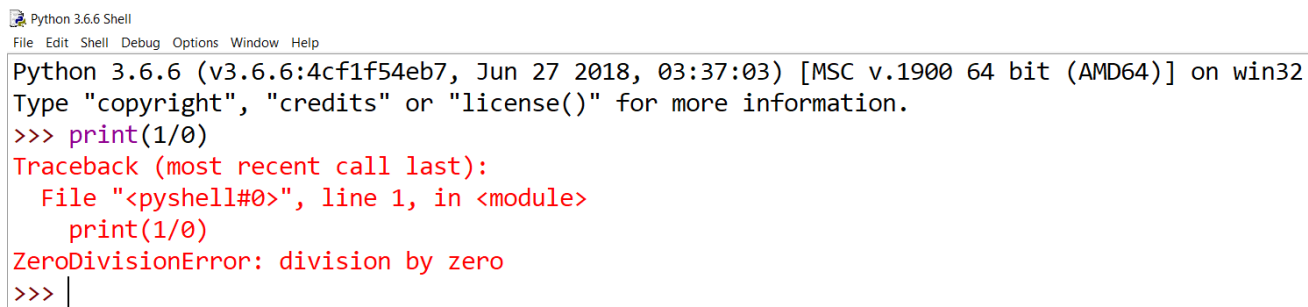
程式設計錯誤可分為三類：語法錯誤 ( syntax errors )、執行期間的錯誤 ( runtime errors )，以及邏輯錯誤 ( logic errors )。

1. 一般最常見的錯誤是語法錯誤。若違反 Python 程式規則，將產生錯誤訊息，例如忘了加雙引號或字拼錯了。



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(learning Python now!)
SyntaxError: invalid syntax
>>> |
```

2. 執行期間的錯誤 ( Runtime errors ) 是會導致程式不正常終止的錯誤。例如，在整數除法運算中，當除數為零的時候將會出現執行期的錯誤。

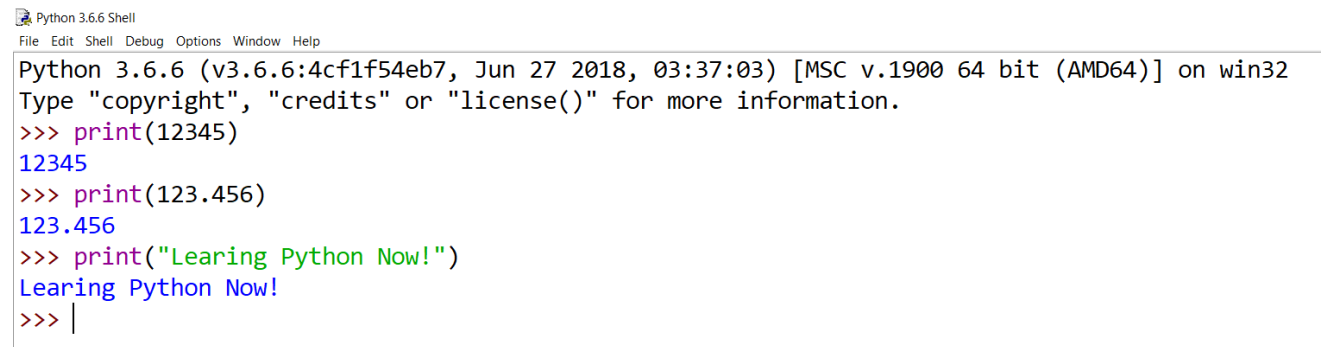


```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(1/0)
Traceback (most recent call last):
  File "<pysHELL#0>", line 1, in <module>
    print(1/0)
ZeroDivisionError: division by zero
>>> |
```

3. 邏輯錯誤 ( Logic errors ) 出現於程式執行的結果與預期中的不同。這種錯誤發生的原因有很多種。例如將華氏 80 度轉為攝氏：為了得到正確的答案，應該使用  $5 / 9 * (80 - 32)$ ，而不是  $5 / 9 * 80 - 32$  運算式。

## 輸出函式 print()

`print` 函式可印出數值、浮點數和字串輸出到螢幕，如以下敘述印出：

A screenshot of a Python 3.6.6 Shell window. The title bar says "Python 3.6.6 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following: "Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", followed by three lines of code and their output: ">>> print(12345)" outputs "12345", ">>> print(123.456)" outputs "123.456", and ">>> print('Learning Python Now!')" outputs "Learning Python Now!". The prompt ">>> |" is at the bottom.

```
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(12345)
12345
>>> print(123.456)
123.456
>>> print("Learning Python Now!")
Learning Python Now!
>>> |
```

為了執行某一特定的動作，須借助轉義序列（`escape sequence`）。

轉義序列	功能說明
<code>\n</code>	換行
<code>\t</code>	跳八格
<code>\\</code>	輸出反斜線
<code>\"</code>	輸出雙引號
<code>\'</code>	輸出單引號

輸出範例：

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Learning Python now!\nPython is fun.")
Learning Python now!
Python is fun.
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("|\\tHello, world|")
|      Hello, world|
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("\\n is skip a line")
\\n is skip a line
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("\\\"Hello, everyone\\\"")
"Hello, everyone"
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Let\\'s go")
Let's go
```

