

---

# Installing Python Modules

リリース 2.7.15

**Guido van Rossum  
and the Python development team**

2月16, 2019

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



# 目次

第 1 章	重要用語集	3
第 2 章	基本的な使い方	5
第 3 章	どうすればいいの...?	7
3.1	... pip を 2.7.9 より前のバージョンの Python でインストールするには?	7
3.2	... パッケージを現在のユーザ用のみにインストールするには?	7
3.3	... 科学技術計算用の Python パッケージをインストールするには?	7
3.4	... インストールされた複数のバージョンの Python を平行して使うには?	8
第 4 章	よくあるインストールに関する問題	9
4.1	Linux で、システムの Python 内にインストールする	9
4.2	pip がインストールされていない	9
4.3	バイナリの拡張のインストール	9
付録 A	用語集	11
付録 B	このドキュメントについて	23
B.1	Python ドキュメント 貢献者	23
付録 C	歴史とライセンス	25
C.1	Python の歴史	25
C.2	Terms and conditions for accessing or otherwise using Python	26
C.3	取り入れられているソフトウェアのライセンスと謝辞	30
付録 D	Copyright	43
索引		45



Email [distutils-sig@python.org](mailto:distutils-sig@python.org)

人気のあるオープンソース開発プロジェクトがそうであるように、Python には貢献者たちとユーザたちの活発なサポートコミュニティがあり、またこれらはほかの Python 開発者たちに、彼らのソフトウェアのオープンソースライセンスのもとでの利用も可能にしてくれています。

これはほかの人が既に挙げた共通の (あるいは時折極めて稀有な!) 問題や、彼ら自身の解法による潜在的な貢献が共通の場所に蓄えられることによる恩恵によって、Python ユーザに共有と協調を効果的に行なうことの助けとなっています。

このガイドはこれらプロセスのうち、インストールについてをカバーします。あなた自身の Python プロジェクトを作成し、シェアするためのガイドについては [distribution guide](#) を参照してください。

---

注釈: あなたが企業や組織のユーザであれば、多くの組織がオープンソースソフトウェアの利用と貢献に関する彼ら独自のポリシーを持っていることに気をつけてください。Python によって提供される配布とインストールのツールを利用する際には、そのようなポリシーを考慮に入れてください。

---



## 第 1 章

# 重要用語集

- `pip` は推奨のインストーラ・プログラムです。Python 2.7.9 からは、Python バイナリ・インストーラに最初から付属するようになりました。
- 仮想環境 (virtual environment) とは、半ば隔離された Python 環境のことです。パッケージをシステム全体用にインストールするのではなく、特定のアプリケーションで用いるためにインストールすることを可能にします。
- `virtualenv` は仮想環境をつくるためのサードパーティ製ツールで、デフォルトで全ての作成済み仮想環境に対して `pip` がインストールされます。
- [Python Packaging Index](#) は Python パッケージのパブリック・リポジトリです。このリポジトリのパッケージは、他の Python ユーザが利用できるように、オープンソースでライセンスされています。
- [Python Packaging Authority](#) は、標準のパッケージングツール、関連するメタデータとファイルフォーマット標準の保守と発展を担っている、開発者・ドキュメントの著者のグループです。彼らは様々なツールやドキュメント、issue tracker を [GitHub](#) と [BitBucket](#) の両方で管理しています。
- `distutils` はオリジナルのビルド・配布システムで、Python 標準ライブラリに 1998 年に最初に追加されました。`distutils` の直接的な利用は段階的に取り払われていきますが、それは今でも現時点でのパッケージングと配布のインフラストラクチャの基礎として鎮座していて、標準ライブラリの一部として残っているだけでなく、その名前はほかの文脈でも生き続けています (Python のパッケージング標準の開発をまとめるのに使われているメーリングリストの名前のように)。





## 第 2 章

# 基本的な使い方

パッケージングのための標準ツールはすべてコマンドラインから使われることを想定しています。

以下のコマンドは、モジュールの最新バージョンとそれが依存するモジュールを、Python Packaging Index からインストールします。

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

---

注釈: POSIX ユーザ (Mac OS X と Linux ユーザを含みます) 向けには、このガイド内の例は、*virtual environment* の利用を前提にしています。そのような環境を作るのに `virtualenv` を、`pip` (`pip install virtualenv`) あるいはあなたのシステムのパッケージマネージャを通じて (大抵 `virtualenv` が `python-virtualenv` で呼ばれています) インストール出来ます。

Windows ユーザ向けには、このガイド内の例は、Python インストール時にシステムの PATH 環境変数が調整されていることを前提にしています。

---

正確なバージョンや最小のバージョンをコマンドライン上で直接指定することもできます。> や < などの比較演算子など、シェルが解釈する特殊文字を使う場合、パッケージ名とバージョンを二重引用符で囲んでください:

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

通常、適合するモジュールがインストール済である場合にそれを再度 `install` 実行を試みても効果はありません。既に存在しているモジュールのアップグレードには、明示的にそれを要求しなければなりません:

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

`pip` とその機能についての詳しい情報とリソースは [Python Packaging User Guide](#) にあります。

参考:

[Python Packaging User Guide: Installing Python Distribution Packages](#)



## 第 3 章

# どうすればいいの...？

以下はよくある課題への簡単な回答もしくは回答へのリンクです。

### 3.1 ... **pip** を 2.7.9 より前のバージョンの **Python** でインストールするには？

`pip` が Python に付属するのは 2.7.9 以降です。それ以前のバージョンでは、“Python Packaging User Guide” の記載にしたがって `pip` 自体をインストールする必要があります。

参考:

[Python Packaging User Guide: Requirements for Installing Packages](#)

### 3.2 ... パッケージを現在のユーザ用のみにインストールするには？

`python -m pip install` に `--user` オプションを付けてください。パッケージはシステムのすべてのユーザ用にではなく、現在のユーザ用のみにインストールされます。

### 3.3 ... 科学技術計算用の **Python** パッケージをインストールするには？

科学技術計算用の Python パッケージの多くがバイナリモジュールに複雑に依存しており、現在のところ `pip` を直接使ってインストールすることは容易ではありません。現時点では `pip` を使ってインストールしようとするよりも、別の手段を用いてインストールするほうが、多くの場合簡単でしょう。

参考:

[Python Packaging User Guide: Installing Scientific Packages](#)

### 3.4 ... インストールされた複数のバージョンの Python を平行して使うには？

Linux や Mac OS X その他の POSIX システムでは、バージョン番号付きの Python コマンドと `-m` スイッチを組み合わせて用いて、適切なバージョンの `pip` を実行してください。

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

( 適切にバージョン番号が付された `pip` コマンドが使えることもあります )

Windows では、Python ランチャーの `py` を `-m` スイッチとの組み合わせで使ってください。

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

## 第 4 章

# よくあるインストールに関する問題

### 4.1 Linux で、システムの Python 内にインストールする

Linux システムでは、Python のインストールは典型的には linux ディストリビューションの一部に含まれています。この Python インストールへのインストールには、システムに対する root 権限が必要で、また、pip によって期待されているものとは異なるコンポーネントにアップグレードすることで、システムのパッケージマネージャのオペレーションやシステムのほかのコンポーネントの邪魔をするかもしれません。

