

---

# Installing Python Modules

發 F 3.11.8

Guido van Rossum and the Python development team

4 月 02, 2024

Python Software Foundation  
Email: [docs@python.org](mailto:docs@python.org)



<b>1</b>	<b>關鍵術語</b>	<b>3</b>
<b>2</b>	<b>基本用法</b>	<b>5</b>
<b>3</b>	<b>我該如何... ?</b>	<b>7</b>
3.1	... 在 Python 3.4 之前的 Python 版本中安裝 pip?	7
3.2	... 只☞目前的使用者安裝套件?	7
3.3	... 安裝科學的 Python 套件?	7
3.4	... 平行安裝多個 Python 版本☞使用它們?	8
<b>4</b>	<b>常見的安裝問題</b>	<b>9</b>
4.1	在 Linux 上安裝套件至系統 Python	9
4.2	未安裝 pip	9
4.3	安裝二進制擴充 (binary extension)	9
<b>A</b>	<b>術語表</b>	<b>11</b>
<b>B</b>	<b>關於這些☞明文件</b>	<b>27</b>
B.1	Python 文件的貢獻者們	27
<b>C</b>	<b>沿革與授權</b>	<b>29</b>
C.1	軟體沿革	29
C.2	關於存取或以其他方式使用 Python 的合約條款	30
C.2.1	用於 PYTHON 3.11.8 的 PSF 授權合約	30
C.2.2	用於 PYTHON 2.0 的 BEOPEN.COM 授權合約	31
C.2.3	用於 PYTHON 1.6.1 的 CNRI 授權合約	32
C.2.4	用於 PYTHON 0.9.0 至 1.2 的 CWI 授權合約	33
C.2.5	用於 PYTHON 3.11.8 ☞明文件☞程式碼的 ZERO-CLAUSE BSD 授權	33
C.3	被收☞軟體的授權與致謝	34
C.3.1	Mersenne Twister	34
C.3.2	Sockets	35
C.3.3	非同步 socket 服務	35
C.3.4	Cookie 管理	36
C.3.5	執行追☞	36
C.3.6	UUencode 與 UUdecode 函式	37
C.3.7	XML 遠端程序呼叫	37
C.3.8	test_epoll	38

C.3.9	Select kqueue . . . . .	38
C.3.10	SipHash24 . . . . .	39
C.3.11	strtod 與 dtoa . . . . .	39
C.3.12	OpenSSL . . . . .	40
C.3.13	expat . . . . .	43
C.3.14	libffi . . . . .	44
C.3.15	zlib . . . . .	44
C.3.16	cfuhash . . . . .	45
C.3.17	libmpdec . . . . .	45
C.3.18	W3C C14N 測試套件 . . . . .	46
C.3.19	Audioop . . . . .	47
C.3.20	asyncio . . . . .	47
<b>D</b>	<b>版權宣告</b>	<b>49</b>
	<b>索引</b>	<b>51</b>

**電子郵件**[distutils-sig@python.org](mailto:distutils-sig@python.org)

作一個普及的開源開發專案，Python 有一個活躍的支持社群，由其貢獻者及使用者組成，而他們也讓他們的軟體可被其他 Python 開發者在開源授權條款下使用。

這樣可以讓 Python 使用者們有效地共享和合作，受益於其他人對常見（有時甚至是罕見）的問題已經建立的解方案，更可以在公用社群中在地貢獻他們自己的解方案。

This guide covers the installation part of the process. For a guide to creating and sharing your own Python projects, refer to the [Python packaging user guide](#).

---

**備註：**對於企業和其他機構的使用者，要注意到，許多組織對於使用和貢獻開源軟體都有自己的政策。在開始使用配備 Python 的發布及安裝工具時，請將那些政策納入考量。

---



---

## 關鍵術語

---

- `pip` 是首選的安裝程式。從 Python 3.4 開始，它被預設包含在 Python 二進制安裝程式中。
- *virtual environment* (虛擬環境) 是一種半隔離的 Python 環境，可以為某個特定應用程式安裝其所需的套件，而不用在整個系統上安裝它們。
- `venv` 是建立虛擬環境的標準工具，它從 Python 3.3 開始成為 Python 的一部分。從 Python 3.4 開始，它會預設地安裝 `pip` 到所有被建立的虛擬環境。
- `virtualenv` 是 `venv` 的一個第三方替代方案（及其前身）。它使虛擬環境可以在 Python 3.4 之前的版本被使用，那些版本要不是根本不提供 `venv`，就是無法自動安裝 `pip` 到所建立的環境中。
- **Python 套件索引 (Python Package Index)** 是開源授權套件的一個公共儲存庫，其中的套件皆可被其他 Python 使用者所使用。
- **Python 封裝管理站 (Python Packaging Authority)** 是一個由開發者和明文件作者組成的團隊，負責維護及改進標準封裝工具，以及相關的元資料 (metadata) 和檔案格式標準。他們在 [GitHub](#) 和 [Bitbucket](#) 這兩個平台上維護各種工具、明文件及問題追蹤系統。
- `distutils` 是最早的建置和發布系統，於 1998 年首次被加入 Python 標準函式庫。雖然直接使用 `distutils` 的方式已經被逐步淘汰，它仍然是現今封裝和發布的基礎結構根基，而且它不僅仍然是標準函式庫的一部分，它的名稱也以其他的方式存活著（例如：用於協調 Python 封裝標準開發的郵寄清單就是以它命名）。

在 3.5 版的變更：對於建立虛擬環境，現在推薦使用 `venv`。

### 也參考：

[Python 封裝使用者指南：建立和使用虛擬環境](#)





### 基本用法

標準封裝工具皆是以能從命令列使用的方式被設計的。

以下指令將從 Python 套件索引安裝一個模組的最新版本及其依賴套件 (dependencies):

```
python -m pip install SomePackage
```

**備註：**對於 POSIX 使用者（包括 macOS 和 Linux 使用者），本指南中的範例皆假設有使用 *virtual environment*。對於 Windows 使用者，本指南中的範例皆假設在安裝 Python 時，「可調整系統 PATH 環境變數」的選項已被選取。

在命令列中直接指定一個明確的或最小的版本也是可行的。當使用像是 >、< 的比較運算子，或某些可被 shell 所解釋的其他特殊字元時，套件名稱與版本編號應該要放在雙引號：

```
python -m pip install SomePackage==1.0.4    # specific version
python -m pip install "SomePackage>=1.0.4"  # minimum version
```

通常，如果一個合適的模組已被安裝，嘗試再次安裝它將不會有任何效果。要升級現有的模組就必須明確地請求：

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

關於 pip 及其能力的更多資訊和資源，可以在 [Python 封裝使用者指南](#) 中找到。

擬環境的建立是使用 venv 模組來完成。要在一個已用的擬環境中安裝套件，可使用前面展示的指令。

#### 也參考：

[Python 封裝使用者指南：安裝 Python 發布套件](#)



---

### 我該如何... ?

---

接下來是關於一些常見任務的快速解答或連結。

### 3.1 ... 在 Python 3.4 之前的 Python 版本中安裝 pip ?

Python 是從 Python 3.4 才開始綁定 pip 的。對於更早的版本，pip 需要被「自助安裝 (bootstrapped)」，請參考 Python 封裝使用者指南中的說明。

**也參考：**

Python 封裝使用者指南：安裝套件的需求

### 3.2 ... 只為目前的使用者安裝套件 ?

把 `--user` 選項傳給 `python -m pip install`，這樣將會只為目前使用者而非系統的所有使用者安裝套件。

### 3.3 ... 安裝科學的 Python 套件 ?

許多科學類 Python 套件都有複雜的二進制依賴套件，且目前不太容易直接使用 pip 安裝。目前為止，使用其他方法而非嘗試用 pip 來安裝它們，對使用者來說通常會更簡單。

**也參考：**

Python 封裝使用者指南：安裝科學套件

### 3.4 ... 平行安裝多個 Python 版本 使用它們？

在 Linux、macOS 以及其他 POSIX 系統中，使用帶有版本編號的 Python 指令 結合 `-m` 開關參數 (switch)，來運行 pip 的適當副本：

```
python2 -m pip install SomePackage # default Python 2
python2.7 -m pip install SomePackage # specifically Python 2.7
python3 -m pip install SomePackage # default Python 3
python3.4 -m pip install SomePackage # specifically Python 3.4
```

使用帶有合適版本編號的 pip 指令，也是可行的。

在 Windows 中，使用 Python 動指令 `py` 結合 `-m` 開關參數 (switch)：

```
py -2 -m pip install SomePackage # default Python 2
py -2.7 -m pip install SomePackage # specifically Python 2.7
py -3 -m pip install SomePackage # default Python 3
py -3.4 -m pip install SomePackage # specifically Python 3.4
```

### 4.1 在 Linux 上安裝套件至系統 Python

在 Linux 系統，Python 的某個安裝版本通常會被包含在 Linux 的發行版中。要安裝套件到這個 Python 版本上需要系統的 root 權限，且可能會干擾到系統套件管理器的運作。如果其他系統組件非預期地以 pip 被升級，也會干擾這些組件的運作。

在這樣的系統上，以 pip 安裝套件時，通常較好的方式是使用虛擬環境，或以個使用者安裝。

### 4.2 未安裝 pip

pip 有預設被安裝也是有可能的。一個在的解法是：

```
python -m ensurepip --default-pip
```

這還有其他關於安裝 pip 的資源。

### 4.3 安裝二進制擴充 (binary extension)

Python 基本上相當倚賴以原始碼為基礎的發布方式，也會期望使用者在安裝過程的某個階段，從原始碼來編譯擴充模組。

隨著引入對二進制 wheel 格式的支援，以及透過 Python 套件索引能至少在 Windows 和 macOS 發布 wheel 檔案，這個問題預期將會逐漸消失，因使用者將能更頻繁地安裝預建置 (pre-built) 的擴充，而不再需要自己建置它們。

有一些解方案，可用來安裝那些還無法以預建置的 wheel 檔案被使用的科學軟體，這些方案也有助於取得其他的二進制擴充，且無需在本機對它們進行建置。

也參考:

Python 封裝使用者指南: 二進制擴充

## 術語表

&gt;&gt;&gt;

互動式 shell 的預設 Python 提示字元。常見於能在直譯器中以互動方式被執行的程式碼範例。

...

