

Installing Python Modules

發  3.14.0rc3

Guido van Rossum and the Python development team

10 月 01, 2025

Python Software Foundation
Email: docs@python.org

1	關鍵術語	3
2	基本用法	5
3	我該如何... ?	7
3.1	... 在 Python 3.4 之前的 Python 版本中安裝 pip?	7
3.2	... 只讓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 版本並使用它們?	7
4	常見的安裝問題	9
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
A	術語表	11
B	關於這份「明文件	29
B.1	Python 文件的貢獻者們	29
C	沿革與授權	31
C.1	軟體沿革	31
C.2	關於存取或以其他方式使用 Python 的合約條款	32
C.2.1	PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2	32
C.2.2	BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0	33
C.2.3	CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1	33
C.2.4	CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2	34
C.2.5	ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION	35
C.3	被收「軟體的授權與致謝	35
C.3.1	Mersenne Twister	35
C.3.2	Sockets	36
C.3.3	非同步 socket 服務	36
C.3.4	Cookie 管理	37
C.3.5	執行追「	37
C.3.6	UUencode 與 UUdecode 函式	38
C.3.7	XML 遠端程序呼叫	38
C.3.8	test_epoll	39
C.3.9	Select kqueue	39
C.3.10	SipHash24	40
C.3.11	strtod 與 dtoa	40
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	mimalloc	47
C.3.20	asyncio	47
C.3.21	Global Unbounded Sequences (GUS)	47
C.3.22	Zstandard bindings	48
D	版權宣告	49
	索引	51

電子郵件

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決方案，更可以在公用社群中在地貢獻他們自己的解決方案。

這份指南涵蓋了上述過程中的安裝部分。如果是要建立及分享你自己的 Python 專案，請參考 [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，可以讓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](#) 平台上維護各種工具、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    # 明確版本  
python -m pip install "SomePackage>=1.0.4" # 最小版本
```

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。

虛擬環境的建立是使用 `venv` 模組來完成。要在一個已用的虛擬環境中安裝套件，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 版本並使用它們？

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 # 預設 Python 2
python2.7 -m pip install SomePackage # 指定 Python 2.7
python3 -m pip install SomePackage # 預設 Python 3
python3.4 -m pip install SomePackage # 指定 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 啟動指令 py 結合 -m 開關參數 (switch):

```
py -2 -m pip install SomePackage # 預設 Python 2
py -2.7 -m pip install SomePackage # 指定 Python 2.7
py -3 -m pip install SomePackage # 預設 Python 3
py -3.4 -m pip install SomePackage # 指定 Python 3.4
```

常見的安裝問題

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

>>>

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) 內部, 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。
- The three dots form of the Ellipsis object.

abstract base class (抽象基底類)

抽象基底類 (又稱 ABC) 提供了一種定義介面的方法, 作 *duck-typing* (鴨子型) 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 用擬的 subclass (子類), 它們不繼承自另一個 class (類), 但仍可被 `isinstance()` 及 `issubclass()` 辨識; 請參 `abc` 模組的說明文件。Python 有許多建的 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。

annotate function (釋函式)

A function that can be called to retrieve the *annotations* of an object. This function is accessible as the `__annotate__` attribute of functions, classes, and modules. Annotate functions are a subset of *evaluate functions*.

annotation (釋)

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

在 runtime 的區域變數釋無法被存取, 但全域變數、類屬性和函式的釋, 分能對模組、類和函式呼叫 `annotationlib.get_annotations()` 來取得。

請參 *variable annotation*、*function annotation*、**PEP 484**、**PEP 526** 和 **PEP 649**, 這些章節皆有此功能的說明。關於釋的最佳實踐方法也請參 `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* (可迭代物件) 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* (非同步生成器迭代器) 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迭代器 (*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 (非同步生成器迭代器)

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* (非同步迭代器), 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)。當非同步生成器迭代器以另一個被 `__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 (非同步可迭代物件)

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* (非同步迭代器)。由 [PEP 492](#) 引入。

asynchronous iterator (非同步迭代器)

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` 會解析非同步迭代器的 `__anext__()` method 所回傳的可等待物件, 直到它引發 `StopAsyncIteration` 例外。由 [PEP 492](#) 引入。

attached thread state

A *thread state* that is active for the current OS thread.

When a *thread state* is attached, the OS thread has access to the full Python C API and can safely invoke the bytecode interpreter.

Unless a function explicitly notes otherwise, attempting to call the C API without an attached thread state will result in a fatal error or undefined behavior. A thread state can be attached and detached explicitly by the user through the C API, or implicitly by the runtime, including during blocking C calls and by the bytecode interpreter in between calls.

On most builds of Python, having an attached thread state implies that the caller holds the *GIL* for the current interpreter, so only one OS thread can have an attached thread state at a given moment. In *free-threaded* builds of Python, threads can concurrently hold an attached thread state, allowing for true parallelism of the bytecode interpreter.

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` 所定義之識別符 (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 (二進位檔案)

一個能讀取和寫入 *bytes-like objects* (類位元組串物件) 的 *file object* (檔案物件)。二進位檔案的例子有：以二進位模式 ('rb'、'wb' 或 'rb+') 開的檔案、`sys.stdin.buffer`、`sys.stdout.buffer`，以及 `io.BytesIO` 和 `gzip.GzipFile` 實例。

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)

在 Python 的 C API 中，借用參照是一個對物件的參照，其中使用該物件的程式碼不擁有這個參照。如果該物件被銷毀，它會成一個迷途指標 (dangling pointer)。例如，一次垃圾回收 (garbage collection) 可以移除對物件的最後一個 *strong reference* (參照)，而將該物件銷毀。

對 *borrowed reference* 呼叫 `Py_INCREF()` 以將它原地 (in-place) 轉成 *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位資料的各種運算；這些運算包括壓縮、儲存至二進位檔案和透過 `socket` (插座) 發送。

有些運算需要二進位資料是可變的。明文文件通常會將這些物件稱「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位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 (可以不用從原始碼重新編譯位元組碼)。這種「中間語言 (intermediate language)」據是運行在一個 *virtual machine* (擬機器) 上，該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 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。

closure variable (閉包變數)

從外部作用域中定義且從巢狀作用域參照的自由變數，不是於 runtime 從全域或建立命名空間解析。可以使用 `nonlocal` 關鍵字明確定義以允許寫入存取，或者如果僅需讀取變數則隱式定義即可。

例如在下面程式碼中的 `inner` 函式中，`x` 和 `print` 都是自由變數，但只有 `x` 是閉包變數：

