
Installing Python Modules

リリース 3.9.21

Guido van Rossum
and the Python development team

12 月 09, 2024

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 3.4 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 を並行して使うには?	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 章	用語集	11
付録 B 章	このドキュメントについて	29
B.1	Python ドキュメント 貢献者	29
付録 C 章	歴史とライセンス	31
C.1	Python の歴史	31
C.2	Terms and conditions for accessing or otherwise using Python	32
C.3	Licenses and Acknowledgements for Incorporated Software	36
付録 D 章	Copyright	51
索引		53
索引		53

Email 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。

重要用語集

- `pip` は推奨されるインストーラ・プログラムです。Python 3.4 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** は、システム全体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使用するためにパッケージをインストールできるようにする半独立型の Python 環境です。
- `venv` は仮想環境をつくるための標準ツールです。Python 3.3 から Python の一部になりました。Python 3.4 以降では、仮想環境をつくる際はそのすべてで `pip` をインストールするように、あらかじめ設定されています。
- `virtualenv` は `venv` のサードパーティの代替案 (で前身) です。`virtualenv` により仮想環境を Python 3.4 より前のバージョンで使うことが出来ます。以前のバージョンは `venv` を提供していませんし、作成された環境に `pip` を自動的にインストールすることも出来ません。
- The [Python Package Index](#) is a public repository of open source licensed packages made available for use by other Python users.
- [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)。

バージョン 3.5 で変更: 仮想環境の作成には、`venv` の使用をお勧めします。

参考:

[Python Packaging User Guide: Creating and using virtual environments](#)

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。

The following command will install the latest version of a module and its dependencies from the Python Package Index:

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

注釈: For POSIX users (including macOS and Linux users), the examples in this guide assume the use of a *virtual environment*.

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。

仮想環境の作成は venv モジュールを通して行われます。有効な仮想環境にインストールするには 上記のコマンドを使います。

参考:

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

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 3.4 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 3.4 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 を並行して使うには？

On Linux, macOS, and other POSIX systems, use the versioned Python commands in combination with the `-m` switch to run the appropriate copy of `pip`:

```
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
```

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。

With the introduction of support for the binary `wheel` format, and the ability to publish wheels for at least Windows and macOS through the Python Package Index, this problem is expected to diminish over time, as users are more regularly able to install pre-built extensions rather than needing to build them themselves.

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:

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。

... 次のものが考えられます:

- 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。
- 組み込みの定数 `Ellipsis`。

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) 抽象基底クラスは *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であったり微妙に誤る (例えば `magic methods` の場合) 場合にインターフェースを定義する方法を提供します。ABC は仮想 (virtual) サブクラスを導入します。これは親クラスから継承しませんが、それでも `isinstance()` や `issubclass()` に認識されます; `abc` モジュールのドキュメントを参照してください。Python には、多くの組み込み ABC が同梱されています。その対象は、(`collections.abc` モジュールで) データ構造、(`numbers` モジュールで) 数、(`io` モジュールで) ストリーム、(`importlib.abc` モジュールで) インポートファインダ及びローダーです。`abc` モジュールを利用して独自の ABC を作成できます。

annotation (アノテーション) 変数、クラス属性、関数のパラメータや返り値に関係するラベルです。慣例により *type hint* として使われています。

ローカル変数のアノテーションは実行時にはアクセスできませんが、グローバル変数、クラス属性、関数のアノテーションはそれぞれモジュール、クラス、関数の `__annotations__` 特殊属性に保持されています。