可以表示：

- 在一個被縮排的程式碼區塊、在一對匹配的左右定界符（delimiter，例如括號、方括號、花括號或三引號）[\[F\]](#)部，或是在指定一個裝飾器（decorator）之後，要輸入程式碼時，互動式 shell 顯示的預設 Python 提示字元。
- [\[F\]](#)建常數 Ellipsis。

**2to3**

一個試著將 Python 2.x 程式碼轉[\[F\]](#)[\[F\]](#) Python 3.x 程式碼的工具，它是透過處理大部分的不相容性來達成此目的，而這些不相容性能[\[F\]](#)透過剖析原始碼和遍歷剖析樹而被檢測出來。

2to3 在標準函式庫中以 lib2to3 被使用；它提供了一個獨立的入口點，在 Tools/scripts/2to3。請參[\[F\]](#) 2to3-reference。

**abstract base class（抽象基底類[\[F\]](#)）**

抽象基底類[\[F\]](#)（又稱[\[F\]](#) ABC）提供了一種定義介面的方法，作[\[F\]](#)[\[F\]](#)duck-typing（鴨子型[\[F\]](#)）的補充。其他類似的技術，像是 hasattr()，則顯得笨拙或是帶有細微的錯誤（例如使用魔術方法（magic method））。ABC [\[F\]](#)用[\[F\]](#)擬的 subclass（子類[\[F\]](#)），它們[\[F\]](#)不繼承自另一個 class（類[\[F\]](#)），但仍可被 isinstance() 及 issubclass() 辨識；請參[\[F\]](#) abc 模組的[\[F\]](#)明文件。Python 有許多[\[F\]](#)建的 ABC，用於資料結構（在 collections.abc 模組）、數字（在 numbers 模組）、串流（在 io 模組）及 import 尋檢器和載入器（在 importlib.abc 模組）。你可以使用 abc 模組建立自己的 ABC。

**annotation（[\[F\]](#)釋）**

一個與變數、class 屬性、函式的參數或回傳值相關聯的標[\[F\]](#)。照慣例，它被用來作[\[F\]](#)type hint（型[\[F\]](#)提示）。

在執行環境（runtime），區域變數的[\[F\]](#)釋無法被存取，但全域變數、class 屬性和函式的[\[F\]](#)解，會分[\[F\]](#)被儲存在模組、class 和函式的 \_\_annotations\_\_ 特殊屬性中。

請參[\[F\]](#)variable annotation、function annotation、PEP 484 和 PEP 526，這些章節皆有此功能的[\[F\]](#)明。關於[\[F\]](#)釋的最佳實踐方法也請參[\[F\]](#) annotations-howto。

**argument (引數)**

呼叫函式時被傳遞給 *function* (或 *method*) 的值。引數有兩種：

- **關鍵字引數 (keyword argument)**: 在函式呼叫中，以識字 (identifier, 例如 `name=`) 開頭的引數，或是以 `**` 後面 *dictionary* (字典) 的值被傳遞的引數。例如，3 和 5 都是以下 `complex()` 呼叫中的關鍵字引數：

```
complex(real=3, imag=5)
complex(**{'real': 3, 'imag': 5})
```

- **位置引數 (positional argument)**: 不是關鍵字引數的引數。位置引數可在一個引數列表的起始處出現，和 (或) 作 `*` 之後的 *iterable* (可代物件) 中的元素被傳遞。例如，3 和 5 都是以下呼叫中的位置引數：

```
complex(3, 5)
complex(*(3, 5))
```

引數會被指定給函式主體中的附名區域變數。關於支配這個指定過程的規則，請參 [calls](#) 章節。在語法上，任何運算式都可以被用來表示一個引數；其評估值會被指定給區域變數。

另請參 [術語表](#) 的 *parameter* (參數) 條目、常見問題中的引數和參數之間的差別，以及 [PEP 362](#)。

**asynchronous context manager (非同步情境管理器)**

一個可以控制 `async with` 陳述式中所見環境的物件，而它是透過定義 `__aenter__()` 和 `__aexit__()` *method* (方法) 來控制的。由 [PEP 492](#) 引入。

**asynchronous generator (非同步生成器)**

一個會回傳 *asynchronous generator iterator* (非同步生成器代器) 的函式。它看起來像一個以 `async def` 定義的協程函式 (coroutine function)，但不同的是它包含了 `yield` 運算式，能生成一系列可用於 `async for` 圈的值。

這個術語通常用來表示一個非同步生成器函式，但在某些情境中，也可能是表示非同步生成器代器 (*asynchronous generator iterator*)。萬一想表達的意思不清楚，那就使用完整的術語，以避免歧義。

一個非同步生成器函式可能包含 `await` 運算式，以及 `async for` 和 `async with` 陳述式。

**asynchronous generator iterator (非同步生成器代器)**

一個由 *asynchronous generator* (非同步生成器) 函式所建立的物件。

這是一個 *asynchronous iterator* (非同步代器)，當它以 `__anext__()` *method* 被呼叫時，會回傳一個可等待物件 (awaitable object)，該物件將執行非同步生成器的函式主體，直到遇到下一個 `yield` 運算式。

每個 `yield` 會暫停處理程序，記住位置執行狀態 (包括區域變數及擱置中的 `try` 陳述式)。當非同步生成器代器以另一個被 `__anext__()` 回傳的可等待物件有效地回復時，它會從停止的地方繼續執行。請參 [PEP 492](#) 和 [PEP 525](#)。

**asynchronous iterable (非同步可代物件)**

一個物件，它可以在 `async for` 陳述式中被使用。必須從它的 `__aiter__()` *method* 回傳一個 *asynchronous iterator* (非同步代器)。由 [PEP 492](#) 引入。

**asynchronous iterator (非同步代器)**

一個實作 `__aiter__()` 和 `__anext__()` *method* 的物件。`__anext__()` 必須回傳一個 *awaitable* (可等待物件)。`async for` 會解析非同步代器的 `__anext__()` *method* 所回傳的可等待物件，直到它引發 `StopAsyncIteration` 例外。由 [PEP 492](#) 引入。

**attribute (屬性)**

一個與某物件相關聯的值，該值大多能透過使用點分隔運算式 (dotted expression) 的名稱被參照。例如，如果物件 `o` 有一個屬性 `a`，則該屬性能以 `o.a` 被參照。



如果一個物件允許，給予該物件一個名稱不是由 `identifiers` 所定義之識符 (identifier) 的屬性是有可能的，例如使用 `setattr()`。像這樣的屬性將無法使用點分隔運算式來存取，而是需要使用 `getattr()` 來取得它。

### awaitable (可等待物件)

一個可以在 `await` 運算式中被使用的物件。它可以是一個 *coroutine* (協程)，或是一個有 `__await__()` method 的物件。另請參 [PEP 492](#)。

### BDFL

Benevolent Dictator For Life (終身仁慈獨裁者)，又名 [Guido van Rossum](#)，Python 的創造者。

### binary file (二進制檔案)

A *file object* able to read and write *bytes-like objects*. Examples of binary files are files opened in binary mode ('rb', 'wb' or 'rb+'), `sys.stdin.buffer`, `sys.stdout.buffer`, and instances of `io.BytesIO` and `gzip.GzipFile`.

另請參 [text file](#) (文字檔案)，它是一個能讀取和寫入 `str` 物件的檔案物件。

### borrowed reference (借用參照)

In Python's C API, a borrowed reference is a reference to an object, where the code using the object does not own the reference. It becomes a dangling pointer if the object is destroyed. For example, a garbage collection can remove the last *strong reference* to the object and so destroy it.

對 *borrowed reference* 呼叫 `Py_INCREF()` 以將它原地 (in-place) 轉為 *strong reference* 是被建議的做法，除非該物件不能在最後一次使用借用參照之前被銷毀。`Py_NewRef()` 函式可用於建立一個新的 *strong reference*。

### bytes-like object (類位元組串物件)

一個支援 `bufferobjects` 且能匯出 *C-contiguous* 緩衝區的物件。這包括所有的 `bytes`、`bytearray` 和 `array.array` 物件，以及許多常見的 `memoryview` 物件。類位元組串物件可用於處理二進制資料的各種運算；這些運算包括壓縮、儲存至二進制檔案和透過 `socket` (插座) 發送。

有些運算需要二進制資料是可變的。明文文件通常會將這些物件稱為「可讀寫的類位元組串物件」。可變緩衝區的物件包括 `bytearray`，以及 `bytearray` 的 `memoryview`。其他的運算需要讓二進制資料被儲存在不可變物件 (「唯讀的類位元組串物件」) 中；這些物件包括 `bytes`，以及 `bytes` 物件的 `memoryview`。

### bytecode (位元組碼)

Python 的原始碼會被編譯成位元組碼，它是 Python 程式在 CPython 直譯器中的內部表示法。該位元組碼也會被暫存在 `.pyc` 檔案中，以便第二次執行同一個檔案時能更快速 (可以不用從原始碼重新編譯位元組碼)。這種「中間語言 (intermediate language)」據說是運行在一個 *virtual machine* (虛擬機器) 上，該虛擬機器會執行與每個位元組碼對應的機器碼 (machine code)。要注意的是，位元組碼理論上是無法在不同的 Python 虛擬機器之間運作的，也不能在不同版本的 Python 之間保持穩定。

位元組碼的指令列表可以在 `dis` 模組的明文文件中找到。

### callable (可呼叫物件)

一個 callable 是可以被呼叫的物件，呼叫時可能以下列形式帶有一組引數 (請見 [argument](#)):

```
callable(argument1, argument2, argumentN)
```

一個 *function* 與其延伸的 *method* 都是 callable。一個有實作 `__call__()` 方法的 `class` 之實例也是個 callable。

### callback (回呼)

作引數被傳遞的一個副程式 (subroutine) 函式，會在未來的某個時間點被執行。

### class (類)

一個用於建立使用者定義物件的模板。Class 的定義通常會包含 `method` 的定義，這些 `method` 可以在 `class` 的實例上進行操作。

**class variable (類變數)**

一個在 class 中被定義，且應該只能在 class 層次（意即不是在 class 的實例中）被修改的變數。

**complex number (複數)**

一個我們熟悉的實數系統的擴充，在此所有數字都會被表示成一個實部和一個虛部之和。複數就是實數單位（-1 的平方根）的實數倍，此單位通常在數學中被寫成  $i$ ，在工程學中被寫成  $j$ 。Python 建立了對複數的支援，它是用後者的記法來表示複數；虛部會帶著一個後綴的  $j$  被編寫，例如  $3+1j$ 。若要将 `math` 模組的工具等效地用於複數，請使用 `cmath` 模組。複數的使用是一個相當進階的數學功能。如果你有察覺到對它們的需求，那幾乎能確定你可以安全地忽略它們。

**context manager (情境管理器)**

An object which controls the environment seen in a `with` statement by defining `__enter__()` and `__exit__()` methods. See [PEP 343](#).