```
def outer():
    x = 0
    def inner():
        nonlocal x
        x += 1
        print(x)
    return inner
```

由於 `codeobject.co_freevars` 屬性（儘管名稱如此，但它僅包含閉包變數的名稱，而不是列出所有參照的自由變數），當預期含義是特指閉包變數時，有時候甚至也會使用更通用的自由變數一詞。

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為一個實部和一個虛部之和。複數就是虛數單位（-1 的平方根）的實數倍，此單位通常在數學中被寫為 i ，在工程學中被寫為 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`。若要將 `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 (情境)

This term has different meanings depending on where and how it is used. Some common meanings:

- The temporary state or environment established by a *context manager* via a `with` statement.
- The collection of keyvalue bindings associated with a particular `contextvars.Context` object and accessed via `ContextVar` objects. Also see *context variable*.
- 一個 `contextvars.Context` 物件。另請參 *current context*。

context management protocol (情境管理協定)

由 `with` 陳述式所呼叫的 `__enter__()` 和 `__exit__()` 方法。另請參 [PEP 343](#)。

context manager (情境管理器)

An object which implements the *context management protocol* and controls the environment seen in a `with` statement. See [PEP 343](#).

context variable (情境變數)

A variable whose value depends on which context is the *current context*. Values are accessed via `contextvars.ContextVar` objects. Context variables are primarily used to isolate state between concurrent asynchronous tasks.

contiguous (連續的)

如果一個緩衝區是 *C-contiguous* 或是 *Fortran contiguous*，則它會確切地被視為是連續的。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瀏覽各個項目時，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 [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，可能包含 `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 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。

current context

The *context* (`contextvars.Context` object) that is currently used by `ContextVar` objects to access (get or set) the values of *context variables*. Each thread has its own current context. Frameworks for executing asynchronous tasks (see `asyncio`) associate each task with a context which becomes the current context whenever the task starts or resumes execution.

cyclic isolate

A subgroup of one or more objects that reference each other in a reference cycle, but are not referenced by objects outside the group. The goal of the *cyclic garbage collector* is to identify these groups and break the reference cycles so that the memory can be reclaimed.

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 (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)

任何定義了 `__get__()`、`__set__()` 或 `__delete__()` method 的物件。當一個 class 屬性是一個描述器時，它的特殊連結行會在屬性查找時被觸發。通常，使用 `a.b` 來取得、設定或除某個屬性時，會在 `a` 的 class 字典中查找名稱 `b` 的物件，但如果 `b` 是一個描述器，則相對應的描述器 method 會被呼叫。對描述器的理解是深入理解 Python 的關鍵，因為它們是許多功能的基礎，這些功能包括函式、method、屬性 (property)、class method、態 method，以及對 super class (父類) 的參照。

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)

一個關聯陣列 (associative array)，其中任意的鍵會被對映到值。鍵可以是任何帶有 `__hash__()` 和 `__eq__()` method 的物件。在 Perl 中被稱雜 (hash)。

dictionary comprehension (字典綜合運算)

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，將處理結果以一個字典回傳。`results = {n: n ** 2 for n in range(10)}` 會生一個字典，它包含了鍵 `n` 對映到值 `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。若要制將字典檢視轉完整的 list (串列)，須使用 `list(dictview)`。請參閱 dict-views。

docstring (明字串)

一個在 class、函式或模組中，作第一個運算式出現的字串文本。雖然它在套件執行時會被忽略，但它會被編譯器辨識，被放入所屬 class、函式或模組的 `__doc__` 屬性中。由於明字串可以透過省 (introspection) 來覽，因此它是物件的明文件存放的標準位置。

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。（「如果它看起來像一鴨子而且叫起來像一鴨子，那它一定是一鴨子。」）因調介面而非特定型，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。

dunder (雙底)

一個非正式的縮寫，代表「雙底 (double underscore)」，當談論到特殊方法時使用。例如，`__init__` 通常被叫做“dunder init”。

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，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。

evaluate function (求值函式)

A function that can be called to evaluate a lazily evaluated attribute of an object, such as the value of type aliases created with the `type` statement.

expression (運算式)

一段可以被評估求值的語法。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)

f-strings (f 字串)

以 `f` 或 `F` 前綴的字串文本通常被稱「f 字串」，它是格式化的字串文本的縮寫。另請參 [PEP 498](#)。

file object (檔案物件)

一個讓使用者透過檔案導向 (file-oriented) API (如 `read()` 或 `write()` 等 `method`) 來操作底層資源的物件。根據檔案物件被建立的方式，它能協調對真實磁碟檔案或是其他類型的儲存器或通訊裝置 (例如標準輸入 / 輸出、記憶體緩衝區、`socket` (插座)、管 (pipe) 等) 的存取。檔案物件也被稱類檔案物件 (*file-like object*) 或串流 (*stream*)。

實際上，有三種檔案物件：原始的二進位檔案、緩衝的二進位檔案和文字檔案。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 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)。

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 [finders-and-loaders](#) 和 `importlib` 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 `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 `-2.75` 被向下無條件舍去。請參 [PEP 238](#)。

free threading (自由執行緒)

一種執行緒模型，多個執行緒可以在同一直譯器中同時運行 Python 位元組碼。這與全域直譯器鎖形成對比，後者一次只允許一個執行緒執行 Python 位元組碼。請參 [PEP 703](#)。

free variable (自由變數)

Formally, as defined in the language execution model, a free variable is any variable used in a namespace which is not a local variable in that namespace. See [closure variable](#) for an example. Pragmatically, due to the name of the `codeobject.co_freevars` attribute, the term is also sometimes used as a synonym for [closure variable](#).

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，來編譯目前的模組。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，能生一系列的 `yield` 值，這些值可用於 `for` 圈，或是以 `next()` 函式，每次檢索其中的一個值。

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，記住執行狀態 (包括區域變數及擱置中的 `try` 陳述式)。當生成器代器回復時，它會從停止的地方繼續執行 (與那些每次呼叫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。該組合運算式會外層函式生多個值：