そのようなシステムでは、pip でパッケージをインストールする際には仮想環境を使うか、ユーザごとのインストールを行うのが、大抵良い方法です。

### 4.2 pip がインストールされていない

pip がデフォルトでインストールされていないことがあります。次のコマンドで問題の解決が見込めます:

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

他にも [pip のインストール](#) についての資料があります。

### 4.3 バイナリの拡張のインストール

Python プロジェクトの多くはソースに基いた配布に強く依存しており、インストールプロセスの一環として、エンドユーザ環境でソースから拡張モジュールをコンパイルすることを期待します。

バイナリ wheel フォーマットのサポートが導入されたことで、また、少なくとも Windows と Mac OS X についての wheels の公開が Python Packaging Index を通して出来るようになったことで、この問題についての開発者の時間の節約と、ユーザにとっては自身でビルドせずにビルド済み拡張をインストールするさらなる標準化に繋がるかもしれません。

ビルド済み wheel ファイルが未だ入手出来ない [scientific software](#) インストールにおけるいくつかの解法が、ローカルにビルドすることなく他のバイナリ拡張を得る助けになるかもしれません。

参考:



## 付録 A

# 用語集

>>> インタラクティブシェルにおけるデフォルトの Python プロンプトです。インタプリタでインタラクティブに実行されるコード例でよく出てきます。

... インタラクティブシェルにおいて、インデントされたコードブロック、対応する左右の区切り文字の組 (丸括弧 `()`、角括弧 `[]`、波括弧 `{}`、三重引用符) の内側、デコレーターの後に、コードを入力する際に表示されるデフォルトの Python プロンプトです。

2to3 Python 2.x のコードを Python 3.x のコードに変換するツールです。ソースコードを解析してその解析木を巡回 (traverse) することで検知できる、非互換性の大部分を処理します。

2to3 は標準ライブラリの `lib2to3` として利用できます。単体のツールとしての使えるスクリプトが `Tools/scripts/2to3` として提供されています。2to3-reference を参照してください。

abstract base class (抽象基底クラス) `abstract-base-classes` は *duck-typing* を補完するもので、`hasattr()` などの別のテクニックでは不恰好になる場合に インタフェースを定義する方法を提供します。Python は沢山のビルトイン ABCs を、(`collections` モジュールで) データ構造、(`numbers` モジュールで) 数値型、(`io` モジュールで) ストリーム型で提供しています。`abc` モジュールを利用して独自の ABC を作成することもできます。

argument 関数を呼び出す際に、*function* (または *method*) に渡す値です。引数には 2 種類あります。

- キーワード引数: 関数呼び出しの際に引数の前に識別子がついたもの (例: `name=`) や、`**` に続けた辞書の中の値として渡された引数。例えば、次の `complex()` の呼び出しでは、`3` と `5` がキーワード引数です:

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

- 位置引数: キーワード引数以外の引数。位置引数は引数リストの先頭に書くことができ、また `*` に続けた *iterable* の要素として渡すことができます。例えば、次の例では `3` と `5` は両方共位置引数です:

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

実引数は関数の実体において名前付きのローカル変数に割り当てられます。割り当てを行う規則については `calls` を参照してください。シンタックスにおいて実引数を表すためにあらゆる式を使うことが出来ます。評価された値はローカル変数に割り当てられます。

仮引数、FAQ の 実引数と仮引数の違いは何ですか? を参照してください。

`attribute` (属性) オブジェクトに関連付けられ、ドット表記式によって名前参照される値です。例えば、オブジェクト `o` が属性 `a` を持っているとき、その属性は `o.a` で参照されます。

`BDFL` 慈悲深き終身独裁者 (Benevolent Dictator For Life) の略です。Python の作者、[Guido van Rossum](#) のことです。

`bytes-like object` (バイト様オブジェクト) `str`、`bytearray` や `memoryview` といった `buffer protocol` をサポートするオブジェクトです。バイト様オブジェクトは圧縮、バイナリーデータの保存、ソケット経由の送信等の様々な操作に用いることが出来ます。操作によってはバイナリーデータはミューテブルでなくてはなりませんが、その場合全てのバイト様オブジェクトを用いることが出来るわけではありません。

`bytecode` (バイトコード) Python のソースコードはバイトコードへとコンパイルされます。バイトコードは Python プログラムの CPython インタプリタ内部での表現です。バイトコードはまた、`.pyc` や `.pyo` ファイルにキャッシュされ、同じファイルの実行が二度目にはより高速になります (ソースコードからバイトコードへの再度のコンパイルは回避されます)。このバイトコードは、各々のバイトコードに対応するサブルーチン呼び出すような "仮想機械 (*virtual machine*)" で動作する "中間言語 (*intermediate language*)" といえます。重要な注意として、バイトコードは異なる Python 仮想マシン間で動作することは期待されていませんし、Python リリース間で安定でもありません。

バイトコードの命令一覧は `dis` モジュールにあります。

`class` (クラス) ユーザー定義オブジェクトを作成するためのテンプレートです。クラス定義は普通、そのクラスのインスタンス上の操作をするメソッドの定義を含みます。

`classic class` (旧スタイルクラス) `object` を継承していないクラス全てを指します。新スタイルクラス (*new-style class*) も参照してください。旧スタイルクラスは Python 3 で削除されます。

`coercion` (型強制) 同じ型の 2 つの引数を要する演算の最中に、ある型のインスタンスを別の型に暗黙のうちに変換することです。例えば、`int(3.15)` は浮動小数点数を整数の 3 にします。しかし、`3+4.5` の場合、各引数は型が異なっていて (一つは整数、一つは浮動小数点数)、加算をする前に同じ型に変換しなければいけません。そうでないと、`TypeError` 例外が投げられます。2 つの被演算子間の型強制は組み込み関数の `coerce` を使って行えます。従って、`3+4.5` は `operator.add(*coerce(3, 4.5))` を呼び出すことに等しく、`operator.add(3.0, 4.5)` という結果になります。型強制を行わない場合、たとえ互換性のある型であっても、すべての引数はプログラマーが、単に `3+4.5` とするのではなく、`float(3)+4.5` というように、同じ型に正規化しなければいけません。

`complex number` (複素数) よく知られている実数系を拡張したもので、すべての数は実部と虚部の和として表されます。虚数は虚数単位 ( $-1$  の平方根) に実数を掛けたもので、一般に数学では  $i$  と書かれ、工学では  $j$  と書かれます。Python は複素数に組み込みで対応し、後者の表記を取っています。虚部は末尾に `j` をつけて書きます。例えば `3+1j` です。 `math` モジュールの複素数版を利用するには、`cmath` を使います。複素数の使用はかなり高度な数学の機能です。必要性を感じなければ、ほぼ間違いなく無視してしまってもよいでしょう。



context manager (コンテキストマネージャ) `with` 文で扱われる、`__enter__()` と `__exit__()` メソッドを定義することで環境を制御するオブジェクトです。PEP 343 を参照してください。