機能の説明がある *variable annotation*, *function annotation*, **PEP 484**, **PEP 526** を参照してください。

argument (実引数) 関数を呼び出す際に、**関数** (または **メソッド**) に渡す値です。実引数には 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を書くことができ、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できます。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?、**PEP 362** を参照してください。

asynchronous context manager (非同期コンテキストマネージャ) `__aenter__()` と `__aexit__()` メソッドを定義することで `async with` 文内の環境を管理するオブジェクトです。**PEP 492** で導入されました。

asynchronous generator (非同期ジェネレータ) *asynchronous generator iterator* を返す関数です。 `async def` で定義されたコルーチン関数に似ていますが、`yield` 式を持つ点で異なります。 `yield` 式は `async for` ループで使用できる値の並びを生成するのに使用されます。

通常は非同期ジェネレータ関数を指しますが、文脈によっては **非同期ジェネレータイテレータ** を指す場合があります。意図された意味が明らかでない場合、明瞭化のために完全な単語を使用します。

非同期ジェネレータ関数には、`async for` 文や `async with` 文だけでなく `await` 式もあることがあります。

asynchronous generator iterator (非同期ジェネレータイテレータ) *asynchronous generator* 関数で生成されるオブジェクトです。

これは、`__anext__()` メソッドを使って呼び出されたときに `awaitable` オブジェクトを返す *asynchronous iterator* です。この `awaitable` オブジェクトは、次の `yield` 式までの非同期ジェネレータ関数の本体を実行します。

`yield` にくるたびに、その位置での実行状態 (ローカル変数と保留状態の `try` 文) 処理は一時停止されます。`__anext__()` で返された他の `awaitable` によって **非同期ジェネレータイテレータ** が実際に再開されたとき、中断した箇所を取得します。**PEP 492** および **PEP 525** を参照してください。

asynchronous iterable (非同期イテラブル) `async for` 文の中で使用できるオブジェクトです。自身の `__aiter__()` メソッドから *asynchronous iterator* を返さなければなりません。**PEP 492** で導入されました。

asynchronous iterator (非同期イテレータ) `__aiter__()` と `__anext__()` メソッドを実装したオブジェクトです。`__anext__` は *awaitable* オブジェクトを返さなければなりません。`async for` は `StopAsyncIteration` 例外を送出するまで、非同期イテレータの `__anext__()` メソッドが返す *awaitable* を解決します。**PEP 492** で導入されました。

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で参照される値です。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。

awaitable (待機可能) `await` 式で使うことが出来るオブジェクトです。*coroutine* か、`__await__()` メソッドがあるオブジェクトです。**PEP 492** を参照してください。

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。

binary file (バイナリファイル) *bytes-like* オブジェクト の読み込みおよび書き込みができる **ファイルオブジェクト** です。バイナリファイルの例は、バイナリモード ('rb', 'wb' or 'rb+') で開かれたファイル、`sys.stdin.buffer`、`sys.stdout.buffer`、`io.BytesIO` や `gzip.GzipFile` のインスタンスです。

`str` オブジェクトの読み書きができるファイルオブジェクトについては、*text file* も参照してください。

bytes-like object `bufferobjects` をサポートしていて、C 言語の意味で 連続した contiguous バッファを提供可能なオブジェクト。`bytes`、`bytearray`、`array.array` や、多くの一般的な `memoryview` オブジェクトがこれに当たります。*bytes-like* オブジェクトは、データ圧縮、バイナリファイルへの保存、ソケットを経由した送信など、バイナリデータを要求するいろいろな操作に利用することができます。

幾つかの操作ではバイナリデータを変更する必要があります。その操作のドキュメントではよく ”読み書き可能な *bytes-like* オブジェクト” に言及しています。変更可能なバッファオブジェクトには、`bytearray` と `bytearray` の `memoryview` などが含まれます。また、他の幾つかの操作では不変なオブジェクト内のバイナリデータ (”読み出し専用の *bytes-like* オブジェクト”) を必要します。それには `bytes` と `bytes` の `memoryview` オブジェクトが含まれます。

bytecode (バイトコード) Python のソースコードは、Python プログラムの CPython インタプリタの内部表現であるバイトコードへとコンパイルされます。バイトコードは `.pyc` ファイルにキャッシュされ、同じファイルが二度目に実行されるとき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 ”中間言語 (intermediate language)” は、各々のバイトコードに対応する機械語を実行する **仮想マシン** で動作するといえます。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や、Python リリース間で安定であることは期待されていません。