```
>>> sum(i*i for i in range(10))      # 平方之和 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 (包括關鍵的建型，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。

從 Python 3.13 開始可以使用 `--disable-gil` 建置設定來停用 GIL。使用此選項建立 Python 後，必須使用 `-X gil=0` 來執行程式碼，或者設定 `PYTHON_GIL=0` 環境變數後再執行程式碼。此功能可以提高多執行緒應用程式的效能，使多核心 CPU 的高效使用變得更加容易。有關更多詳細資訊，請參 [PEP 703](#)。

In prior versions of Python's C API, a function might declare that it requires the GIL to be held in order to use it. This refers to having an *attached thread state*.

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)

如果一個物件有一個雜值，該值在其生命期中永不改變 (它需要一個 `__hash__()` method)，且可與其他物件互相比較 (它需要一個 `__eq__()` method)，那它就是一個可雜物件。比較結果相等的多個可雜物件，它們必須擁有相同的雜值。

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 不可變建物件都是可雜的；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。

immortal (不滅)

不滅物件 (Immortal objects) 是 [PEP 683](#) 引入的 *CPython* 實作細節。

如果一個物件是不滅的，它的參照計數永遠不會被修改，因此在直譯器運行時它永遠不會被釋放。例如，`True` 和 `None` 在 *CPython* 中是不滅的。

Immutable objects can be identified via `sys._is_immortal()`, or via `PyUnstable_IsImmortal()` in the C API.