CPython [python.org](http://python.org) で配布されている、Python プログラミング言語の標準的な実装です。”CPython” という単語は、この実装を Jython や IronPython といった他の実装と区別する必要がある場合に利用されます。

decorator (デコレータ) 別の関数を返す関数で、通常、`@wrapper` 構文で関数変換として適用されます。デコレータの一般的な利用例は、`classmethod()` と `staticmethod()` です。

デコレータの文法はシンタックスシュガーです。次の 2 つの関数定義は意味的に同じものです:

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

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

同じ概念がクラスにも存在しますが、あまり使われません。デコレータについて詳しくは、関数定義およびクラス定義のドキュメントを参照してください。

descriptor (デスクリプタ) メソッド `__get__()`, `__set__()`, あるいは `__delete__()` が定義されている新スタイル (*new-style*) のオブジェクトです。あるクラス属性がデスクリプタである場合、その属性を参照するときに、そのデスクリプタに束縛されている特別な動作を呼び出します。通常、`get`, `set`, `delete` のために `a.b` と書くと、`a` のクラス辞書内でオブジェクト `b` を検索しますが、`b` がデスクリプタの場合にはデスクリプタで定義されたメソッドを呼び出します。デスクリプタの理解は、Python を深く理解する上で鍵となります。というのは、デスクリプタこそが、関数、メソッド、プロパティ、クラスメソッド、静的メソッド、そしてスーパクラスの参照といった多くの機能の基盤だからです。

デスクリプタのメソッドに関して詳しくは、`descriptors` を参照してください。

dictionary (辞書) 任意のキーを値に対応付ける連想配列です。`__hash__()` メソッドと `__eq__()` メソッドを実装した任意のオブジェクトをキーにできます。Perl ではハッシュ (`hash`) と呼ばれています。

dictionary view (ビュー) `dict.viewkeys()`, `dict.viewvalues()`, `dict.viewitems()` が返すオブジェクトを辞書ビュー (dictionary view) と呼びます。これらは辞書のエントリに対する動的なビューで、つまり辞書の変更されるとビューはそれら変更を反映します。辞書ビューを完全なリストにするには `list(dictview)` としてください。`dict-views` を参照してください。

docstring クラス、関数、モジュールの最初の式である文字列リテラルです。そのスイートの実行時には無視されますが、コンパイラによって識別され、そのクラス、関数、モジュールの `__doc__` 属性として保存されます。イントロスペクションできる (訳注: 属性として参照できる) ので、オブジェクトのドキュメントを書く標準的な場所です。

duck-typing あるオブジェクトが正しいインタフェースを持っているかを決定するのにオブジェクトの型を見ないプログラミングスタイルです。代わりに、単純にオブジェクトのメソッドや属性が呼ばれたり使われたりします。(「アヒルのように見えて、アヒルのように鳴けば、それはアヒルである。’) インタフェースを型より重視することで、上手くデザインされたコードは、ポリモーフィックな代替を許し

で柔軟性を向上させます。ダックタイピングは `type()` や `isinstance()` による判定を避けます。(ただし、ダックタイピングを [抽象基底クラス](#) で補完することもできます。) その代わり、典型的に `hasattr()` 判定や [EAFP](#) プログラミングを利用します。

**EAFP** 「認可をとるより許しを請う方が容易 (easier to ask for forgiveness than permission、マーフィーの法則)」の略です。この Python で広く使われているコーディングスタイルでは、通常は有効なキーや属性が存在するものと仮定し、その仮定が誤っていた場合に例外を捕捉します。この簡潔で手早く書けるコーディングスタイルには、`try` 文および `except` 文がたくさんあるのが特徴です。このテクニックは、C のような言語でよく使われている [LBYL](#) スタイルと対照的なものです。

**expression** (式) 何かの値に評価される、一つづきの構文 (a piece of syntax). 言い換えると、リテラル、名前、属性アクセス、演算子や関数呼び出しといった、値を返す式の要素の組み合わせ。他の多くの言語と違い、Python は言語の全ての構成要素が式というわけではありません。 `print` や `if` のように、式にはならない、文 (*statement*) もあります。代入も式ではなく文です。

**extension module** (拡張モジュール) C や C++ で書かれたモジュールで、Python の C API を利用して Python コアやユーザーコードとやりとりします。

**file object** (ファイルオブジェクト) 下位のリソースへのファイル志向 API (`read()` や `write()` メソッドを持つもの) を公開しているオブジェクトです。ファイルオブジェクトは、作成された手段によって、実際のディスク上のファイルや、その他のタイプのストレージや通信デバイス (例えば、標準入出力、インメモリバッファ、ソケット、パイプ、等) へのアクセスを媒介できます。ファイルオブジェクトは *file-like objects* や *streams* と呼ばれます。

ファイルオブジェクトには 3 つの種類があります: 生のバイナリーファイル、バッファされたバイナリーファイル、そしてテキストファイルです。これらのインターフェースは `io` モジュール内で定義されています。ファイルオブジェクトを作成する標準的な方法は `open()` 関数を利用することです。

**file-like object** [file object](#) と同義です。

**finder** モジュールの [loader](#) を探すオブジェクト。 `findmodule()` という名前のメソッドを実装していなければなりません。詳細については [PEP 302](#) を参照してください。

**floor division** 一番近い小さい整数に丸める数学除算。floor division 演算子は `//` です。例えば、 `11 // 4` は 2 になり、 `float` の true division の結果 `2.75` と異なります。 `(-11) // 4` は `-2.75` を小さい方に丸めるので `-3` になることに注意してください。 [PEP 238](#) を参照してください。

**関数** (関数) 呼び出し側に値を返す一連の文のことです。関数には 0 以上の [実引数](#) を渡すことが出来ます。実体の実行時に引数を使用することが出来ます。 [仮引数](#)、[メソッド](#)、`function` を参照してください。

**\_\_future\_\_** 互換性のない新たな機能を現在のインタプリタで有効にするためにプログラマが利用できる擬似モジュールです。例えば、式 `11/4` は現状では 2 になります。この式を実行しているモジュールで

```
from __future__ import division
```

を行って 真の除算操作 (*true division*) を有効にすると、式 `11/4` は `2.75` になります。実際に `__future__` モジュールを `import` してその変数を評価すれば、新たな機能が初めて追加されたのがいつで、いつデフォルトの機能になる予定かわかります。

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

garbage collection (ガベージコレクション) それ以上使われなくなったメモリを解放する処理です。Python は、参照カウントと、循環参照を見つけて破壊する循環ガベージコレクタと、を使ってガベージコレクションを行います。

generator A function which returns an iterator. It looks like a normal function except that it contains `yield` statements for producing a series of values usable in a for-loop or that can be retrieved one at a time with the `next()` function. Each `yield` temporarily suspends processing, remembering the location execution state (including local variables and pending try-statements). When the generator resumes, it picks up where it left off (in contrast to functions which start fresh on every invocation).