**context variable (情境變數)**

一個變數，其值可以根據上下文的情境而有所不同。這類似執行緒區域儲存區 (Thread-Local Storage)，在其中，一個變數在每個執行緒可能具有不同的值。然而，關於情境變數，在一個執行緒中可能會有多个情境，而情境變數的主要用途，是在行的非同步任務 (concurrent asynchronous task) 中，對於變數狀態的追蹤。請參閱 `contextvars`。

**contiguous (連續的)**

如果一個緩衝區是 *C-contiguous* 或是 *Fortran contiguous*，則它會確切地被視作是連續的。零維 (zero-dimensional) 的緩衝區都是 C 及 Fortran contiguous。在一維 (one-dimensional) 陣列中，各項目必須在記憶體中彼此相鄰地排列，而其索引順序是從零開始遞增。在多維的 (multidimensional) C-contiguous 陣列中，按記憶體位址的順序訪問各個項目時，最後一個索引的變化最快。然而，在 Fortran contiguous 陣列中，第一個索引的變化最快。

**coroutine (協程)**

協程是副程式 (subroutine) 的一種更廣義的形式。副程式是在某個時間點被進入並在另一個時間點被退出。協程可以在許多不同的時間點被進入、退出和回復。它們能以 `async def` 陳述式被實作。另請參閱 [PEP 492](#)。

**coroutine function (協程函式)**

一個回傳 *coroutine* (協程) 物件的函式。一個協程函式能以 `async def` 陳述式被定義，它可能會包含 `await`、`async for` 和 `async with` 關鍵字。這些關鍵字由 [PEP 492](#) 引入。

**CPython**

Python 程式語言的標準實作 (canonical implementation)，被發布在 [python.org](https://python.org) 上。「CPython」這個術語在必要時被使用，以區分此實作與其它語言的實作，例如 Jython 或 IronPython。

**decorator (裝飾器)**

一個函式，它會回傳另一個函式，通常它會使用 `@wrapper` 語法，被應用一種函式的變換 (function transformation)。裝飾器的常見範例是 `classmethod()` 和 `staticmethod()`。

裝飾器語法只是語法糖。以下兩個函式定義在語義上是等效的：

```
def f(arg):
    ...
f = staticmethod(f)

@staticmethod
def f(arg):
    ...
```

Class 也存在相同的概念，但在那比較不常用。關於裝飾器的更多內容，請參閱函式定義和 class 定義的說明文件。

**descriptor (描述器)**

Any object which defines the methods `__get__()`, `__set__()`, or `__delete__()`. When a class attribute is a descriptor, its special binding behavior is triggered upon attribute lookup. Normally, using `a.b` to get, set or

delete an attribute looks up the object named *b* in the class dictionary for *a*, but if *b* is a descriptor, the respective descriptor method gets called. Understanding descriptors is a key to a deep understanding of Python because they are the basis for many features including functions, methods, properties, class methods, static methods, and reference to super classes.

關於描述器 method 的更多資訊，請參閱 descriptors 或描述器使用指南。

### dictionary (字典)

An associative array, where arbitrary keys are mapped to values. The keys can be any object with `__hash__()` and `__eq__()` methods. Called a hash in Perl.

### dictionary comprehension (字典綜合運算)

一種緊密的方法，用來處理一個可迭代物件中的全部或部分元素，`dict` 將處理結果以一個字典回傳。`results = {n: n ** 2 for n in range(10)}` 會生成一個字典，它包含了鍵 *n* 映射到值 *n* \*\* 2。請參閱 comprehensions。

### dictionary view (字典檢視)

從 `dict.keys()`、`dict.values()` 及 `dict.items()` 回傳的物件被稱作字典檢視。它們提供了字典中項目的動態檢視，這表示當字典有變動時，該檢視會反映這些變動。若要限制將字典檢視轉為完整的 list (串列)，須使用 `list(dictview)`。請參閱 dict-views。

### docstring (說明字串)

A string literal which appears as the first expression in a class, function or module. While ignored when the suite is executed, it is recognized by the compiler and put into the `__doc__` attribute of the enclosing class, function or module. Since it is available via introspection, it is the canonical place for documentation of the object.

### duck-typing (鴨子型)

一種程式設計風格，它不是藉由檢查一個物件的型別來確定它是否具有正確的介面；取而代之的是，method 或屬性會單純地被呼叫或使用。（「如果它看起來像一隻鴨子而且叫起來像一隻鴨子，那它一定是一隻鴨子。」）因調介面而非特定型別，精心設計的程式碼能讓多形替代 (polymorphic substitution) 來增進它的靈活性。鴨子型要避免使用 `type()` 或 `isinstance()` 進行測試。（但是請注意，鴨子型可以用抽象基底類 (abstract base class) 來補充。）然而，它通常會用 `hasattr()` 測試，或是 EAFP 程式設計風格。

### EAFP

Easier to ask for forgiveness than permission. (請求寬恕比請求許可更容易。) 這種常見的 Python 編碼風格會先假設有效的鍵或屬性的存在，`try` 在該假設被推翻時再捕獲例外。這種乾且快速的風格，其特色是存在許多的 `try` 和 `except` 陳述式。該技術與許多其他語言 (例如 C) 常見的 LBYL 風格形成了對比。

### expression (運算式)

一段可以被評估求值的語法。一句話，一個運算式就是文字、名稱、屬性存取、運算子或函式呼叫等運算式元件的累積，而這些元件都能回傳一個值。與許多其他語言不同的是，非所有的 Python 語言構造都是運算式。另外有一些 statement (陳述式) 不能被用作運算式，例如 `while`。賦值 (assignment) 也是陳述式，而不是運算式。

### extension module (擴充模組)

一個以 C 或 C++ 編寫的模組，它使用 Python 的 C API 來與核心及使用者程式碼進行互動。

### f-string (f 字串)

以 'f' 或 'F' 前綴的字串文本通常被稱作「f 字串」，它是格式化的字串文本的縮寫。另請參閱 PEP 498。

### file object (檔案物件)

An object exposing a file-oriented API (with methods such as `read()` or `write()`) to an underlying resource. Depending on the way it was created, a file object can mediate access to a real on-disk file or to another type of storage or communication device (for example standard input/output, in-memory buffers, sockets, pipes, etc.). File objects are also called *file-like objects* or *streams*.

實際上，有三種檔案物件：原始的二進制檔案、緩衝的二進制檔案和文字檔案。它們的介面在 `io` 模組中被定義。建立檔案物件的標準方法是使用 `open()` 函式。

**file-like object (類檔案物件)**

*file object* (檔案物件) 的同義字。

**filesystem encoding and error handler (檔案系統編碼和錯誤處理函式)**

Python 所使用的一種編碼和錯誤處理函式，用來解碼來自作業系統的位元組，以及將 Unicode 編碼到作業系統。

檔案系統編碼必須保證能成功解碼所有小於 128 的位元組。如果檔案系統編碼無法提供此保證，則 API 函式會引發 `UnicodeError`。

`sys.getfilesystemencoding()` 和 `sys.getfilesystemencodeerrors()` 函式可用於取得檔案系統編碼和錯誤處理函式。

*filesystem encoding and error handler* (檔案系統編碼和錯誤處理函式) 會在 Python 啟動時由 `PyConfig_Read()` 函式來配置：請參 [filesystem\\_encoding](#)，以及 `PyConfig` 的成員 `filesystem_errors`。

另請參 [locale encoding](#) (區域編碼)。

**finder (尋檢器)**

一個物件，它會嘗試正在被 `import` 的模組尋找 *loader* (載入器)。

從 Python 3.3 開始，有兩種類型的尋檢器：*元路徑尋檢器 (meta path finder)* 會使用 `sys.meta_path`，而*路徑項目尋檢器 (path entry finder)* 會使用 `sys.path_hooks`。

請參 [PEP 302](#)、[PEP 420](#) 和 [PEP 451](#) 以了解更多細節。

**floor division (向下取整除法)**

向下無條件舍去到最接近整數的數學除法。向下取整除法的運算子是 `//`。例如，運算式 `11 // 4` 的計算結果是 `2`，與 `float` (浮點數) 真除法所回傳的 `2.75` 不同。請注意，`(-11) // 4` 的結果是 `-3`，因為 `-2.75` 被向下無條件舍去。請參 [PEP 238](#)。

**function (函式)**

一連串的陳述式，它能向呼叫者回傳一些值。它也可以被傳遞零個或多個引數，這些引數可被使用於函式本體的執行。另請參 [parameter](#) (參數)、[method](#) (方法)，以及 [function](#) 章節。

**function annotation (函式釋)**

函式參數或回傳值的一個 *annotation* (釋)。

函式釋通常被使用於 [型提示](#)：例如，這個函式預期會得到兩個 `int` 引數，會有一個 `int` 回傳值：

```
def sum_two_numbers(a: int, b: int) -> int:
    return a + b
```

函式釋的語法在 [function](#) 章節有詳細解釋。

請參 [variable annotation](#) 和 [PEP 484](#)，皆有此功能的描述。關於釋的最佳實踐方法，另請參 [annotations-howto](#)。

**\_\_future\_\_**

`future` 陳述式：`from __future__ import <feature>`，會指示編譯器使用那些在 Python 未來的發布版本中將成標準的語法或語義，來編譯當前的模組。而 `__future__` 模組則記了 *feature* (功能) 可能的值。透過 `import` 此模組對其變數求值，你可以看見一個新的功能是何時首次被新增到此語言中，以及它何時將會 (或已經) 成預設的功能：

```
>>> import __future__
>>> __future__.division
_Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 8192)
```