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。它們在需要固定值的地方，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組和包的非常大的方法 (請記住 `help(x)`)。更多互動式模式相關資訊請見 `tut-interac`。

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，因為有位元組碼 (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，因為它所依賴的資源可能不再有了 (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 (可迭代物件)

一種能一次回傳一個其中成員的物件。可迭代物件的例子包括所有的序列型 (像是 `list`、`str` 和 `tuple`) 和某些非序列型，像是 `dict`、*檔案物件*，以及你所定義的任何 `class` 物件，只要那些 `class` 有實作 *sequence* (序列) 語意的 `__iter__()` 或是 `__getitem__()` *method*，該物件就是可迭代物件。

可迭代物件可用於 `for` 圈和許多其他需要一個序列的地方 (`zip()`、`map()`...)。當一個可迭代物件作引數被傳遞給建構函式 `iter()` 時，它會該物件回傳一個 *迭代器*。此 *迭代器* 適用於針對一組值進行一遍 (*one pass*) 運算。使用 *迭代器* 時，通常不一定要呼叫 `iter()` 或自行處理 *迭代器* 物件。`for` 陳述式會自動地你處理這些事，它會建立一個暫時性的未命名變數，用於在圈期間保有該 *迭代器*。另請參 *iterator* (*迭代器*)、*sequence* (序列) 和 *generator* (*生成器*)。

iterator (迭代器)

一個表示資料流的物件。重地呼叫 *迭代器* 的 `__next__()` *method* (或是將它傳遞給建構函式 `next()`) 會依序回傳資料流中的各項目。當不再有資料時，則會引發 `StopIteration` 例外。此時，該 *迭代器* 物件已被用盡，而任何對其 `__next__()` *method* 的進一步呼叫，都只會再次引發 `StopIteration`。*迭代器* 必須有一個 `__iter__()` *method*，它會回傳 *迭代器* 物件本身，所以每個 *迭代器* 也都是可迭代物件，且可以用於大多數適用其他可迭代物件的場合。一個明顯的例外，是嘗試多遍 *迭代* (*multiple iteration passes*) 的程式碼。一個容器物件 (像是 `list`) 在每次你將它傳遞給 `iter()` 函式或在 `for` 圈中使用它時，都會生一個全新的 *迭代器*。使用 *迭代器* 嘗試此事 (多遍 *迭代*) 時，只會回傳在前一遍 *迭代* 中被用過的、同一個已被用盡的 *迭代器* 物件，使其看起來就像一個空的容器。

在 `typeiter` 文中可以找到更多資訊。

CPython 不是始終如一地都會檢查「代器有定義 `__iter__()`」這個規定。另請注意，`free-threading`（自由執行緒）CPython 不保證代器操作的執行緒安全。

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，請參如何排序。

keyword argument（關鍵字引數）

請參 [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。

lexical analyzer（詞法分析器）

`tokenizer` 的正式名稱；請參 [token](#)。

list（串列）

一個 Python 建的 *sequence*（序列）。管它的名字是 `list`，它其實更類似其他語言中的一個陣列 (array) 而較不像一個鏈結串列 (linked list)，因存取元素的時間複雜度是 $O(1)$ 。

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]` 會生一個字串 `list`，其中包含 0 到 255 範圍，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 `exec_module()` 和 `create_module()` 方法以實作 `Loader` 介面。載入器通常是被 *finder*（尋檢器）回傳。更多細節請參：

- `finders-and-loaders`
- `importlib.abc.Loader`
- **PEP 302**

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 碼頁 (code page，例如 "cp1252")。

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

`locale.getencoding()` 可以用來取得區域編碼。

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 `Python` 自 2.3 版直譯器所使用的演算法細節，請參 `python_2.3_mro`。