バイトコードの命令一覧は `dis` モジュール にあります。

callback (コールバック) 将来のある時点で実行されるために引数として渡される関数

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。

class variable (クラス変数) クラス上に定義され、クラスレベルで (つまり、クラスのインスタンス上ではなく) 変更されることを目的としている変数です。

coercion (型強制) 同じ型の 2 引数を伴う演算の最中に行われる、ある型のインスタンスの別の型への暗黙の変換です。例えば、`int(3.15)` は浮動小数点数を整数 3 に変換します。しかし `3+4.5` では、各引数は型が異なり (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できなければ `TypeError` 例外が投げられます。型強制がなかったら、すべての引数は、たとえ互換な型であっても、単に `3+4.5` ではなく `float(3)+4.5` というように、プログラマーが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よいでしょう。

context manager (コンテキストマネージャ) `__enter__()` と `__exit__()` メソッドを定義することで `with` 文内の環境を管理するオブジェクトです。[PEP 343](#) を参照してください。

context variable (コンテキスト変数) コンテキストに依存して異なる値を持つ変数。これは、ある変数の値が各々の実行スレッドで異なり得るスレッドローカルストレージに似ています。しかしコンテキスト変数では、1 つの実行スレッドにいくつかのコンテキストがあり得、コンテキスト変数の主な用途は並列な非同期タスクの変数の追跡です。`contextvars` を参照してください。

contiguous (隣接、連続) バッファが厳密に **C-連続** または **Fortran 連続** である場合に、そのバッファは連続しているとみなせます。ゼロ次元バッファは C 連続であり Fortran 連続です。一次元の配列では、その要素は必ずメモリ上で隣接するように配置され、添字がゼロから始まり増えていく順序で並びます。多次元の C-連続な配列では、メモリアドレス順に要素を巡る際には最後の添え字が最初に変わるのに対し、Fortran 連続な配列では最初の添え字が最初に動きます。

coroutine (コルーチン) コルーチンはサブルーチンのより一般的な形式です。サブルーチンには決められた地点から入り、別の決められた地点から出ます。コルーチンには多くの様々な地点から入る、出る、再開することができます。コルーチンは `async def` 文で実装できます。[PEP 492](#) を参照してください。

coroutine function (コルーチン関数) *coroutine* オブジェクトを返す関数です。コルーチン関数は `async def` 文で実装され、`await`、`async for`、および `async with` キーワードを持つことが出来ます。これらは [PEP 492](#) で導入されました。

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

(次のページに続く)

(前のページからの続き)

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

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__()` を定義しているオブジェクトです。あるクラス属性がデスクリプタであるとき、属性探索によって、束縛されている特別な動作が呼び出されます。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` がデスクリプタであればそれぞれのデスクリプタメソッドが呼び出されます。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する詳細は、`descriptors` や `Descriptor How To Guide` 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 comprehension (辞書内包表記) `iterable` 内の全てあるいは一部の要素を処理して、その結果からなる辞書を返すコンパクトな書き方です。`results = {n: n ** 2 for n in range(10)}` とすると、キー `n` を値 `n ** 2` に対応付ける辞書を生成します。`comprehensions` を参照してください。

dictionary view (辞書ビュー) `dict.keys()`、`dict.values()`、`dict.items()` が返すオブジェクトです。辞書の項目の動的なビューを提供します。すなわち、辞書が変更されるとビューはそれを反映します。辞書ビューを強制的に完全なリストにするには `list(dictview)` を使用してください。`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ー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。’) インター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て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) 何かの値と評価される、一まとまりの構文 (a piece of syntax) です。言い換えると、式とはリテラル、名前、属性アクセス、演算子や関数呼び出しなど、値を返す式の要素の積み重ねです。他の多くの言語と違い、Python で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。while のように、式としては使えない **文** 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。

f-string 'f' や 'F' が先頭に付いた文字列リテラルは "f-string" と呼ばれ、これは フォーマット済み文字列リテラル の短縮形の名称です。PEP 498 も参照してください。