**garbage collection (垃圾回收)**

當記憶體不再被使用時，將其釋放的過程。Python 執行垃圾回收，是透過參照計數 (reference counting)，



以及一個能檢測和中斷參照循環 (reference cycle) 的循環垃圾回收器 (cyclic garbage collector) 來完成。垃圾回收器可以使用 `gc` 模組對其進行控制。

### generator (生成器)

一個會回傳 *generator iterator* (生成器迭代器) 的函式。它看起來像一個正常的函式，但不同的是它包含了 `yield` 運算式，能生成一系列的値，這些値可用於 `for` 圈，或是以 `next()` 函式，每次檢索其中的一個値。

這個術語通常用來表示一個生成器函式，但在某些情境中，也可能是表示生成器迭代器。萬一想表達的意思不清楚，那就使用完整的術語，以避免歧義。

### generator iterator (生成器迭代器)

一個由 *generator* (生成器) 函式所建立的物件。

每個 `yield` 會暫停處理程序，記住位置執行狀態 (包括區域變數及擱置中的 `try` 陳述式)。當生成器迭代器回復時，它會從停止的地方繼續執行 (與那些每次調用時都要重新開始的函式有所不同)。

### generator expression (生成器運算式)

一個會回傳迭代器的運算式。它看起來像一個正常的運算式，後面接著一個 `for` 子句，該子句定義了圈變數、範圍以及一個選擇性的 `if` 子句。該組合運算式會在外層函式生成多個値：

```
>>> sum(i*i for i in range(10))           # sum of squares 0, 1, 4, ... 81
285
```

### generic function (泛型函式)

一個由多個函式組成的函式，該函式會對不同的型實作相同的運算。呼叫期間應該使用哪種實作，是由調度演算法 (dispatch algorithm) 來決定。

另請參 *single dispatch* (單一調度) 術語表條目、`functools.singledispatch()` 裝飾器和 **PEP 443**。

### generic type (泛型型)

一個能被參數化 (parameterized) 的 *type* (型)；通常是一個容器型，像是 `list` 和 `dict`。它被用於型提示和解釋。

詳情請參泛型名、**PEP 483**、**PEP 484**、**PEP 585** 和 `typing` 模組。

## GIL

請參 *global interpreter lock* (全域直譯器鎖)。

### global interpreter lock (全域直譯器鎖)

*CPython* 直譯器所使用的機制，用以確保每次都只有一個執行緒能執行 Python 的 *bytecode* (位元組碼)。透過使物件模型 (包括關鍵的型，如 `dict`) 自動地避免行存取 (concurrent access) 的危險，此機制可以簡化 *CPython* 的實作。鎖定整個直譯器，會使直譯器更容易成多執行緒 (multi-threaded)，但代價是會犧牲掉多處理器的機器能提供的一大部分平行性 (parallelism)。

然而，有些擴充模組，無論是標準的或是第三方的，它們被設計成在執行壓縮或雜等計算密集 (computationally intensive) 的任務時，可以解除 GIL。另外，在執行 I/O 時，GIL 總是會被解除。

過去對於建立「無限制執行緒」直譯器 (以更高的精細度鎖定共享資料的直譯器) 的努力未成功，因在一般的單一處理器情況下，效能會有所損失。一般認為，若要克服這個效能問題，會使實作變得雜許多，進而付出更高的維護成本。

### hash-based pyc (雜架構的 pyc)

一個位元組碼 (bytecode) 暫存檔，它使用雜值而不是對應原始檔案的最後修改時間，來確定其有效性。請參 `pyc-invalidation`。

### hashable (可雜的)

An object is *hashable* if it has a hash value which never changes during its lifetime (it needs a `__hash__()` method), and can be compared to other objects (it needs an `__eq__()` method). Hashable objects which compare equal must have the same hash value.

可雜性 (hashability) 使一個物件可用作 dictionary (字典) 的鍵和 set (集合) 的成員，因這些資料結構都在其部使用了雜值。

大多數的 Python 不可變物件都是可雜的；可變的容器 (例如 list 或 dictionary) 不是；而不可變的容器 (例如 tuple (元組) 和 frozenset)，只有當它們的元素是可雜的，它們本身才是可雜的。若物件是使用者自定 class 的實例，則這些物件會被預設可雜的。它們在互相比較時都是不相等的 (除非它們與自己比較)，而它們的雜值則是衍生自它們的 `id()`。

## IDLE

Python 的 Integrated Development and Learning Environment (整合開發與學習環境)。idle 是一個基本的編輯器和直譯器環境，它和 Python 的標準發行版本一起被提供。

## immutable (不可變物件)

一個具有固定值的物件。不可變物件包括數字、字串和 tuple (元組)。這類物件是不能被改變的。如果一個不同的值必須被儲存，則必須建立一個新的物件。它們在需要定雜值的地方，扮演重要的角色，例如 dictionary (字典) 中的一個鍵。

## import path (引入路徑)

一個位置 (或路徑項目) 的列表，而那些位置就是在 import 模組時，會被 *path based finder* (基於路徑的尋檢器) 搜尋模組的位置。在 import 期間，此位置列表通常是來自 `sys.path`，但對於子套件 (subpackage) 而言，它也可能是來自父套件的 `__path__` 屬性。

## importing (引入)

一個過程。一個模組中的 Python 程式碼可以透過此過程，被另一個模組中的 Python 程式碼使用。

## importer (引入器)

一個能尋找及載入模組的物件；它既是 *finder* (尋檢器) 也是 *loader* (載入器) 物件。

## interactive (互動的)

Python 有一個互動式直譯器，這表示你可以在直譯器的提示字元輸入陳述式和運算式，立即執行它們且看到它們的結果。只要啟動 python，不需要任何引數 (可能藉由從你的電腦的主選單選擇它)。這是測試新想法或檢查模塊和包的非常大的方法 (請記住 `help(x)`)。

## interpreted (直譯的)

Python 是一種直譯語言，而不是編譯語言，不過這個區分可能有些模糊，因有位元組碼 (bytecode) 編譯器的存在。這表示原始檔案可以直接被運行，而不需明確地建立另一個執行檔，然後再執行它。直譯語言通常比編譯語言有更短的開發 / 除錯期，不過它們的程式通常也運行得較慢。另請參 *interactive* (互動的)。

## interpreter shutdown (直譯器關閉)

當 Python 直譯器被要求關閉時，它會進入一個特殊階段，在此它逐漸釋放所有被配置的資源，例如模組和各種關鍵部結構。它也會多次呼叫 *垃圾回收器* (*garbage collector*)。這能觸發使用者自定的解構函式 (destructor) 或弱引用的回呼 (weakref callback)，執行其中的程式碼。在關閉階段被執行的程式碼會遇到各種例外，因它所依賴的資源可能不再有作用了 (常見的例子是函式庫模組或是警告機制)。

直譯器關閉的主要原因，是 `__main__` 模組或正被運行的本已經執行完成。

## iterable (可代物件)

An object capable of returning its members one at a time. Examples of iterables include all sequence types (such as list, str, and tuple) and some non-sequence types like dict, *file objects*, and objects of any classes you define with an `__iter__()` method or with a `__getitem__()` method that implements *sequence* semantics.

Iterables can be used in a for loop and in many other places where a sequence is needed (`zip()`, `map()`, ...). When an iterable object is passed as an argument to the built-in function `iter()`, it returns an iterator for the object. This iterator is good for one pass over the set of values. When using iterables, it is usually not necessary to call `iter()` or deal with iterator objects yourself. The for statement does that automatically for you, creating a temporary unnamed variable to hold the iterator for the duration of the loop. See also *iterator*, *sequence*, and *generator*.

**iterator (迭代器)**

An object representing a stream of data. Repeated calls to the iterator's `__next__()` method (or passing it to the built-in function `next()`) return successive items in the stream. When no more data are available a `StopIteration` exception is raised instead. At this point, the iterator object is exhausted and any further calls to its `__next__()` method just raise `StopIteration` again. Iterators are required to have an `__iter__()` method that returns the iterator object itself so every iterator is also iterable and may be used in most places where other iterables are accepted. One notable exception is code which attempts multiple iteration passes. A container object (such as a `list`) produces a fresh new iterator each time you pass it to the `iter()` function or use it in a `for` loop. Attempting this with an iterator will just return the same exhausted iterator object used in the previous iteration pass, making it appear like an empty container.

在 `typeiter` 文中可以找到更多資訊。

**CPython 實作細節：** CPython does not consistently apply the requirement that an iterator define `__iter__()`.

**key function (鍵函式)**

鍵函式或理序函式 (collation function) 是一個可呼叫 (callable) 函式，它會回傳一個用於排序 (sorting) 或定序 (ordering) 的值。例如，`locale.strxfrm()` 被用來產生一個了解區域特定排序慣例的排序鍵。

Python 中的許多工具，都接受以鍵函式來控制元素被定序或分組的方式。它們包括 `min()`、`max()`、`sorted()`、`list.sort()`、`heapq.merge()`、`heapq.nsmallest()`、`heapq.nlargest()` 和 `itertools.groupby()`。

有幾種方法可以建立一個鍵函式。例如，`str.lower()` method 可以作不分大小寫排序的鍵函式。或者，一個鍵函式也可以從 `lambda` 運算式被建造，例如 `lambda r: (r[0], r[2])`。另外，`operator.attrgetter()`、`operator.itemgetter()` 和 `operator.methodcaller()` 三個鍵函式的建構函式 (constructor)。關於如何建立和使用鍵函式的範例，請參閱如何排序。

**keyword argument (關鍵字引數)**

請參閱 [argument](#) (引數)。

**lambda**

由單一 *expression* (運算式) 所組成的一個匿名行函式 (inline function)，於該函式被呼叫時求值。建立 `lambda` 函式的語法是 `lambda [parameters]: expression`

**LBYL**

Look before you leap. (三思而後行。) 這種編碼風格會在進行呼叫或查找之前，明確地測試先條件。這種風格與 *EAFP* 方式形成對比，且它的特色是會有許多 `if` 陳述式的存在。

在一個多執行緒環境中，LBYL 方式有在「三思」和「後行」之間引入了競態條件 (race condition) 的風險。例如以下程式碼 `if key in mapping: return mapping[key]`，如果另一個執行緒在測試之後但在查找之前，從 `mapping` 中移除了 `key`，則該程式碼就會失效。這個問題可以用鎖 (lock) 或使用 *EAFP* 編碼方式來解。

**list (串列)**

A built-in Python *sequence*. Despite its name it is more akin to an array in other languages than to a linked list since access to elements is  $O(1)$ .