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。

另請參 `module-specs`。

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。

有些建型是 `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]           # 以索引存取
1024
>>> sys.float_info.max_exp      # 以欄位名稱存取
1024
>>> isinstance(sys.float_info, tuple) # 屬於 tuple 型
True
```

有些 `named tuple` 是建型 (如上例)。或者，一個 `named tuple` 也可以從一個正規的 `class` 定義來建立，只要該 `class` 是繼承自 `tuple`，且定義了附名欄位 (`named field`) 即可。這類的 `class` 可以手工編

寫、可以繼承自 `typing.NamedTuple` 來建立，也可以使用工廠函式 (factory function) `collections.namedtuple()` 來建立。後者技術也增加了一些額外的 `method`，這些 `method` 可能是在手寫或建立的 `named tuple` 中，無法找到的。

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 *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。

命名空間套件允許數個可獨立安裝的套件擁有一個共同的父套件。除此之外，建議使用 *regular package*。

更多資訊，請參 [PEP 420](#) 和 `reference-namespace-package`。

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)

一個舊名，它是指現在所有的 `class` 物件所使用的 `class` 風格。在早期的 Python 版本中，只有新式 `class` 才能使用 Python 較新的、多樣的功能，像是 `__slots__`、描述器 (descriptor)、屬性 (property)、`__getattr__()`、`class method` (類方法) 和 `static method` (態方法)。

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)。

optimized scope (最佳化作用域)

A scope where target local variable names are reliably known to the compiler when the code is compiled, allowing optimization of read and write access to these names. The local namespaces for functions, generators, coroutines, comprehensions, and generator expressions are optimized in this fashion. Note: most interpreter optimizations are applied to all scopes, only those relying on a known set of local and nonlocal variable names are restricted to optimized scopes.

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 (路徑項目勾)

在 *sys.path_hooks* 列表中的一個可呼叫物件 (callable)，若它知道如何在一個特定的 *path entry* 中尋找模組，則會回傳一個 *path entry finder* (路徑項目尋檢器)。

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 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 的設計策的記，這些過程的主要機制。PEP 的作者要負責在社群建立共識反對意見。

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，只要它們被標示暫行的，理論上不會有重大的變更，但如果核心開發人員認

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 [PEP 411](#) 了解更多細節。

provisional package（暫行套件）

請參 [provisional API](#)（暫行 API）。

Python 3000

Python 3.x 系列版本的稱（很久以前創造的，當時第 3 版的發布是在遠的未來。）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，對一個可代物件的所有元素進行圈。許多其他語言有這種類型的架構，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 (*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*)。有些物件是「不滅的 (*immortal*)」擁有不會被改變的參照計數，也因此永遠不會被解除配置。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。

在 [CPython](#) 中，參照計數不被視為穩定或明確定義的值；對物件的參照數量，以及該數量如何受到 Python 程式碼的影響，在不同版本之間可能會有所不同。

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)。

REPL

「read-eval-print [圈](#) (read-eval-print loop)」的縮寫，是 [互動式直譯器 shell](#) 的另一個名稱。

__slots__

在 class [區](#) 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[區](#)。

sequence (序列)

一個 [iterable](#) (可 [區](#) 代物件)，它透過 `__getitem__()` special method (特殊方法)，使用整數索引來支援高效率的元素存取，[區](#) 定義了一個 `__len__()` method 來回傳該序列的長度。一些 [區](#) 建序列型 [區](#) 包括 list、str、tuple 和 bytes。請注意，雖然 dict 也支援 `__getitem__()` 和 `__len__()`，但它被視 [區](#) 對映 (mapping) 而不是序列，因 [區](#) 其查找方式是使用任意的 [hashable](#) 鍵，而不是整數。

抽象基底類 [區](#) (abstract base class) `collections.abc.Sequence` 定義了一個更加豐富的介面，[區](#) 不僅止於 `__getitem__()` 和 `__len__()`，還增加了 `count()`、`index()`、`__contains__()` 和 `__reversed__()`。實作此擴充介面的型 [區](#)，可以使用 `register()` 被明確地 [區](#) [區](#)。更多關於序列方法的文件，請見常見序列操作。

set comprehension (集合綜合運算)

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

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 [區](#) 部，會使用 slice 物件。

soft deprecated (軟性 [區](#) 用)

被軟性 [區](#) 用的 API 代表不應再用於新程式碼中，但在現有程式碼中繼續使用它仍會是安全的。API 仍會以文件記 [區](#) [區](#) 會被測試，但不會被繼續改進。

與正常 [區](#) 用不同，軟性 [區](#) 用 [區](#) 有 [區](#) 除 API 的規劃，也不會發出警告。

請參 [區](#) PEP 387: 軟性 [區](#) 用。

special method (特殊方法)

一種會被 Python 自動呼叫的 method，用於對某種型 [區](#) 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 [區](#)。Special method 在 `specialnames` 中有詳細 [區](#) 明。

標準函式庫

包含 [套件](#)、[模組](#) 和 [擴充模組](#) 的集合，它們是作 [區](#) 官方 Python 直譯器套件的一部分來發行。該集合的成員可能會因平台、可用的系統函式庫或其他條件而有所不同。相關文件可以在 `library-index` 中找到。

請參 [區](#) `sys.stdlib_module_names` 以取得所有可能的標準函式庫模組名稱的列表。

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 ([區](#) 態型 [區](#) 檢查器)

會讀取 Python 程式碼 [區](#) 分析的外部工具，能 [區](#) 找出錯誤，像是使用了不正確的型 [區](#)。另請參 [區](#) 型 [區](#) 提示 ([type hints](#)) 以及 `typing` 模組。

stdlib (標準函式庫)

[standard library](#) 的縮寫。

strong reference ([區](#) 參照)

在 Python 的 C API 中，[區](#) 參照是對物件的參照，該物件 [區](#) 持有該參照的程式碼所擁有。建立參照時透過呼叫 `Py_INCREF()` 來獲得 [區](#) 參照、[區](#) 除參照時透過 `Py_DECREF()` 釋放 [區](#) 參照。

`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)。

t-string (t 字串)

t-strings (t 字串)

以 `t` 或 `T` 前綴的字串文本通常被稱「t 字串」，它是模板化的字串文本的縮寫。

text encoding (文字編碼)

Python 中的字串是一個 Unicode 編碼位置 (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* (二進位檔案)，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。

thread state

The information used by the *CPython* runtime to run in an OS thread. For example, this includes the current exception, if any, and the state of the bytecode interpreter.

Each thread state is bound to a single OS thread, but threads may have many thread states available. At most, one of them may be *attached* at once.

An *attached thread state* is required to call most of Python's C API, unless a function explicitly documents otherwise. The bytecode interpreter only runs under an attached thread state.

Each thread state belongs to a single interpreter, but each interpreter may have many thread states, including multiple for the same OS thread. Thread states from multiple interpreters may be bound to the same thread, but only one can be *attached* in that thread at any given moment.

See Thread State and the Global Interpreter Lock for more information.

token

原始碼的小單位，由詞法分析器 (也稱 *tokenizer*) 生。名稱、數字、字串、運算子、行符號等都以 token 表示。

`tokenize` 模組公開了 Python 的詞法分析器。`token` 模組包含各種 token 類型的資訊。

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。它們讓你可以在字串中包含未跳 (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* ([解釋](#))，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。

型提示是選擇性的，而不是被 Python 制的，但它們對 *態型檢查器 (static type checkers)* 很有用，能協助 IDE 完成程式碼的補全 (completion) 和重構 (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 (變數解釋)

一個變數或 class 屬性的 *annotation* ([解釋](#))。

解釋變數或 class 屬性時，賦值是選擇性的：

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

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