generator expression (ジェネレータ式) イテレータを返す式です。普通の式に、ループ変数を定義する `for` 式、範囲、そして省略可能な `if` 式がつづいているように見えます。こうして構成された式は、外側の関数に向けて値を生成します:

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

GIL *global interpreter lock* を参照してください。

global interpreter lock (グローバルインタプリタロック) *CPython* インタプリタが利用している、一度に Python のバイトコード (*bytecode*) 実行するスレッドは一つだけであることを保証する仕組みです。これにより (`dict` などの重要な組み込み型を含む) などのオブジェクトモデルが同時アクセスに対して暗黙的に安全になるので、*CPython* の実装がシンプルになります。インタプリタ全体をロックすることで、マルチプロセッサマシンが生じる並列化のコストに対して、楽にインタプリタをマルチスレッド化できます。

ただし、標準あるいは外部のいくつかの拡張モジュールは、圧縮やハッシュ計算などの計算の重い処理をするときに GIL を解放するように設計されています。また、I/O 処理をするときも GIL は常に解放されます。

過去に”自由なマルチスレッド化”したインタプリタ (供用されるデータを細かい粒度でロックする) が開発されましたが、一般的なシングルスプロセッサの場合のパフォーマンスが悪かったので成功しませんでした。このパフォーマンスの問題を克服しようとする、実装がより複雑になり保守コストが増加すると考えられています。

hashable (ハッシュ可能) ハッシュ可能なオブジェクトとは、生存期間中変わらないハッシュ値を持ち (`__hash__()` メソッドが必要)、他のオブジェクトと比較ができる (`__eq__()` か `__cmp__()` メソッドが必要) オブジェクトです。同値なハッシュ可能オブジェクトは必ず同じハッシュ値を持つ必要があります。

ハッシュ可能なオブジェクトは辞書のキーや集合のメンバーとして使えます。辞書や集合のデータ構造は内部でハッシュ値を使っているからです。

Python のイミュータブルな組み込みオブジェクトはハッシュ可能ですが、ミュータブルなコンテ

ナ (例えばリストや辞書) はハッシュ不可能です。ユーザー定義のクラスのインスタンスであるようなオブジェクトはデフォルトでハッシュ可能です。それらは全て非等価を比較し (自身を除いて)、ハッシュ値は `id()` より得られます。

**IDLE** Python の統合開発環境 (Integrated DeveLopment Environment) です。IDLE は Python の標準的な配布に同梱されている基本的な機能のエディタとインタプリタ環境です。

**immutable** (イミュータブル) 固定の値を持ったオブジェクトです。イミュータブルなオブジェクトには、数値、文字列、おびタプルなどがあります。これらのオブジェクトは値を変えられません。別の値を記憶させる際には、新たなオブジェクトを作成しなければなりません。イミュータブルなオブジェクトは、固定のハッシュ値が必要となる状況で重要な役割を果たします。辞書のキーがその例です。

**integer division** (整数除算) 剰余を考慮しない数学的除算です。例えば、式  $11/4$  は現状では  $2.75$  ではなく  $2$  になります。これは切り捨て除算 (*floor division*) とも呼ばれます。二つの整数間で除算を行うと、結果は (端数切捨て関数が適用されて) 常に整数になります。しかし、被演算子の一方が (`float` のような) 別の数値型の場合、演算の結果は共通の型に型強制されます (型強制 (*coercion*) 参照)。例えば、浮動小数点数で整数を除算すると結果は浮動小数点になり、場合によっては端数部分を伴います。// 演算子を / の代わりに使うと、整数除算を強制できます。 `__future__` も参照してください。

**importing** あるモジュールの Python コードが別のモジュールの Python コードで使えるようにする処理です。

**importer** モジュールを探してロードするオブジェクト。 `finder` と `loader` のどちらでもあるオブジェクト。

**interactive** (対話的) Python には対話的インタプリタがあり、文や式をインタプリタのプロンプトに入力すると即座に実行されて結果を見ることができます。 `python` と何も引数を与えずに実行してください。(コンピュータのメインメニューから Python の対話的インタプリタを起動できるかもしれません。) 対話的インタプリタは、新しいアイデアを試してみたり、モジュールやパッケージの中を覗いてみる (`help(x)` を覚えておいてください) のに非常に便利なツールです。

**interpreted** Python はインタプリタ形式の言語であり、コンパイラ言語の対極に位置します。(バイトコードコンパイラがあるために、この区別は曖昧ですが。) ここでのインタプリタ言語とは、ソースコードのファイルを、まず実行可能形式にしてから実行させるといった操作なしに、直接実行できることを意味します。インタプリタ形式の言語は通常、コンパイラ形式の言語よりも開発 / デバッグのサイクルは短いものの、プログラムの実行は一般に遅いです。対話的 (*interactive*) も参照してください。

**iterable** (反復可能オブジェクト) 要素を一つずつ返せるオブジェクトです。反復可能オブジェクトの例には、 (`list`, `str`, `tuple` といった) 全てのシーケンス型や、 `dict` や `file` といった幾つかの非シーケンス型、あるいは `__iter__()` か `__getitem__()` メソッドを実装したクラスのインスタンスが含まれます。反復可能オブジェクトは `for` ループ内やその他多くのシーケンス (訳注: ここでのシーケンスとは、シーケンス型ではなくただの列という意味) が必要となる状況 (`zip()`, `map()`, ...) で利用できます。反復可能オブジェクトを組み込み関数 `iter()` の引数として渡すと、オブジェクトに対するイテレータを返します。このイテレータは一連の値を引き渡す際に便利です。反復可能オブジェクトを使う際には、通常 `iter()` を呼んだり、イテレータオブジェクトを自分で扱う必要はありません。 `for` 文ではこの操作を自動的に扱い、無名の変数を作成してループの間イテレータを記憶します。イテレータ (*iterator*) シーケンス (*sequence*), およびジェネレータ (*generator*) も参照してください。

**iterator** (イテレータ) データの流れを表現するオブジェクトです。イテレータの `next()` メソッドを繰り返して呼び出すと、流れの中の要素を一つずつ返します。データがなくなると、代わりに `StopIteration`

例外を送出します。その時点で、イテレータオブジェクトは尽きており、それ以降は `next()` を何度呼んでも `StopIteration` を送じます。イテレータは、そのイテレータオブジェクト自体を返す `__iter__()` メソッドを実装しなければならないので、イテレータは他の `iterable` を受理するほとんどの場所で利用できます。はっきりとした例外は複数の反復を行うようなコードです。(list のような) コンテナオブジェクトは、自身を `iter()` 関数にオブジェクトに渡したり `for` ループ内で使うたびに、新たな未使用のイテレータを生成します。これをイテレータで行おうとすると、前回のイテレーションで使用済みの同じイテレータオブジェクトを単純に返すため、空のコンテナのようになってしまいます。