以上的 `print` 函式皆只印出一個項目而已，它也可以印出多個項目時，

此時可利用逗號將其分開即可。如下敘述：

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print('Radius =', 12)
Radius = 12
>>>
```

若要將輸出結果看起來更美觀的話，則需借助格式化的輸出。格式化

的輸出有三種格式：

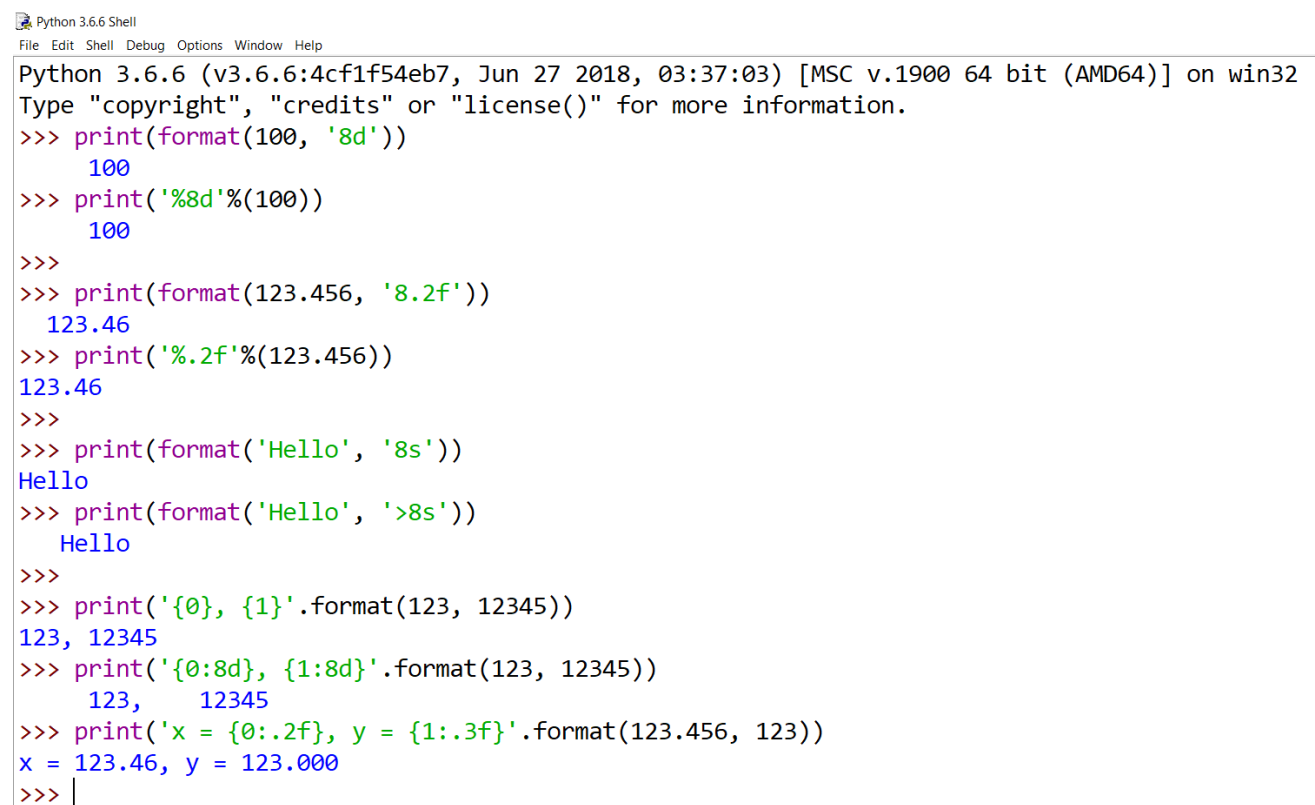
```
print(format(item, format-specifier))
```

```
print('%format-specifier'%(item))
```

```
print('{0: format-specifier}, {1: format-specifier}'.format(item0, item1))
```

當整數時，則使用 'd' 指定器；若為浮點數，則使用 'f' 指定器；若

為字串，則使用 's' 指定器。



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(100, '8d'))
      100
>>> print('%8d'%(100))
      100
>>>
>>> print(format(123.456, '8.2f'))
    123.46
>>> print('%8.2f'%(123.456))
    123.46
>>>
>>> print(format('Hello', '8s'))
Hello
>>> print(format('Hello', '>8s'))
      Hello
>>>
>>> print('{0}, {1}'.format(123, 12345))
123, 12345
>>> print('{0:8d}, {1:8d}'.format(123, 12345))
      123,      12345
>>> print('x = {0:.2f}, y = {1:.3f}'.format(123.456, 123))
x = 123.46, y = 123.000
>>> |
```

而每一指定器前可加入數值，表示其欄位寬，如 '8d'表示有 8 個欄位

空間。還有向左或向右靠齊，分別以 < 和 > 來加以控制，它可以用

於整數、浮點數，以及字串皆可。



格式指定器	說明
"10.2f"	以欄位寬 10，將浮點數格式化到小數點後 2 位。
"8d"	以欄位寬 8，將整數格式化。
"30s"	以欄位寬 30，將字串格式化。
"<10.2f"	將浮點數格式化的項目向左靠齊。
">10.2f"	將浮點數格式化的項目向右靠齊。

## 多個項目

```
print('x = {0:5d}, y = {1:7d}'.format(123, 12345))
print('x = {p:5d}, y = {q:7d}'.format(p=123, q=12345))
```

### 【練習題目】

1. 試撰寫一程式，將下列兩行字串向右靠齊：

```
Python is fun  
Let's learn
```

2. 試撰寫一程式，將下列整數排列整齊：

```
12345 12 1234567  
12 1234567 123  
1234567 123 12345
```

3. 改錯題 ( Debugs )

```
print("%10d %9d %9d"$(12345, 12, 1234567))  
print("Radius:{a:3d}, Area:{b:10.2f}".format(5, 5*3.14159))
```