```
count: int = 0
```

變數解釋的語法在 [annassign](#) 章節有詳細的解釋。

請參 [function annotation](#) (函式解釋)、[PEP 484](#) 和 [PEP 526](#)，皆有此功能的描述。關於解釋的最佳實踐方法，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。

walrus operator (海象運算子)

A light-hearted way to refer to the assignment expression operator `:` because it looks a bit like a walrus if you turn your head.

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。

關於這份📖明文件

Python 📖明文件是透過使用 [Sphinx](#)（一個原📖 Python 而生的文件📖生器、目前是以獨立專案形式來維護）將使用 [reStructuredText](#) 撰寫的原始檔轉📖而成。

如同 Python 自身，透過自願者的努力下📖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，📖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📖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📖 Python 這門語言、Python 標準函式庫和 Python 📖明文件貢獻過。Python 所發📖的原始碼中含有部份貢獻者的清單，請見 [Misc/ACKS](#)。

正因📖 Python 社群的撰寫與貢獻才有這份這📖棒的📖明文件 -- 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 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，後來成 Zope Corporation。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 過去是 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。

發版本	源自	年份	擁有者	GPL 相容性 ? (1)
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	是 (2)
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

i 備

- (1) GPL 相容不表示我們是在 GPL 下發 Python。不像 GPL，所有的 Python 授權都可以讓你發修改後的版本，但不一定要使你的變更成開源。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。

- (2) 根據 Richard Stallman 的發法，1.6.1 不是 GPL 相容的，因發其授權有一個法律選擇條款。然而根據 CNRI 的發法，Stallman 的律師告訴 CNRI 的律師，1.6.1 與 GPL 「不相容」。

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 Python 軟體基金會授權第二版 (Python Software Foundation License Version 2)。

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 授權第二版以及 *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 PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") 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 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001 Python Software Foundation; All Rights Reserved" are retained in Python 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 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.
4. PSF is making Python 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 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, 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, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.2 BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

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.
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 CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

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

(繼續下一頁)

(繼續上一頁)

- 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.
 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 CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

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

(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.2.5 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION

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 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 版本中所收 的第三方軟體。

C.3.1 Mersenne Twister

random 模組底下的 _random C 擴充程式包含了以 <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> 的下載 容 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

(繼續下一頁)

(繼續上一頁)

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

socket 模組使用 `getaddrinfo()` 和 `getnameinfo()` 函式，它們在 WIDE 專案 (<https://www.wide.ad.jp/>) 中，於不同的原始檔案中被編碼：

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

`test.support.asyncchat` 和 `test.support.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

(繼續下一頁)

(繼續上一頁)