詳細な情報は `typeiter` にあります。

**key function** (キー関数) キー関数、あるいは照合関数とは、ソートや順序比較のための値を返す呼び出し可能なオブジェクト (callable) です。例えば、`locale.strxfrm()` をキー関数に使用すれば、ロケール依存のソートの慣習にのっとったソートキーを返します。

Python には、キー関数を受け付けて要素がどのように順序付けられたりグループ化されたりするかを制御するような、いくつかのツールがあります。例えば、`min()`, `max()`, `sorted()`, `list.sort()`, `heapq.nsmallest()`, `heapq.nlargest()`, `itertools.groupby()` です。

キー関数を作る方法はいくつかあります。例えば、`str.lower()` メソッドをキー関数として使って大文字小文字を区別しないソートができます。他には、`lambda` 式を使って `lambda r: (r[0], r[2])` のようなアドホックなキー関数を作ることができます。また、`operator` モジュールは `attrgetter()`, `itemgetter()`, `methodcaller()` というキー関数コンストラクタを提供しています。キー関数の作り方、使い方に関する例は、`Sorting HOW TO` を参照してください。

**keyword argument** [実引数](#) を参照してください。

**lambda** (ラムダ) 無名のインライン関数で、関数が呼び出されたときに評価される 1 つの式 (*expression*) を含みます。ラムダ関数を作る構文は `lambda [parameters]: expression` です。

**LBYL** 「ころばぬ先の杖 (look before you leap)」の略です。このコーディングスタイルでは、呼び出しや検索を行う前に、明示的に前提条件 (pre-condition) 判定を行います。[EAFP](#) アプローチと対照的で、`if` 文がたくさん使われるのが特徴的です。

マルチスレッド化された環境では、LBYL アプローチは「見る」過程と「飛ぶ」過程の競合状態を引き起こすリスクがあります。例えば、`if key in mapping: return mapping[key]` というコードは、判定の後、別のスレッドが探索の前に `mapping` から `key` を取り除くと失敗します。この問題は、ロックするか EAFP アプローチを使うことで解決できます。

**list** (リスト) Python の組み込みのシーケンス (*sequence*) です。リストという名前ですが、リンクリストではなく、他の言語で言う配列 (array) と同種のもので、要素へのアクセスは  $O(1)$  です。

**list comprehension** (リスト内包表記) シーケンス内の全てあるいは一部の要素を処理して、その結果からなるリストを返す、コンパクトな書き方です。`result = ["0x%02x" % x for x in range(256) if x % 2 == 0]` とすると、0 から 255 までの偶数を 16 進数表記 (0x..) した文字列からなるリストを生成します。`if` 節はオプションです。`if` 節がない場合、`range(256)` の全ての要素が処理されます。

**loader** モジュールをロードするオブジェクト。`load_module()` という名前のメソッドを定義していなけ



ればなりません。詳細は [PEP 302](#) を参照してください。

**mapping** (マップ、マッピング) 任意のキーに対する検索をサポートしていて、`Mapping` が `MutableMapping` の抽象基底クラスを実装しているコンテナオブジェクト。例えば、`dict`, `collections.defaultdict`, `collections.OrderedDict`, `collections.Counter` はマップ型です。

**metaclass** (メタクラス) クラスのクラスです。クラス定義は、クラス名、クラスの辞書と、基底クラスのリストを作ります。メタクラスは、それら 3 つを引数として受け取り、クラスを作る責任を負います。ほとんどのオブジェクト指向言語は (訳注:メタクラスの) デフォルトの実装を提供しています。Python が特別なのはカスタムのメタクラスを作成できる点です。ほとんどのユーザーにとって、メタクラスは全く必要のないものです。しかし、一部の場面では、メタクラスは強力でエレガントな方法を提供します。たとえば属性アクセスのログを取ったり、スレッドセーフ性を追加したり、オブジェクトの生成を追跡したり、シングルトンを実装するなど、多くの場面で利用されます。

詳細は `metaclasses` を参照してください。

**method** (メソッド) クラス本体の中で定義された関数。そのクラスのインスタンスの属性として呼び出された場合、メソッドはインスタンスオブジェクトを第一引数 (*argument*) として受け取ります (この第一引数は通常 `self` と呼ばれます)。関数 と [ネストされたスコープ](#) も参照してください。

**method resolution order** (メソッド解決順序) 探索中に基底クラスが構成要素を検索される順番です。2.3 以降の Python インタープリタが使用するアルゴリズムの詳細については [The Python 2.3 Method Resolution Order](#) を参照してください。

**モジュール** (モジュール) Python コードの組織単位としてはたらくオブジェクトです。モジュールは任意の Python オブジェクトを含む名前空間を持ちます。モジュールは *importing* の処理によって Python に読み込まれます。

[パッケージ](#) を参照してください。

**MRO** *method resolution order* を参照してください。

**mutable** (ミュータブル) ミュータブルなオブジェクトは、`id()` を変えることなく値を変更できます。イミュータブル (*immutable*) も参照してください。

**named tuple** (名前付きタプル) タプルに似ていて、インデックス指定できる要素に名前付き属性でもアクセス出来るクラスです (例えば、`time.localtime()` はタプルに似たオブジェクトを返し、その `year` には `t[0]` のようなインデックスによるアクセスと、`t.tm_year` のような名前付き要素としてのアクセスが可能です)。

名前付きタプルには、`time.struct_time` のような組み込み型もありますし、通常のクラス定義によって作成することもできます。名前付きタプルを `collections.namedtuple()` ファクトリ関数で作成することもできます。最後の方法で作った名前付きタプルには自動的に、`Employee(name='jones', title='programmer')` のような自己ドキュメント表現 (self-documenting representation) などの機能が付いてきます。

**namespace** (名前空間) 変数を記憶している場所です。名前空間は辞書を用いて実装されています。名前空間には、ローカル、グローバル、組み込み名前空間、そして (メソッド内の) オブジェクトのネストされた名前空間があります。例えば、関数 `_builtin_.open()` と `os.open()` は名前空間で区別さ

れます。名前空間はまた、ある関数をどのモジュールが実装しているかをはっきりさせることで、可読性やメンテナンス性に寄与します。例えば、`random.seed()`、`itertools.izip()` と書くことで、これらの関数がそれぞれ `random` モジュールや `itertools` モジュールで実装されていることがはっきりします。

**nested scope** (ネストされたスコープ) 外側で定義されている変数を参照する機能。具体的に言えば、ある関数が別の関数の中で定義されている場合、内側の関数は外側の関数中の変数を参照できます。ネストされたスコープからは変数の参照だけができ、外側の変数を変更できないことに注意してください。それは常に最も内側のスコープに対する書き込みになります。対照的に、ローカル変数は最も内側のスコープ内の読み書き両方します。同様に、グローバル変数を使うとグローバル名前空間の値を読み書きします。

**new-style class** (新スタイルクラス) `object` から継承したクラス全てを指します。これには `list` や `dict` のような全ての組み込み型が含まれます。`__slots__()`、デスクリプタ、プロパティ、`__getattr__()` といった、Python の新しい機能を使えるのは新スタイルクラスだけです。