**list comprehension (串列綜合運算)**

一種用來處理一個序列中的全部或部分元素，將處理結果以一個 `list` 回傳的簡要方法。`result = ['{:04x}'.format(x) for x in range(256) if x % 2 == 0]` 會產生一個字串 `list`，其中包含 0 到 255 範圍內，所有偶數的十六進位數 (0x.)。 `if` 子句是選擇性的。如果省略它，則 `range(256)` 中的所有元素都會被處理。

**loader (載入器)**

一個能載入模組的物件。它必須定義一個名 `load_module()` 的 `method` (方法)。載入器通常是被 *finder* (尋檢器) 回傳。更多細節請參閱 [PEP 302](#)，關於 *abstract base class* (抽象基底類)，請參閱 `importlib.abc.Loader`。

**locale encoding (區域編碼)**

在 Unix 上, 它是 LC\_CTYPE 區域設定的編碼。它可以用 `locale.setlocale(locale.LC_CTYPE, new_locale)` 來設定。

在 Windows 上, 它是 ANSI 代碼頁 (code page, 例如 "cp1252")。

在 Android 和 VxWorks 上, Python 使用 "utf-8" 作區域編碼。

`locale.getencoding()` can be used to get the locale encoding.

也請參考 *filesystem encoding and error handler*。

**magic method (魔術方法)**

*special method* (特殊方法) 的一個非正式同義詞。

**mapping (對映)**

一個容器物件, 它支援任意鍵的查找, 且能實作 abstract base classes (抽象基底類) 中, `collections.abc.Mapping` 或 `collections.abc.MutableMapping` 所指定的 method。範例包括 `dict`、`collections.defaultdict`、`collections.OrderedDict` 和 `collections.Counter`。

**meta path finder (元路徑尋檢器)**

一種經由搜尋 `sys.meta_path` 而回傳的 *finder* (尋檢器)。元路徑尋檢器與路徑項目尋檢器 (*path entry finder*) 相關但是不同。

關於元路徑尋檢器實作的 method, 請參 `importlib.abc.MetaPathFinder`。

**metaclass (元類)**

一種 class 的 class。Class 定義過程會建立一個 class 名稱、一個 class dictionary (字典), 以及一個 base class (基底類) 的列表。Metaclass 負責接受這三個引數, 建立該 class。大多數的物件導向程式語言會提供一個預設的實作。Python 的特之處在於它能建立自訂的 metaclass。大部分的使用者從未需要此工具, 但是當需要時, metaclass 可以提供大且優雅的解方案。它們已被用於記屬性存取、增加執行緒安全性、追物件建立、實作單例模式 (singleton), 以及許多其他的任務。

更多資訊可以在 metaclasses 章節中找到。

**method (方法)**

一個在 class 本體被定義的函式。如果 method 作其 class 實例的一個屬性被呼叫, 則它將會得到該實例物件成它的第一個 *argument* (引數) (此引數通常被稱 `self`)。請參 *function* (函式) 和 *nested scope* (巢狀作用域)。

**method resolution order (方法解析順序)**

方法解析順序是在查找某個成員的過程中, base class (基底類) 被搜尋的順序。關於第 2.3 版至今, Python 直譯器所使用的演算法細節, 請參 *Python 2.3 版方法解析順序*。

**module (模組)**

一個擔任 Python 程式碼的組織單位 (organizational unit) 的物件。模組有一個命名空間, 它包含任意的 Python 物件。模組是藉由 *importing* 的過程, 被載入至 Python。

另請參 *package* (套件)。

**module spec (模組規格)**

一個命名空間, 它包含用於載入模組的 import 相關資訊。它是 `importlib.machinery.ModuleSpec` 的一個實例。

**MRO**

請參 *method resolution order* (方法解析順序)。

**mutable (可變物件)**

可變物件可以改變它們的值, 但維持它們的 `id()`。另請參 *immutable* (不可變物件)。

**named tuple (附名元組)**

術語「named tuple (附名元組)」是指從 tuple 繼承的任何型或 class, 且它的可索引 (indexable) 元素也可以用附名屬性來存取。這些型或 class 也可以具有其他的特性。



有些型是 **named tuple**，包括由 `time.localtime()` 和 `os.stat()` 回傳的值。另一個例子是 `sys.float_info`：

```
>>> sys.float_info[1]           # indexed access
1024
>>> sys.float_info.max_exp      # named field access
1024
>>> isinstance(sys.float_info, tuple) # kind of tuple
True
```

Some named tuples are built-in types (such as the above examples). Alternatively, a named tuple can be created from a regular class definition that inherits from `tuple` and that defines named fields. Such a class can be written by hand, or it can be created by inheriting `typing.NamedTuple`, or with the factory function `collections.namedtuple()`. The latter techniques also add some extra methods that may not be found in hand-written or built-in named tuples.

### namespace (命名空間)

變數被儲存的地方。命名空間是以 **dictionary** (字典) 被實作。有區域的、全域的及建立的命名空間，而在物件中 (在 **method** 中) 也有巢狀的命名空間。命名空間藉由防止命名衝突，來支援模組化。例如，函式 `builtins.open` 和 `os.open()` 是透過它們的命名空間來區分彼此。命名空間也藉由明確地區分是哪個模組在實作一個函式，來增進可讀性及可維護性。例如，寫出 `random.seed()` 或 `itertools.islice()` 明確地表示，這些函式分是由 `random` 和 `itertools` 模組在實作。

### namespace package (命名空間套件)

一個 **PEP 420 package** (套件)，它只能作子套件 (subpackage) 的一個容器。命名空間套件可能沒有實體的表示法，而且具體來它們不像是一個 **regular package** (正規套件)，因它們有 `__init__.py` 這個檔案。

另請參 **module** (模組)。

### nested scope (巢狀作用域)

能參照外層定義 (enclosing definition) 中的變數的能力。舉例來，一個函式如果是在另一個函式中被定義，則它便能參照外層函式中的變數。請注意，在預設情況下，巢狀作用域僅適用於參照，而無法用於賦值。區域變數能在最層作用域中讀取及寫入。同樣地，全域變數是在全域命名空間中讀取及寫入。`nonlocal` 容許對外層作用域進行寫入。

### new-style class (新式類)

Old name for the flavor of classes now used for all class objects. In earlier Python versions, only new-style classes could use Python's newer, versatile features like `__slots__`, descriptors, properties, `__getattr__()`, class methods, and static methods.

### object (物件)

具有狀態 (屬性或值) 及被定義的行 (method) 的任何資料。它也是任何 **new-style class** (新式類) 的最終 **base class** (基底類)。

### package (套件)

一個 Python 的 **module** (模組)，它可以包含子模組 (submodule) 或是遞的子套件 (subpackage)。技術上而言，套件就是具有 `__path__` 屬性的一個 Python 模組。

另請參 **regular package** (正規套件) 和 **namespace package** (命名空間套件)。

### parameter (參數)

在 **function** (函式) 或 **method** 定義中的一個命名實體 (named entity)，它指明該函式能接受的一個 **argument** (引數)，或在某些情況下指示多個引數。共有五種不同的參數類型：

- **positional-or-keyword** (位置或關鍵字)：指明一個可以按照位置或是作關鍵字引數被傳遞的引數。這是參數的預設類型，例如以下的 `foo` 和 `bar`：

```
def func(foo, bar=None): ...
```

- *positional-only* (僅限位置): 指明一個只能按照位置被提供的引數。在函式定義的參數列表中包含一個 / 字元, 就可以在該字元前面定義僅限位置參數, 例如以下的 *posonly1* 和 *posonly2*:

```
def func(posonly1, posonly2, /, positional_or_keyword): ...
```

- *keyword-only* (僅限關鍵字): 指明一個只能以關鍵字被提供的引數。在函式定義的參數列表中, 包含一個任意數量位置參數 (var-positional parameter) 或是單純的 \* 字元, 就可以在其後方定義僅限關鍵字參數, 例如以下的 *kw\_only1* 和 *kw\_only2*:

```
def func(arg, *, kw_only1, kw_only2): ...
```

- *var-positional* (任意數量位置): 指明一串能以任意序列被提供的位置引數 (在已被其他參數接受的任何位置引數之外)。這類參數是透過在其參數名稱字首加上 \* 來定義的, 例如以下的 *args*:

```
def func(*args, **kwargs): ...
```

- *var-keyword* (任意數量關鍵字): 指明可被提供的任意數量關鍵字引數 (在已被其他參數接受的任何關鍵字引數之外)。這類參數是透過在其參數名稱字首加上 \*\* 來定義的, 例如上面範例中的 *kwargs*。

參數可以指明引數是選擇性的或必需的, 也可以一些選擇性的引數指定預設值。

另請參閱術語表的 *argument* (引數) 條目、常見問題中的引數和參數之間的差別、`inspect.Parameter` class、function 章節, 以及 [PEP 362](#)。

### path entry (路徑項目)

在 `import path` (引入路徑) 中的一個位置, 而 *path based finder* (基於路徑的尋檢器) 會參考該位置來尋找要 import 的模組。

### path entry finder (路徑項目尋檢器)

被 `sys.path_hooks` 中的一個可呼叫物件 (callable) (意即一個 *path entry hook*) 所回傳的一種 *finder*, 它知道如何以一個 *path entry* 定位模組。

關於路徑項目尋檢器實作的 method, 請參閱 `importlib.abc.PathEntryFinder`。

### path entry hook (路徑項目)

A callable on the `sys.path_hooks` list which returns a *path entry finder* if it knows how to find modules on a specific *path entry*.