```
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,
```

(繼續下一頁)

(繼續上一頁)

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 編解碼器包含以下聲明：

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 MERCHANTABILITY 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

test.test_epoll 模組包含以下聲明：

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 kqueue

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 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/siphhash24/little)
  djb (supercop/crypto_auth/siphhash24/little2)
  Jean-Philippe Aumasson (https://131002.net/siphhash/siphhash24.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。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.
 */
```

C.3.12 OpenSSL

如果 OpenSSL 函式庫可被作業系統使用，則 hashlib、posix、ssl 模組會使用它來提升效能。此外，因 Windows 和 macOS 的 Python 安裝程式可能包含 OpenSSL 函式庫的副本，所以我們也在此收 OpenSSL 授權的副本。對於 OpenSSL 3.0 版本以及由此衍生的更新版本則適用 Apache 許可證 v2：

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

(繼續下一頁)

(繼續上一頁)

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, 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.

(繼續下一頁)

(繼續上一頁)

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, 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

(繼續下一頁)

(繼續上一頁)

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 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

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

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

(繼續下一頁)

(繼續上一頁)

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` 模組底下 `_ctypes` 擴充程式時設定 `--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

(繼續下一頁)

(繼續上一頁)

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 模組底下 `_decimal` C 擴充程式時設定 `--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.

(繼續下一頁)

(繼續上一頁)

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 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 mimalloc

MIT 授權：

```
Copyright (c) 2018-2021 Microsoft Corporation, Daan Leijen

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.3.20 asyncio

asyncio 模組的部分內容是從 `uvloop 0.16` 中收過來，其基於 MIT 授權來發：

```
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.
```

C.3.21 Global Unbounded Sequences (GUS)

The file `Python/qsbr.c` is adapted from FreeBSD's "Global Unbounded Sequences" safe memory reclamation scheme in `subr_smr.c`. The file is distributed under the 2-Clause BSD License:

```
Copyright (c) 2019,2020 Jeffrey Roberson <jeff@FreeBSD.org>

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 unmodified, this list of conditions, and the following
```

(繼續下一頁)

(繼續上一頁)

```
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 "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 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.3.22 Zstandard bindings

Zstandard bindings in `Modules/_zstd` and `Lib/compression/zstd` are based on code from the [pyzstd library](#), copyright Ma Lin and contributors. The `pyzstd` code is distributed under the 3-Clause BSD License:

```
Copyright (c) 2020-present, Ma Lin and contributors.
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 copyright holder 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 HOLDER 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.
```

版權宣告

Python 和這份圖明文件的版權：

Copyright © 2001 Python 軟體基金會。保留所有權利。

Copyright © 2000 BeOpen.com 保留所有權利。

Copyright © 1995-2000 Corporation for National Research Initiatives 保留所有權利。

Copyright © 1991-1995 Stichting Mathematisch Centrum 保留所有權利。

完整的授權條款資訊請參見[沿革與授權](#)。

非依字母順序

..., [11](#)
 節號字面值, [11](#)
 >>>, [11](#)
 __future__, [17](#)
 __slots__, [25](#)
 環境變數
 PYTHON_GIL, [18](#)

A

abstract base class (抽象基底類 [F](#)), [11](#)
 annotate function ([F](#)釋函式), [11](#)
 annotation ([F](#)釋), [11](#)
 argument (引數), [11](#)
 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](#)
 attached thread state, [12](#)
 attribute (屬性), [13](#)
 awaitable (可等待物件), [13](#)

B

BDFL, [13](#)
 binary file (二進位檔案), [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](#)
 closure variable (閉包變數), [14](#)
 complex number ([F](#)數), [14](#)
 context management protocol (情境管理協定), [14](#)
 context manager (情境管理器), [14](#)

context variable (情境變數), [14](#)
 context (情境), [14](#)
 contiguous (連續的), [14](#)
 coroutine function (協程函式), [14](#)
 coroutine (協程), [14](#)
 CPython, [15](#)
 current context, [15](#)
 cyclic isolate, [15](#)