より詳しい情報は `newstyle` を参照してください。

**object** (オブジェクト) 状態 (属性や値) と定義された振る舞い (メソッド) をもつ全てのデータ。もしくは、全ての新スタイルクラス (*new-style class*) の究極の基底クラスのこと。

**package** (パッケージ) サブモジュールや再帰的にサブパッケージを含むことの出来る *module* のことです。専門的には、パッケージは `__path__` 属性を持つ Python オブジェクトです。

**parameter** (仮引数) 名前付の実体で `term:関数<function>` (や `term:メソッド<method>`) の定義において関数が受ける **実引数** を明示します。仮引数には 4 種類あります:

- 位置またはキーワード: **位置** または **キーワード引数** として渡すことができる引数を指定します。これはたとえば以下の `foo` や `bar` のように、デフォルトの仮引数の種類です:

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

- 位置専用: 位置によってのみ与えられる引数を指定します。Python に位置専用仮引数を定義する文法はありません。しかし、組み込み関数には位置専用仮引数を持つもの (例: `abs()`) があります。
- 可変長位置: (他の仮引数で既に受けられた任意の位置引数に加えて) 任意の個数の位置引数が与えられることを指定します。このような仮引数は、以下の `args` のように仮引数名の前に `*` をつけることで定義できます:

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

- 可変長キーワード: (他の仮引数で既に受けられた任意のキーワード引数に加えて) 任意の個数のキーワード引数が与えられることを指定します。このような仮引数は、上の例の `kwargs` のように仮引数名の前に `**` をつけることで定義できます。

仮引数はオプションと必須の引数のどちらも指定でき、オプションの引数にはデフォルト値も指定できます。

*argument*、FAQ の 実引数と仮引数の違いは何ですか?、`function` を参照してください。

**PEP** Python Enhancement Proposal. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment. PEPs should provide a concise technical specification and a rationale for proposed features.

PEPs are intended to be the primary mechanisms for proposing major new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python. The PEP author is responsible for building consensus within the community and documenting dissenting opinions.

See [PEP 1](#).

位置引数 [実引数](#) を参照してください。

**Python 3000** Python 3.x リリースラインのニックネームです。(Python 3 が遠い将来の話だった頃に作られた言葉です。) ”Py3k” と略されることもあります。

**Pythonic** 他の言語で一般的な考え方で書かれたコードではなく、Python の特に一般的なイディオムに従った考え方やコード片。例えば、Python の一般的なイディオムでは `for` 文を使ってイテラブルのすべての要素に渡ってループします。他の多くの言語にはこの仕組みはないので、Python に慣れていない人は代わりに数値のカウンターを使うかもしれません:

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

これに対し、きれいな Pythonic な方法は:

```
for piece in food:
    print piece
```

**reference count** (参照カウント) あるオブジェクトに対する参照の数。参照カウントが 0 になったとき、そのオブジェクトは破棄されます。参照カウントは通常は Python のコード上には現れませんが、[CPython](#) 実装の重要な要素です。 `sys` モジュールは、プログラマーが任意のオブジェクトの参照カウントを知るための `getrefcount()` 関数を提供しています。

**\_\_slots\_\_** 新スタイルクラス ([new-style class](#)) 内で、インスタンス属性の記憶に必要な領域をあらかじめ定義しておき、それとひきかえにインスタンス辞書を排除してメモリの節約を行うための宣言です。これはよく使われるテクニックですが、正しく動作させるのには少々手際を要するので、例えばメモリが死活問題となるようなアプリケーション内にインスタンスが大量に存在するといった稀なケースを除き、使わないのがベストです。

**sequence** (シーケンス) 特殊メソッド `__getitem__()` で整数インデックスによる効率的な要素へのアクセスをサポートし、`len()` で長さを返すような反復可能オブジェクト ([iterable](#)) です。組み込みシーケンス型には、`list`, `str`, `tuple`, `unicode` などがあります。 `dict` は `__getitem__()` と `__len__()` もサポートしますが、検索の際に任意の変更不能 ([immutable](#)) なキーを使うため、シーケンスではなくマップ (mapping) とみなされているので注意してください。

**slice** (スライス) 多くの場合、シーケンス ([sequence](#)) の一部を含むオブジェクト。スライスは、添字記号 `[]` で数字の間にコロンを書いたときに作られます。例えば、`variable_name[1:3:5]` です。添字記号は `slice` オブジェクトを内部で利用しています。(もしくは、古いバージョンの、`__getslice__()` と `__setslice__()` を利用します。)



**special method** (特殊メソッド) ある型に特定の操作、例えば加算をするために Python から暗黙に呼び出されるメソッド。この種類のメソッドは、メソッド名の最初と最後にアンダースコア 2 つがついています。特殊メソッドについては `specialnames` で解説されています。

**statement** (文) 文はスイート(コードの”ブロック”)に不可欠な要素です。文は [式](#) かキーワードから構成されるもののどちらかです。後者には `if`、`while`、`for` があります。

**struct sequence** A tuple with named elements. Struct sequences expose an interface similiar to *named tuple* in that elements can be accessed either by index or as an attribute. However, they do not have any of the named tuple methods like `_make()` or `_asdict()`. Examples of struct sequences include `sys.float_info` and the return value of `os.stat()`.

**triple-quoted string** (三重クォート文字列) 3 つの連続したクォート記号 (") かアポストロフィー (') で囲まれた文字列。通常の (一重) クォート文字列に比べて表現できる文字列に違いはありませんが、幾つかの理由で有用です。1 つか 2 つの連続したクォート記号をエスケープ無しに書くことができますし、行継続文字 (\) を使わなくても複数行にまたがることのできるので、ドキュメンテーション文字列を書く時に特に便利です。

**型** (型) Python オブジェクトの型はオブジェクトがどのようなものかを決めます。あらゆるオブジェクトは型を持っています。オブジェクトの型は `__class__` 属性でアクセスしたり、`type(obj)` で取得したり出来ます。

**universal newlines** テキストストリームの解釈法の一つで、以下のすべてを行末と認識します: Unix の行末規定 `'\n'`、Windows の規定 `'\r\n'`、古い Macintosh の規定 `'\r'`。利用法について詳しくは、[PEP 278](#) と [PEP 3116](#)、さらに `str.splitlines()` も参照してください。

**virtual environment** (仮想環境) 協調的に切り離された実行環境です。これにより Python ユーザとアプリケーションは同じシステム上で動いている他の Python アプリケーションの挙動に干渉することなく Python パッケージのインストールと更新を行うことができます。

**virtual machine** (仮想マシン) 完全にソフトウェアにより定義されたコンピュータ。Python の仮想マシンは、バイトコードコンパイラが出力したバイトコード (*bytecode*) を実行します。

**Zen of Python** (Python の悟り) Python を理解し利用する上での導きとなる、Python の設計原則と哲学をリストにしたものです。対話プロンプトで `import this` とするとこのリストを読めます。



## 付録 B

# このドキュメントについて

このドキュメントは、Python のドキュメントを主要な目的として作られた ドキュメントプロセッサの [Sphinx](#) を利用して、[reStructuredText](#) 形式のソースから生成されました。

ドキュメントとそのツールセットの開発は、Python 自身と同様に完全にボランティアの努力です。もしあなたが貢献したいなら、どのようにすればよいかについて [reporting-bugs](#) ページをご覧ください。新しいボランティアはいつでも歓迎です!