### path based finder (基於路徑的尋檢器)

預設的元路徑尋檢器 (*meta path finder*) 之一, 它會在一個 `import path` 中搜尋模組。

### path-like object (類路徑物件)

一個表示檔案系統路徑的物件。類路徑物件可以是一個表示路徑的 `str` 或 `bytes` 物件, 或是一個實作 `os.PathLike` 協定的物件。透過呼叫 `os.fspath()` 函式, 一個支援 `os.PathLike` 協定的物件可以被轉換成 `str` 或 `bytes` 檔案系統路徑; 而 `os.fsdecode()` 及 `os.fsencode()` 則分別可用於確保 `str` 及 `bytes` 的結果。由 [PEP 519](#) 引入。

## PEP

Python Enhancement Proposal (Python 增進提案)。PEP 是一份設計明文件, 它能向 Python 社群提供資訊, 或是描述 Python 的一個新功能或該功能的程序和環境。PEP 應該要提供簡潔的技術規範以及被提案功能的運作原理。

PEP 的存在目的, 是要成為重大新功能的提案、社群中關於某個問題的意見收集, 以及已納入 Python 的設計決策的記錄, 這些過程的主要機制。PEP 的作者要負責在社群中建立共識及反對意見。

請參閱 [PEP 1](#)。

### portion (部分)

在單一目標中的一組檔案 (也可能儲存在一個 zip 檔中), 這些檔案能對一個命名空間套件 (namespace package) 有所貢獻, 如同 [PEP 420](#) 中的定義。

**positional argument (位置引數)**

請參閱 [argument](#) (引數)。

**provisional API (暫行 API)**

暫行 API 是指，從標準函式庫的向後相容性 (backwards compatibility) 保證中，故意被排除的 API。雖然此類介面，只要它們被標示為暫行的，理論上不會有重大的變更，但如果核心開發人員認為有必要，也可能會出現向後不相容的變更（甚至包括移除該介面）。這種變更不會無端地發生——只有 API 被納入之前未察覺的嚴重基本缺陷被揭露時，它們才會發生。

即使對於暫行 API，向後不相容的變更也會被視為「最後的解決方案」——對於任何被發現的問題，仍然會盡可能找出一個向後相容的解決方案。

這個過程使得標準函式庫能隨著時間不斷進化，而避免耗費過長的時間去鎖定有問題的設計錯誤。請參閱 [PEP 411](#) 了解更多細節。

**provisional package (暫行套件)**

請參閱 [provisional API](#) (暫行 API)。

**Python 3000**

Python 3.x 系列版本的暱稱（很久以前創造的，當時第 3 版的發布是在很遠的未來。）也可以縮寫為「Py3k」。

**Pythonic (Python 風格的)**

一個想法或一段程式碼，它應用了 Python 語言最常見的慣用語，而不是使用其他語言常見的概念來實作程式碼。例如，Python 中常見的一種習慣用法，是使用一個 `for` 陳述式，對一個可迭代物件的所有元素進行遍歷。許多其他語言也有這種類型的架構，所以不熟悉 Python 的人有時會使用一個數值計數器來代替：

```
for i in range(len(food)):
    print(food[i])
```

相較之下，以下方法更簡潔、更具有 Python 風格：

```
for piece in food:
    print(piece)
```

**qualified name (限定名稱)**

一個「點分隔名稱」，它顯示從一個模組的全域作用域到該模組中定義的 `class`、函式或 `method` 的「路徑」，如 [PEP 3155](#) 中的定義。對於頂層的函式和 `class` 而言，限定名稱與其物件名稱相同：

```
>>> class C:
...     class D:
...         def meth(self):
...             pass
...
>>> C.__qualname__
'C'
>>> C.D.__qualname__
'C.D'
>>> C.D.meth.__qualname__
'C.D.meth'
```

當用於引用模組時，完全限定名稱 (*fully qualified name*) 是表示該模組的完整點分隔路徑，包括任何的父套件，例如 `email.mime.text`：

```
>>> import email.mime.text
>>> email.mime.text.__name__
'email.mime.text'
```

**reference count (參照計數)**

對於一個物件的參照次數。當一個物件的參照計數下降到零時，它會被解除配置 (deallocated)。參照計數通常在 Python 程式碼中看不到，但它 [是 CPython 實作的一個關鍵元素](#)。程式設計師可以呼叫 `getrefcount()` 函式來回傳一個特定物件的參照計數。

**regular package (正規套件)**

一個傳統的 *package* (套件)，例如一個包含 `__init__.py` 檔案的目錄。

另請參 [namespace package](#) (命名空間套件)。

**`__slots__`**

在 `class` 部的一個宣告，它藉由預先宣告實例屬性的空間，以及消除實例 `dictionary` (字典)，來節省記憶體。雖然該技術很普遍，但它有點難以正確地使用，最好保留給那種在一個記憶體關鍵 (memory-critical) 的應用程式中存在大量實例的罕見情況。

**sequence (序列)**

An *iterable* which supports efficient element access using integer indices via the `__getitem__()` special method and defines a `__len__()` method that returns the length of the sequence. Some built-in sequence types are `list`, `str`, `tuple`, and `bytes`. Note that `dict` also supports `__getitem__()` and `__len__()`, but is considered a mapping rather than a sequence because the lookups use arbitrary *immutable* keys rather than integers.

The `collections.abc.Sequence` abstract base class defines a much richer interface that goes beyond just `__getitem__()` and `__len__()`, adding `count()`, `index()`, `__contains__()`, and `__reversed__()`. Types that implement this expanded interface can be registered explicitly using `register()`. For more documentation on sequence methods generally, see [Common Sequence Operations](#).

**set comprehension (集合綜合運算)**

一種緊密的方法，用來處理一個可代物件中的全部或部分元素，[將處理結果以一個 set 回傳](#)。  
`results = {c for c in 'abracadabra' if c not in 'abc'}` 會生一個字串 `set: {'r', 'd'}`。請參 [comprehensions](#)。

**single dispatch (單一調度)**

*generic function* (泛型函式) 調度的一種形式，在此，實作的選擇是基於單一引數的型。

**slice (切片)**

一個物件，它通常包含一段 *sequence* (序列) 的某一部分。建立一段切片的方法是使用下標符號 (subscript notation) `[]`，若要給出多個數字，則在數字之間使用冒號，例如 `variable_name[1:3:5]`。在括號 (下標) 符號的部，會使用 `slice` 物件。

**special method (特殊方法)**

一種會被 Python 自動呼叫的 `method`，用於對某種型執行某種運算，例如加法。這種 `method` 的名稱會在開頭和結尾有兩個下底。Special method 在 `specialnames` 中有詳細明。

**statement (陳述式)**

陳述式是一個套組 (suite，一個程式碼「區塊」) 中的一部分。陳述式可以是一個 *expression* (運算式)，或是含有關鍵字 (例如 `if`、`while` 或 `for`) 的多種結構之一。

**static type checker**

An external tool that reads Python code and analyzes it, looking for issues such as incorrect types. See also *type hints* and the `typing` module.

**strong reference (參照)**

In Python's C API, a strong reference is a reference to an object which is owned by the code holding the reference. The strong reference is taken by calling `Py_INCREF()` when the reference is created and released with `Py_DECREF()` when the reference is deleted.

`Py_NewRef()` 函式可用於建立一個對物件的參照。通常，在退出參照的作用域之前，必須在該參照上呼叫 `Py_DECREF()` 函式，以避免漏一個參照。

另請參 [borrowed reference](#) (借用參照)。

### text encoding (文字編碼)

Python 中的字串是一個 Unicode 碼點 (code point) 的序列 (範圍在 U+0000 -- U+10FFFF 之間)。若要儲存或傳送一個字串，它必須被序列化一個位元組序列。

將一個字串序列化位元組序列，稱「編碼」，而從位元組序列重新建立該字串則稱「解碼 (decoding)」。

有多種不同的文字序列化編解碼器 (codecs)，它們被統稱「文字編碼」。

### text file (文字檔案)

一個能讀取和寫入 `str` 物件的一個 *file object* (檔案物件)。通常，文字檔案實際上是存取位元組導向的資料流 (byte-oriented datastream) 會自動處理 *text encoding* (文字編碼)。文字檔案的例子有：以文字模式 ('r' 或 'w') 開的檔案、`sys.stdin`、`sys.stdout` 以及 `io.StringIO` 的實例。

另請參 *binary file* (二進制檔案)，它是一個能讀取和寫入類位元組串物件 (*bytes-like object*) 的檔案物件。

### triple-quoted string (三引號字串)

由三個雙引號 (") 或單引號 (') 的作邊界的一個字串。雖然它們有提供於單引號字串的任何額外功能，但基於許多原因，它們仍是很有用的。它們讓你可以字串中包含未跳 (unescaped) 的單引號和雙引號，而且它們不需使用連續字元 (continuation character) 就可以跨越多行，這使得它們在編寫明字串時特別有用。

### type (型)

一個 Python 物件的型定了它是什類型的物件；每個物件都有一個型。一個物件的型可以用它的 `__class__` 屬性來存取，或以 `type(obj)` 來檢索。

### type alias (型名)

一個型的同義詞，透過將型指定給一個識符 (identifier) 來建立。

型名對於簡化型提示 (*type hint*) 很有用。例如：

```
def remove_gray_shades(
    colors: list[tuple[int, int, int]]) -> list[tuple[int, int, int]]:
    pass
```

可以寫成這樣，更具有可讀性：

```
Color = tuple[int, int, int]

def remove_gray_shades(colors: list[Color]) -> list[Color]:
    pass
```

請參 `typing` 和 **PEP 484**，有此功能的描述。

### type hint (型提示)

一種 *annotation* (釋)，它指定一個變數、一個 class 屬性或一個函式的參數或回傳值的預期型。

Type hints are optional and are not enforced by Python but they are useful to *static type checkers*. They can also aid IDEs with code completion and refactoring.