D

decorator (裝飾器), [15](#)
 descriptor (描述器), [15](#)
 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](#)
 dunder (雙底 [F](#)), [16](#)

E

EAFP, [16](#)
 evaluate function (求值函式), [16](#)
 expression (運算式), [16](#)
 extension module (擴充模組), [16](#)

F

f-strings (f 字串), [16](#)
 f-string (f 字串), [16](#)
 file object (檔案物件), [16](#)
 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), [17](#)
 Fortran contiguous (Fortran 連續的), [14](#)
 free threading (自由執行緒), [17](#)
 free variable (自由變數), [17](#)
 function annotation (函式 [F](#)釋), [17](#)
 function (函式), [17](#)

G

garbage collection (垃圾回收), [17](#)
 generator expression ([F](#)生器運算式), [17](#), [18](#)

generator iterator (生成器代器), 17
 generator (生成器), 17
 generic function (泛型函式), 18
 generic type (泛型型), 18
 GIL, 18
 global interpreter lock (全域直譯器鎖), 18

H

hash-based pyc (雜架構的 pyc), 18
 hashable (可雜的), 18

I

IDLE, 18
 immortal (不滅), 18
 immutable (不可變物件), 19
 import path (引入路徑), 19
 importer (引入器), 19
 importing (引入), 19
 interactive (互動的), 19
 interpreted (直譯的), 19
 interpreter shutdown (直譯器關閉), 19
 iterable (可代物件), 19
 iterator (代器), 19

K

key function (鍵函式), 20
 keyword argument (關鍵字引數), 20

L

lambda, 20
 LBYL, 20
 lexical analyzer (詞法分析器), 20
 list comprehension (串列綜合運算), 20
 list (串列), 20
 loader (載入器), 20
 locale encoding (區域編碼), 20

M

magic
 method (方法), 21
 magic method (魔術方法), 21
 mapping (對映), 21
 meta path finder (元路徑尋檢器), 21
 metaclass (元類), 21
 method resolution order (方法解析順序), 21
 method (方法), 21
 magic, 21
 special, 25
 module spec (模組規格), 21
 module (模組), 21
 MRO, 21
 mutable (可變物件), 21

N

named tuple (附名元組), 21
 namespace package (命名空間套件), 22
 namespace (命名空間), 22

nested scope (巢狀作用域), 22
 new-style class (新式類), 22

O

object (物件), 22
 optimized scope (最佳化作用域), 22

P

package (套件), 22
 parameter (參數), 22
 path based finder (基於路徑的尋檢器), 23
 path entry finder (路徑項目尋檢器), 23
 path entry hook (路徑項目), 23
 path entry (路徑項目), 23
 path-like object (類路徑物件), 23
 PEP, 23
 portion (部分), 23
 positional argument (位置引數), 23
 provisional API (暫行 API), 23
 provisional package (暫行套件), 24
 Python 3000, 24
 Python Enhancement Proposals
 PEP 1, 23
 PEP 238, 17
 PEP 278, 27
 PEP 302, 20
 PEP 343, 14
 PEP 362, 12, 23
 PEP 411, 24
 PEP 420, 22, 23
 PEP 443, 18
 PEP 483, 18
 PEP 484, 11, 17, 18, 27
 PEP 492, 1214
 PEP 498, 16
 PEP 519, 23
 PEP 525, 12
 PEP 526, 11, 27
 PEP 585, 18
 PEP 649, 11
 PEP 683, 18
 PEP 703, 17, 18
 PEP 3116, 27
 PEP 3155, 24
 PYTHON_GIL, 18
 Pythonic (Python 風格的), 24

Q

qualified name (限定名稱), 24

R

reference count (參照計數), 24
 regular package (正規套件), 24
 REPL, 25

S

sequence (序列), 25

set comprehension (集合綜合運算), 25
single dispatch (單一調度), 25
slice (切片), 25
soft deprecated (軟性廢用), 25
special
 method (方法), 25
special method (特殊方法), 25
statement (陳述式), 25
static type checker (靜態型檢查器), 25
stdlib (標準函式庫), 25
strong reference (強參照), 25

T

t-strings (t 字串), 26
t-string (t 字串), 26
text encoding (文字編碼), 26
text file (文字檔案), 26
thread state, 26
token, 26
triple-quoted string (三引號字串), 26
type alias (型別名), 26
type hint (型別提示), 27
type (型別), 26

U

universal newlines (通用行字元), 27

V

variable annotation (變數釋), 27
virtual environment (虛擬環境), 27
virtual machine (虛擬機器), 27

W

walrus operator (海象運算子), 27
標準函式庫, 25

Z

Zen of Python (Python 之), 27