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指向 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* とも呼ばれます。

ファイルオブジェクトには実際には 3 種類あります: 生の **バイナリファイル**、バッファされた **バイナリファイル**、そして **テキストファイル** です。インターフェイスは io モジュールで定義されています。ファイルオブジェクトを作る標準的な方法は open() 関数を使うことです。

file-like object *file object* と同義です。

finder (ファインダ) インポートされているモジュールの *loader* の発見を試行するオブジェクトです。

Python 3.3 以降では 2 種類のファインダがあります。sys.meta_path で使用される *meta path finder* と、sys.path_hooks で使用される *path entry finder* です。

詳細については PEP 302、PEP 420 および PEP 451 を参照してください。

floor division (切り捨て除算) 一番近い整数に切り捨てる数学的除算。切り捨て除算演算子は // です。例えば、11 // 4 は 2 になり、それとは対称に浮動小数点数の真の除算では 2.75 が返ってきます。(-11) // 4 は -2.75 を **小さい方に丸める** (訳注: 負の無限大への丸めを行う) 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。

function (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。

function annotation (関数アノテーション) 関数のパラメータや戻り値の *annotation* です。

関数アノテーションは、通常は **型ヒント** のために使われます: 例えば、この関数は 2 つの int 型の引数を取ると期待され、また int 型の戻り値を持つと期待されています。

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

関数アノテーションの文法は function の節で解説されています。

機能の説明がある *variable annotation* と PEP 484 を参照してください。

`__future__` A future statement, `from __future__ import <feature>`, directs the compiler to compile the current module using syntax or semantics that will become standard in a future release of Python. The `__future__` module documents the possible values of *feature*. By importing this module and evaluating its variables, you can see when a new feature was first added to the language and when it will (or did) become the default:

```
>>> 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) これ以降使われることのないメモリを解放する処理です。Python は、参照カウントと、循環参照を検出し破壊する循環ガベージコレクタを使ってガベージコレクションを行います。ガベージコレクタは `gc` モジュールを使って操作できます。

generator (ジェネレータ) *generator iterator* を返す関数です。通常の関数に似ていますが、`yield` 式を持つ点で異なります。`yield` 式は、`for` ループで使用できたり、`next()` 関数で値を 1 つずつ取り出したりできる、値の並びを生成するのに使用されます。

通常はジェネレータ関数を指しますが、文脈によっては **ジェネレータイテレータ** を指す場合があります。意図された意味が明らかでない場合、明瞭化のために完全な単語を使用します。

generator iterator (ジェネレータイテレータ) *generator* 関数で生成されるオブジェクトです。

`yield` のたびに局所実行状態 (局所変数や未処理の `try` 文などを含む) を記憶して、処理は一時的に中断されます。**ジェネレータイテレータ** が再開されると、中断した位置を取得します (通常の関数が実行のたびに新しい状態から開始するのと対照的です)。

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` 節、範囲、そして省略可能な `if` 節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
```

generic function (ジェネリック関数) 異なる型に対し同じ操作をする関数群から構成される関数です。呼び出し時にどの実装を用いるかはディスパッチアルゴリズムにより決定されます。

single dispatch、`functools.singledispatch()` デコレータ、**PEP 443** を参照してください。

generic type A *type* that can be parameterized; typically a container class such as `list` or `dict`. Used for *type hints* and *annotations*.

For more details, see generic alias types, **PEP 483**, **PEP 484**, **PEP 585**, and the `typing` module.