全域變數、class 屬性和函式 (不含區域變數) 的型提示，都可以使用 `typing.get_type_hints()` 來存取。

請參 `typing` 和 **PEP 484**，有此功能的描述。

### universal newlines (通用行字元)

一種解譯文字流 (text stream) 的方式，會將以下所有的情識一行的結束：Unix 行尾慣例 '\n'、Windows 慣例 '\r\n' 和舊的 Macintosh 慣例 '\r'。請參 **PEP 278** 和 **PEP 3116**，以及用於 `bytes.splitlines()` 的附加用途。



**variable annotation (變數釋)**

一個變數或 class 屬性的 *annotation* (釋)。

釋變數或 class 屬性時，賦值是選擇性的：

```
class C:
    field: 'annotation'
```

變數釋通常用於型提示 (*type hint*)：例如，這個變數預期會取得 `int` (整數) 值：

```
count: int = 0
```

變數釋的語法在 `annassign` 章節有詳細的解釋。

請參 *function annotation* (函式釋)、**PEP 484** 和 **PEP 526**，皆有此功能的描述。關於釋的最佳實踐方法，另請參 `annotations-howto`。

**virtual environment (擬環境)**

一個協作隔離 (*cooperatively isolated*) 的執行環境，能讓 Python 的使用者和應用程式得以安裝和升級 Python 發套件，而不會對同一個系統上運行的其他 Python 應用程式的行產生干擾。

另請參 `venv`。

**virtual machine (擬機器)**

一部完全由軟體所定義的電腦 (*computer*)。Python 的擬機器會執行由 *bytecode* (位元組碼) 編譯器所發出的位元組碼。

**Zen of Python (Python 之)**

Python 設計原則與哲學的列表，其容有助於理解和使用此語言。此列表可以透過在互動式提示字元後輸入 `import this` 來找到它。

---

## 關於這些文檔文件

---

這些文檔文件是透過 [Sphinx](#)（一個專為 Python 文檔文件所撰寫的文件處理器）將使用 [reStructuredText](#) 撰寫的原始檔轉換而成。

如同 Python 自身，透過自願者的努力下輸出文件與封裝後自動化執行工具。若想要回報臭蟲，請見 [reporting-bugs](#) 頁面，包含相關資訊。我們永遠歡迎新的自願者加入！

致謝：

- Fred L. Drake, Jr., 原始 Python 文件工具集的創造者以及一大部份內容的作者；
- 創造 [reStructuredText](#) 和 [Docutils](#) 工具組的 [Docutils](#) 專案；
- Fredrik Lundh 先生，Sphinx 從他的 [Alternative Python Reference](#) 計劃中獲得許多的好主意。

### B.1 Python 文件的貢獻者們

許多人都曾為 Python 這門語言、Python 標準函式庫和 Python 文檔文件貢獻過。Python 所發出的原始碼中含有部份貢獻者的清單，請見 [Misc/ACKS](#)。

正因為 Python 社群的撰寫與貢獻才有這份這棒的文檔文件 -- 感謝所有貢獻過的人們！





## 沿革與授權

## C.1 軟體沿革

Python 是由荷蘭數學和計算機科學研究學會（CWI，見 <https://www.cwi.nl/>）的 Guido van Rossum 於 1990 年代早期所創造，目的是作一種稱作 ABC 語言的後繼者。儘管 Python 包含了許多來自其他人的貢獻，Guido 仍是其主要作者。

1995 年，Guido 在維吉尼亞州雷斯頓的國家創新研究公司（CNRI，見 <https://www.cnri.reston.va.us/>）繼續他在 Python 的工作，在那發行了該軟體的多個版本。

2000 年五月，Guido 和 Python 核心開發團隊轉移到 BeOpen.com 成立了 BeOpen PythonLabs 團隊。同年十月，PythonLabs 團隊轉移到 Digital Creations（現 Zope Corporation；見 <https://www.zope.org/>）。2001 年，Python 軟體基金會（PSF，見 <https://www.python.org/psf/>）成立，這是一個專擁有 Python 相關的智慧財產權而創立的非營利組織。Zope Corporation 是 PSF 的一個贊助會員。

所有的 Python 版本都是開源的（有關開源的定義，參見 <https://opensource.org/>）。歷史上，大多數但非全部的 Python 版本，也是 GPL 相容的；以下表格總結各個版本的差異。

發行版本	源自	年份	擁有者	GPL 相容性？
0.9.0 至 1.2	不適用	1991-1995	CWI	是
1.3 至 1.5.2	1.2	1995-1999	CNRI	是
1.6	1.5.2	2000	CNRI	否
2.0	1.6	2000	BeOpen.com	否
1.6.1	1.6	2001	CNRI	否
2.1	2.0+1.6.1	2001	PSF	否
2.0.1	2.0+1.6.1	2001	PSF	是
2.1.1	2.1+2.0.1	2001	PSF	是
2.1.2	2.1.1	2002	PSF	是
2.1.3	2.1.2	2002	PSF	是
2.2 以上	2.1.1	2001 至今	PSF	是

**備註：**GPL 相容不表示我們是在 GPL 下發 Python。不像 GPL，所有的 Python 授權都可以讓您發修改後的版本，但不一定要使您的變更成開源。GPL 相容的授權使得 Python 可以結合其他在 GPL 下發的軟體一起使用；但其它的授權則不行。

感謝許多的外部志工，在 Guido 指導下的付出，使得這些版本的發成可能。

## C.2 關於存取或以其他方式使用 Python 的合約條款

Python 軟體和明文件的授權是基於 *PSF* 授權合約。

從 Python 3.8.6 開始，明文件中的範例、程式庫和其他程式碼，是被雙重授權 (dual licensed) 於 PSF 授權合約以及 *Zero-Clause BSD* 授權。

有些被納入 Python 中的軟體是基於不同的授權。這些授權將會與其授權之程式碼一起被列出。關於這些授權的不完整清單，請參被收軟體的授權與致謝。

### C.2.1 用於 PYTHON 3.11.8 的 PSF 授權合約

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),  
→ and  
the Individual or Organization ("Licensee") accessing and otherwise using  
→ Python  
3.11.8 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby  
grants Licensee a nonexclusive, royalty-free, world-wide license to  
→ reproduce,  
analyze, test, perform and/or display publicly, prepare derivative works,  
distribute, and otherwise use Python 3.11.8 alone or in any derivative  
version, provided, however, that PSF's License Agreement and PSF's notice  
→ of  
copyright, i.e., "Copyright © 2001–2023 Python Software Foundation; All  
→ Rights  
Reserved" are retained in Python 3.11.8 alone or in any derivative version  
prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or  
incorporates Python 3.11.8 or any part thereof, and wants to make the  
derivative work available to others as provided herein, then Licensee  
→ hereby  
agrees to include in any such work a brief summary of the changes made to  
→ Python  
3.11.8.
4. PSF is making Python 3.11.8 available to Licensee on an "AS IS" basis.  
PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF  
EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION  
→ OR  
WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT  
→ THE  
USE OF PYTHON 3.11.8 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.11.8 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.11.8, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 3.11.8, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## C.2.2 用於 PYTHON 2.0 的 BEOPEN.COM 授權合約

### BEOPEN PYTHON 開源授權合約第 1 版

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").
2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.
3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

(繼續下一頁)

(繼續上一頁)

6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions granted on that web page.
7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

### C.2.3 用於 PYTHON 1.6.1 的 CNRI 授權合約

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the internet using the following URL: <http://hdl.handle.net/1895.22/1013>."
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

(繼續下一頁)

(繼續上一頁)

7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## C.2.4 用於 PYTHON 0.9.0 至 1.2 的 CWI 授權合約

Copyright © 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## C.2.5 用於 PYTHON 3.11.8 明文文件程式碼的 ZERO-CLAUSE BSD 授權

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## C.3 被收錄軟體的授權與致謝

本節是一個不完整但持續增加的授權與致謝清單，對象是在 Python 發行版本中所收錄的第三方軟體。

### C.3.1 Mersenne Twister

`_random` 模組包含了以 <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> 的下載內容為基礎的程式碼。以下是原始程式碼的完整聲明：

```
A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using init_genrand(seed)
or init_by_array(init_key, key_length).

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

3. The names of its contributors may not be used to endorse or promote
   products derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Any feedback is very welcome.
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)
```

### C.3.2 Sockets

The `socket` module uses the functions, `getaddrinfo()`, and `getnameinfo()`, which are coded in separate source files from the WIDE Project, <https://www.wide.ad.jp/>.

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### C.3.3 非同步 socket 服務

`asynchat` 和 `asyncore` 模組包含以下聲明:

Copyright 1996 by Sam Rushing

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Sam Rushing not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

### C.3.4 Cookie 管理

http.cookies 模組包含以下聲明：

```
Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>

    All Rights Reserved

Permission to use, copy, modify, and distribute this software
and its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Timothy O'Malley not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR
ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

### C.3.5 執行追F

trace 模組包含以下聲明：

```
portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err... reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
http://zooko.com/
mailto:zooko@zooko.com

Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.

Permission to use, copy, modify, and distribute this Python software and
its associated documentation for any purpose without fee is hereby
granted, provided that the above copyright notice appears in all copies,
and that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of neither Automatrix,
Bioreason or Mojam Media be used in advertising or publicity pertaining to
distribution of the software without specific, written prior permission.
```



### C.3.6 UUencode 與 UUdecode 函式

uu 模組包含以下聲明：

```
Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
    All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
  version is still 5 times faster, though.
- Arguments more compliant with Python standard
```

### C.3.7 XML 遠端程序呼叫

xmlrpc.client 模組包含以下聲明：

```
The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its
associated documentation, you agree that you have read, understood,
and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and
its associated documentation for any purpose and without fee is
hereby granted, provided that the above copyright notice appears in
all copies, and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Secret Labs AB or the author not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANT-
ABILITY AND FITNESS. IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR
BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
```

(繼續下一頁)

(繼續上一頁)

ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

### C.3.8 test\_epoll

The `test.test_epoll` module contains the following notice:

Copyright (c) 2001–2006 Twisted Matrix Laboratories.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### C.3.9 Select queue

`select` 模組對於 `kqueue` 介面包含以下聲明:

Copyright (c) 2000 Doug White, 2006 James Knight, 2007 Christian Heimes  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY

(繼續下一頁)

(繼續上一頁)

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### C.3.10 SipHash24

Python/pyhash.c 檔案包含 Marek Majkowski 基於 Dan Bernstein 的 SipHash24 演算法的實作。它包含以下聲明：

```
<MIT License>
Copyright (c) 2013 Marek Majkowski <marek@popcount.org>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
</MIT License>

Original location:
  https://github.com/majek/csiphash/

Solution inspired by code from:
  Samuel Neves (supercop/crypto_auth/siphash24/little)
  djb (supercop/crypto_auth/siphash24/little2)
  Jean-Philippe Aumasson (https://131002.net/siphash/siphash24.c)
```

### C.3.11 strtod 與 dtoa

Python/dtoa.c 檔案提供了 C 的 `dtoa` 和 `strtod` 函式，用於將 C 的雙精度浮點數和字串互相轉換。該檔案是衍生自 David M. Gay 建立的同名檔案，後者現在可以從 <https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c> 下載。於 2009 年 3 月 16 日所檢索的原始檔案包含以下版權與授權聲明：