多大な感謝を:

- Fred L. Drake, Jr., オリジナルの Python ドキュメントツールセットの作成者で、ドキュメントの多くを書きました。
- [Docutils](#) プロジェクト. [reStructuredText](#) と [docutils](#) ツールセットを作成しました。
- Fredrik Lundh の [Alternative Python Reference](#) プロジェクトから Sphinx は多くのアイデアを得ました。

## B.1 Python ドキュメント 貢献者

多くの方々が Python 言語、Python 標準ライブラリ、そして Python ドキュメンテーションに貢献してくれています。ソース配布物の [Misc/ACKS](#) に、それら貢献してくれた人々を部分的にはありますがリストアップしてあります。

Python コミュニティからの情報提供と貢献がなければこの素晴らしいドキュメンテーションは生まれませんでした – ありがとう!



## 付録 C

# 歴史とライセンス

## C.1 Python の歴史

Python は 1990 年代の始め、オランダにある Stichting Mathematisch Centrum (CWI, <https://www.cwi.nl/> 参照) で Guido van Rossum によって ABC と呼ばれる言語の後継言語として生み出されました。その後多くの人々が Python に貢献していますが、Guido は今日でも Python 製作者の先頭に立っています。

1995 年、Guido は米国ヴァージニア州レストンにある Corporation for National Research Initiatives (CNRI, <https://www.cnri.reston.va.us/> 参照) で Python の開発に携わり、いくつかのバージョンをリリースしました。

2000 年 3 月、Guido と Python のコア開発チームは BeOpen.com に移り、BeOpen PythonLabs チームを結成しました。同年 10 月、PythonLabs チームは Digital Creations (現在の Zope Corporation, <http://www.zope.com/> 参照) に移りました。そして 2001 年、Python に関する知的財産を保有するための非営利組織 Python Software Foundation (PSF、<https://www.python.org/psf/> 参照) を立ち上げました。このとき Zope Corporation は PSF の賛助会員になりました。

Python のリリースは全てオープンソース (オープンソースの定義は <https://opensource.org/> を参照してください) です。歴史的にみて、ごく一部を除くほとんどの Python リリースは GPL 互換になっています; 各リリースについては下表にまとめてあります。

リリース	ベース	西暦年	権利	GPL 互換
0.9.0 - 1.2	n/a	1991-1995	CWI	yes
1.3 - 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	no
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2 以降	2.1.1	2001-現在	PSF	yes

注釈: 「GPL 互換」という表現は、Python が GPL で配布されているという意味ではありません。Python のライセンスは全て、GPL と違い、変更したバージョンを配布する際に変更をオープンソースにしなくてもかまいません。GPL 互換のライセンスの下では、GPL でリリースされている他のソフトウェアと Python を組み合わせられますが、それ以外のライセンスではそうではありません。

---

Guido の指示の下、これらのリリースを可能にくださった多くのボランティアのみなさんに感謝します。

## C.2 Terms and conditions for accessing or otherwise using Python

### C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 2.7.15

1. This LICENSE AGREEMENT is between the Python Software Foundation,  
→ ("PSF"), and  
the Individual or Organization ("Licensee") accessing and otherwise,  
→ using Python  
2.7.15 software in source or binary form and its associated,  
→ documentation.
2. Subject to the terms and conditions of this License Agreement, PSF,  
→ hereby  
grants Licensee a nonexclusive, royalty-free, world-wide license to,  
→ reproduce,  
analyze, test, perform and/or display publicly, prepare derivative,  
→ works,  
distribute, and otherwise use Python 2.7.15 alone or in any derivative  
version, provided, however, that PSF's License Agreement and PSF's,  
→ notice of  
copyright, i.e., "Copyright ^c2^a9 2001-2019 Python Software,  
→ Foundation; All Rights  
Reserved" are retained in Python 2.7.15 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 2.7.15 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  
2.7.15.

4. PSF is making Python 2.7.15 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 2.7.15 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.15  
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A  
→RESULT OF  
MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.15, 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 2.7.15, 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

- |   |
|---|
| <p>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").</p> <p>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</p> |
|---|

(次のページに続く)

(前のページからの続き)

- 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,

(次のページに続く)



(前のページからの続き)

Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: <http://hdl.handle.net/1895.22/1013>."

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
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.

(次のページに続く)

(前のページからの続き)

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

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

### C.3 取り入れられているソフトウェアのライセンスと謝辞

この節では、Python 配布物に取り入れられているサードパーティソフトウェアのライセンスと謝辞の、不完全だが成長し続けるリストを記述します。

#### C.3.1 メルセンヌツイスター (Mersenne Twister)

`random` モジュールは、<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> からダウンロードしたコードに基づいたコードベースを含んでいます。オリジナルのコードのコメントをそのまま次に示します:

```
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.
```

(次のページに続く)

(前のページからの続き)

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

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

Any feedback is very welcome.

<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)

### C.3.2 ソケット

socket モジュールは、WIDE プロジェクト <http://www.wide.ad.jp/> の別々のソースファイルにコーディングされていた `getaddrinfo()` 関数と `getnameinfo()` 関数を使っています:

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)

(次のページに続く)

(前のページからの続き)

```
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 浮動小数点数例外の制御

fpectl モジュールのソースコードは次の告知を含んでいます:

```
-----  
/                               Copyright (c) 1996.                               \  
|                               The Regents of the University of California.          \  
|                               All rights reserved.                                \  
|                                                                                   \  
| Permission to use, copy, modify, and distribute this software for                \  
| any purpose without fee is hereby granted, provided that this en-               \  
| tire 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 work was produced at the University of California, Lawrence                 \  
| Livermore National Laboratory under contract no. W-7405-ENG-48                   \  
| between the U.S. Department of Energy and The Regents of the                   \  
| University of California for the operation of UC LLNL.                          \  
|                                                                                   \  
|                               DISCLAIMER                                             \  
|                                                                                   \  
| This software was prepared as an account of work sponsored by an                \  
| agency of the United States Government. Neither the United States                \  
| Government nor the University of California nor any of their em-                 \  
| ployees, makes any warranty, express or implied, or assumes any                 \  
| liability or responsibility for the accuracy, completeness, or                   \  
| usefulness of any information, apparatus, product, or process                   \  
| disclosed, or represents that its use would not infringe                       \  
| privately-owned rights. Reference herein to any specific commer-                 \  
| cial products, process, or service by trade name, trademark,                     \  
| manufacturer, or otherwise, does not necessarily constitute or                  \  
| imply its endorsement, recommendation, or favoring by the United                \  
| States Government or the University of California. The views and                 \  
| opinions of authors expressed herein do not necessarily state or                 \  
| reflect those of the United States Government or the University                  \  
| of California, and shall not be used for advertising or product                  \  
| endorsement purposes.                                                            \  
-----
```

### C.3.4 MD5 メッセージダイジェストアルゴリズム

md5 モジュールのソースコードは次の告知を含んでいます:

Copyright (C) 1999, 2002 Aladdin Enterprises. All rights reserved.

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

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

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

L. Peter Deutsch  
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose text is available at

<http://www.ietf.org/rfc/rfc1321.txt>

The code is derived from the text of the RFC, including the test suite (section A.5) but excluding the rest of Appendix A. It does not include any code or documentation that is identified in the RFC as being copyrighted.

The original and principal author of md5.h is L. Peter Deutsch <ghost@aladdin.com>. Other authors are noted in the change history that follows (in reverse chronological order):

2002-04-13 lpd Removed support for non-ANSI compilers; removed references to Ghostscript; clarified derivation from RFC 1321; now handles byte order either statically or dynamically.  
1999-11-04 lpd Edited comments slightly for automatic TOC extraction.  
1999-10-18 lpd Fixed typo in header comment (ansi2knr rather than md5); added conditionalization for C++ compilation from Martin Purschke <purschke@bnl.gov>.  
1999-05-03 lpd Original version.

### C.3.5 非同期ソケットサービス

asynchat モジュールと asyncore モジュールは次の告知を含んでいます:

Copyright 1996 by Sam Rushing

(次のページに続く)

(前のページからの続き)

All Rights Reserved

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

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

### C.3.6 Cookie の管理

Cookie モジュールは次の告知を含んでいます:

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.3.7 実行の追跡

trace モジュールは次の告知を含んでいます:

portions copyright 2001, Autonomous Zones Industries, Inc., all rights...  
err... reserved and offered to the public under the terms of the  
Python 2.2 license.

Author: Zooko O'Whielacronx  
http://zooko.com/  
mailto:zooko@zooko.com

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

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

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

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

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

### C.3.8 UUencode 関数と UUdecode 関数

uu モジュールは次の告知を含んでいます:

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:

(次のページに続く)

(前のページからの続き)

- 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.3.9 XML リモートプロシージャコール

xmlrpclib モジュールは次の告知を含んでいます:

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

### C.3.10 test\_epoll

test\_epoll は次の告知を含んでいます:

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

(次のページに続く)



(前のページからの続き)

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.3.11 Select kqueue

select は kqueue インターフェースについての次の告知を含んでいます:

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.3.12 strtod と dtoa

The file `Python/dtoa.c`, which supplies C functions `dtoa` and `strtod` for conversion of C doubles to and from strings, is derived from the file of the same name by David M. Gay, currently available from <http://www.netlib.org/fp/>. The original file, as retrieved on March 16, 2009, contains the following copyright and licensing notice:

```
/*****
 *
 * 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.3.13 OpenSSL

The modules `hashlib`, `posix`, `ssl`, `crypt` use the OpenSSL library for added performance if made available by the operating system. Additionally, the Windows and Mac OS X installers for Python may include a copy of the OpenSSL libraries, so we include a copy of the OpenSSL license here:

```
LICENSE ISSUES
=====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts. Actually both licenses are BSD-style
Open Source licenses. In case of any license issues related to OpenSSL
please contact openssl-core@openssl.org.

OpenSSL License
-----

/* =====
 * Copyright (c) 1998-2008 The OpenSSL 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
```

(次のページに続く)

(前のページからの続き)

```

*   distribution.
*
* 3. All advertising materials mentioning features or use of this
*   software must display the following acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
*   endorse or promote products derived from this software without
*   prior written permission. For written permission, please contact
*   openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
*   nor may "OpenSSL" appear in their names without prior written
*   permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
*   acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED 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 OpenSSL PROJECT OR
* ITS 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.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).  This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

-----

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*

```

(次のページに続く)

(前のページからの続き)

```
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* 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 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. All advertising materials mentioning features or use of this software
*    must display the following acknowledgement:
*    "This product includes cryptographic software written by
*      Eric Young (eay@cryptsoft.com)"
*    The word 'cryptographic' can be left out if the rouines from the library
*    being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*    the apps directory (application code) you must include an acknowledgement:
*    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``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.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

### C.3.14 expat

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

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

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

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

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

### C.3.15 libffi

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

```
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,
```

(次のページに続く)

(前のページからの続き)

```
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.3.16 zlib

The `zlib` extension is built using an included copy of the `zlib` sources if the `zlib` version found on the system is too old to be used for the build:

```
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler
```

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

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

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

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

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

## 付録 D

# Copyright

Python and this documentation is:

Copyright 2001-2019 Python Software Foundation. All rights reserved.

Copyright 2000 BeOpen.com. All rights reserved.

Copyright 1995-2000 Corporation for National Research Initiatives. All rights reserved.

Copyright 1991-1995 Stichting Mathematisch Centrum. All rights reserved.

---

ライセンスおよび許諾に関する完全な情報は、[歴史とライセンス](#) を参照してください。





# 索引

..., 11

\_\_future\_\_, 14

\_\_slots\_\_, 20

>>>, 11

2to3, 11

abstract base class, 11

argument, 11

attribute, 12

BDFL, 12

bytecode, 12

bytes-like object, 12

class, 12

classic class, 12

coercion, 12

complex number, 12

context manager, 13

CPython, 13

decorator, 13

descriptor, 13

dictionary, 13

dictionary view, 13

docstring, 13

duck-typing, 13

EAFP, 14

expression, 14

extension module, 14

file object, 14

file-like object, 14

finder, 14

floor division, 14

garbage collection, 15

generator, 15, 15

generator expression, 15, 15

GIL, 15

global interpreter lock, 15

hashable, 15

IDLE, 16

immutable, 16

importer, 16

importing, 16

integer division, 16

interactive, 16

interpreted, 16

iterable, 16

iterator, 16

key function, 17

keyword argument, 17

lambda, 17

LBYL, 17

list, 17

list comprehension, 17

loader, 17

mapping, 18

metaclass, 18

method, 18

method resolution order, 18

MRO, 18

mutable, 18

named tuple, 18

namespace, 18

nested scope, 19

new-style class, 19

object, 19

package, 19

parameter, 19

PEP, 20

Python 3000, 20

Python Enhancement Proposals

PEP 1, 20

PEP 238, 14

PEP 278, 21

PEP 302, 14, 18

PEP 3116, 21

PEP 343, 13

Pythonic, 20

reference count, 20

sequence, 20

slice, 20

special method, 21

statement, 21

struct sequence, 21

triple-quoted string, 21

universal newlines, 21

virtual environment, 21

virtual machine, 21

Zen of Python, 21

モジュール, 18

位置引数, 20

関数, 14

型, 21