```
/* *****
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose without fee is hereby granted, provided that this entire notice
 * is included in all copies of any software which is or includes a copy
 * or modification of this software and in all copies of the supporting
 * documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
 * REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
 * OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
 */
```

(繼續下一頁)

(繼續上一頁)

```
*
*****/
```

### C.3.12 OpenSSL

The modules `hashlib`, `posix`, `ssl`, `crypt` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows and macOS installers for Python may include a copy of the OpenSSL libraries, so we include a copy of the OpenSSL license here. For the OpenSSL 3.0 release, and later releases derived from that, the Apache License v2 applies:

```
Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/
```

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of,

(繼續下一頁)

(繼續上一頁)

the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work,

(繼續下一頁)

(繼續上一頁)

excluding those notices that do not pertain to any part of the Derivative Works; and

- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the

(繼續下一頁)

(繼續上一頁)

Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

### C.3.13 expat

The pyexpat extension is built using an included copy of the expat sources unless the build is configured `--with-system-expat`:

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd  
and Clark Cooper

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### C.3.14 libffi

除非在建置 `_ctypes` 擴充時設定 `--with-system-libffi`，否則該擴充會用一個含 `libffi` 原始碼的副本來建置：

```
Copyright (c) 1996-2008 Red Hat, Inc and others.
```

```
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
`Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```

```
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED `AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

### C.3.15 zlib

如果在系統上找到的 `zlib` 版本太舊以致於無法用於建置 `zlib` 擴充，則該擴充會用一個含 `zlib` 原始碼的副本來建置：

```
Copyright (C) 1995-2011 Jean-loup Gailly and Mark Adler
```

```
This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.
```

```
Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:
```

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

```
Jean-loup Gailly
jloup@gzip.org
```

```
Mark Adler
madler@alumni.caltech.edu
```



### C.3.16 cfuhash

tracemalloc 使用的雜表 (hash table) 實作，是以 cfuhash 專案基礎：

```
Copyright (c) 2005 Don Owens
All rights reserved.
```

```
This code is released under the BSD license:
```

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

```
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

### C.3.17 libmpdec

除非在建置 `_decimal` 模組時設定 `--with-system-libmpdec`，否則該模組會用一個含 `libmpdec` 函式庫的副本來建置：

```
Copyright (c) 2008-2020 Stefan Krah. All rights reserved.
```

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(繼續下一頁)

(繼續上一頁)

```
THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

### C.3.18 W3C C14N 測試套件

test 程式包中的 C14N 2.0 測試套件 (Lib/test/xmltestdata/c14n-20/) 是從 W3C 網站 <https://www.w3.org/TR/xml-c14n2-testcases/> 被檢索，且是基於 3-clause BSD 授權被發：

```
Copyright (c) 2013 W3C(R) (MIT, ERCIM, Keio, Beihang),
All Rights Reserved.
```

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

- \* Redistributions of works must retain the original copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the original copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of the W3C nor the names of its contributors may be used to endorse or promote products derived from this work without specific prior written permission.

```
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

### C.3.19 Audioop

The audioop module uses the code base in g771.c file of the SoX project. <https://sourceforge.net/projects/sox/files/sox/12.17.7/sox-12.17.7.tar.gz>

This source code is a product of Sun Microsystems, Inc. and is provided for unrestricted use. Users may copy or modify this source code without charge.

SUN SOURCE CODE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

Sun source code is provided with no support and without any obligation on the part of Sun Microsystems, Inc. to assist in its use, correction, modification or enhancement.

SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY THIS SOFTWARE OR ANY PART THEREOF.

In no event will Sun Microsystems, Inc. be liable for any lost revenue or profits or other special, indirect and consequential damages, even if Sun has been advised of the possibility of such damages.

Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, California 94043

### C.3.20 asyncio

Parts of the asyncio module are incorporated from uvloop 0.16, which is distributed under the MIT license:

Copyright (c) 2015-2021 MagicStack Inc. <http://magic.io>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



---

### 版權宣告

---

Python 和這份圖明文件的版權：

Copyright © 2001-2023 Python Software Foundation 保留一切權利。

Copyright © 2000 BeOpen.com 保留一切權利。

Copyright © 1995-2000 Corporation for National Research Initiatives 保留一切權利。

Copyright © 1991-1995 Stichting Mathematisch Centrum 保留一切權利。

---

完整的授權條款資訊請參見[沿革與授權](#)。



## 非依字母順序

..., [11](#)  
 2to3, [11](#)  
 >>>, [11](#)  
 \_\_future\_\_, [16](#)  
 \_\_slots\_\_, [24](#)

## A

abstract base class (抽象基底類 [F](#)), [11](#)  
 annotation ([F](#)釋), [11](#)  
 argument (引數), [12](#)  
 asynchronous context manager (非同步情境管理器), [12](#)  
 asynchronous generator iterator (非同步 [F](#)生器 [F](#)代器), [12](#)  
 asynchronous generator (非同步 [F](#)生器), [12](#)  
 asynchronous iterable (非同步可 [F](#)代物件), [12](#)  
 asynchronous iterator (非同步 [F](#)代器), [12](#)  
 attribute (屬性), [12](#)  
 awaitable (可等待物件), [13](#)

## B

BDFL, [13](#)  
 binary file (二進制檔案), [13](#)  
 borrowed reference (借用參照), [13](#)  
 bytecode (位元組碼), [13](#)  
 bytes-like object (類位元組串物件), [13](#)

## C

callable (可呼叫物件), [13](#)  
 callback (回呼), [13](#)  
 C-contiguous (C 連續的), [14](#)  
 class variable (類 [F](#)變數), [14](#)  
 class (類 [F](#)), [13](#)  
 complex number ([F](#)數), [14](#)  
 context manager (情境管理器), [14](#)  
 context variable (情境變數), [14](#)  
 contiguous (連續的), [14](#)

coroutine function (協程函式), [14](#)  
 coroutine (協程), [14](#)  
 CPython, [14](#)

## D

decorator (裝飾器), [14](#)  
 descriptor (描述器), [14](#)  
 dictionary comprehension (字典綜合運算), [15](#)  
 dictionary view (字典檢視), [15](#)  
 dictionary (字典), [15](#)  
 docstring ([F](#)明字串), [15](#)  
 duck-typing (鴨子型 [F](#)), [15](#)

## E

EAFP, [15](#)  
 expression (運算式), [15](#)  
 extension module (擴充模組), [15](#)

## F

f-string (f 字串), [15](#)  
 file object (檔案物件), [15](#)  
 file-like object (類檔案物件), [16](#)  
 filesystem encoding and error handler (檔案系統編碼和錯誤處理函式), [16](#)  
 finder (尋檢器), [16](#)  
 floor division (向下取整除法), [16](#)  
 Fortran contiguous (Fortran 連續的), [14](#)  
 function annotation (函式 [F](#)釋), [16](#)  
 function (函式), [16](#)

## G

garbage collection (垃圾回收), [16](#)  
 generator expression ([F](#)生器運算式), [17](#)  
 generator iterator ([F](#)生器 [F](#)代器), [17](#)  
 generator ([F](#)生器), [17](#)  
 generic function (泛型函式), [17](#)  
 generic type (泛型型 [F](#)), [17](#)  
 GIL, [17](#)

global interpreter lock (全域直譯器鎖), 17

## H

hash-based pyc (雜 架構的 pyc), 17

hashable (可雜 的), 17

## I

IDLE, 18

immutable (不可變物件), 18

import path (引入路徑), 18

importer (引入器), 18

importing (引入), 18

interactive (互動的), 18

interpreted (直譯的), 18

interpreter shutdown (直譯器關閉), 18

iterable (可 代物件), 18

iterator ( 代器), 19

## K

key function (鍵函式), 19

keyword argument (關鍵字引數), 19

## L

lambda, 19

LBYL, 19

list comprehension (串列綜合運算), 19

list (串列), 19

loader (載入器), 19

locale encoding (區域編碼), 20

## M

magic

method (方法), 20

magic method (魔術方法), 20

mapping (對映), 20

meta path finder (元路徑尋檢器), 20

metaclass (元類 ), 20

method resolution order (方法解析順序), 20

method (方法), 20

magic, 20

special, 24

module spec (模組規格), 20

module (模組), 20

MRO, 20

mutable (可變物件), 20

## N

named tuple (附名元組), 20

namespace package (命名空間套件), 21

namespace (命名空間), 21

nested scope (巢狀作用域), 21

new-style class (新式類 ), 21

## O

object (物件), 21

## P

package (套件), 21

parameter (參數), 21

path based finder (基於路徑的尋檢器), 22

path entry finder (路徑項目尋檢器), 22

path entry hook (路徑項目 ), 22

path entry (路徑項目), 22

path-like object (類路徑物件), 22

PEP, 22

portion (部分), 22

positional argument (位置引數), 23

provisional API (暫行 API), 23

provisional package (暫行套件), 23

Python 3000, 23

Python Enhancement Proposals

PEP 1, 22

PEP 238, 16

PEP 278, 25

PEP 302, 16, 19

PEP 343, 14

PEP 362, 12, 22

PEP 411, 23

PEP 420, 16, 21, 22

PEP 443, 17

PEP 451, 16

PEP 483, 17

PEP 484, 11, 16, 17, 25, 26

PEP 492, 1214

PEP 498, 15

PEP 519, 22

PEP 525, 12

PEP 526, 11, 26

PEP 585, 17

PEP 3116, 25

PEP 3155, 23

Pythonic (Python 風格的), 23

## Q

qualified name (限定名稱), 23

## R

reference count (參照計數), 24

regular package (正規套件), 24

## S

sequence (序列), 24

set comprehension (集合綜合運算), 24

single dispatch (單一調度), 24

slice (切片), 24

special



method (方法), 24  
special method (特殊方法), 24  
statement (陳述式), 24  
static type checker, 24  
strong reference (F 參照), 24

## T

text encoding (文字編碼), 25  
text file (文字檔案), 25  
triple-quoted string (三引號 F 字串), 25  
type alias (型 F 名), 25  
type hint (型 F 提示), 25  
type (型 F), 25

## U

universal newlines (通用 F 行字元), 25

## V

variable annotation (變數 F 釋), 26  
virtual environment (F 擬環境), 26  
virtual machine (F 擬機器), 26

## Z

Zen of Python (Python 之 F), 26