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** を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) 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と引き換えに、インタプリタを簡単にマルチスレッド化できるようになります。

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 を解除するように設計されています。また、I/O 処理をする場合 GIL は常に解除されます。

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。

hash-based pyc (ハッシュベース pyc ファイル) 正当性を判別するために、対応するソースファイルの最終更新時刻ではなくハッシュ値を使用するバイトコードのキャッシュファイルです。

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__()` 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、ほとんどがハッシュ可能です。(リストや辞書のような) ミュータブルなコンテナはハッシュ不可能です。(タプルや `frozenset` のような) イミュータブルなコンテナは、要素がハッシュ可能であるときのみハッシュ可能です。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 (自身を除いて) 比較結果は非等価であり、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、およ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。

import path *path based finder* が import するモジュールを検索する場所 (または *path entry*) のリスト。import 中、このリストは通常 `sys.path` から来ますが、サブパッケージの場合は親パッケージの `__path__` 属性からも来ます。

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](#) も参照してください。

interpreter shutdown Python インタープリターはシャットダウンを要請された時に、モジュールやすべてのクリティカルな内部構造をなどの、すべての確保したリソースを段階的に開放する、特別なフェーズに入ります。このフェーズは [ガベージコレクタ](#) を複数回呼び出します。これによりユーザー定義のデストラクターや `weakref` コールバックが呼び出されることがあります。シャットダウンフェーズ中に実行されるコードは、それが依存するリソースがすでに機能しない (よくある例はライブラリーモジュールや `warning` 機構です) ために様々な例外に直面します。

インタープリタがシャットダウンする主な理由は `__main__` モジュールや実行されていたスクリプトの実行が終了したことです。

iterable (反復可能オブジェクト) 要素を一度に 1 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 [ファイルオブジェクト](#) といった幾つかの非シーケンス型、あるいは [Sequence](#) 意味論を実装した `__iter__()` メソッドか `__getitem__()` メソッドを持つ任意の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。通常は反復可能オブジェクトを使う際には、`iter()` を呼んだりイテレータオブジェクトを自分で操作する必要はありません。`for` 文ではこの操作を自動的に扱い、一時的な無名の変数を作成してループを回している間イテレータを保持します。[イテレータ](#)、[シーケンス](#)、[ジェネレータ](#) 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呼び出す (または組み込み関数 `next()` に渡す) 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 out します。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()` をキー関数に使えば、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 の多くのツールはキー関数を受け取り要素の並び順やグループ化を管理します。`min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()`, `itertools.groupby()` 等があります。

キー関数を作る方法はいくつかあります。例えば `str.lower()` メソッドを大文字小文字を区別しないソートを行うキー関数として使うことが出来ます。あるいは、`lambda r: (r[0], r[2])` のような `lambda` 式から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 3 つのキー関数コンストラクタを提供しています。キー関数の作り方と使い方の例は [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](#) 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](#) 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) シーケンス中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、コンパクトな方法です。`result = ['{:04x}'.format(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。ローダーは一般的に [finder](#) から返されます。詳細は [PEP 302](#) を、[abstract base class](#) については `importlib.abc.Loader` を参照してください。

magic method [special method](#) のくだけた同義語です。

`mapping` (マッピング) 任意のキー探索をサポートしていて、`Mapping` か `MutableMapping` の抽象基底クラスで指定されたメソッドを実装しているコンテナオブジェクトです。例えば、`dict`, `collections.defaultdict`, `collections.OrderedDict`, `collections.Counter` などです。

`meta path finder` `sys.meta_path` を検索して得られた [finder](#)。meta path finder は [path entry finder](#) と関係はありますが、別物です。

meta path finder が実装するメソッドについては `importlib.abc.MetaPathFinder` を参照してください。

`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

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** 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。

module (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。

module spec モジュールをロードするのに使われるインポート関連の情報を含む名前空間です。`importlib.machinery.ModuleSpec` のインスタンスです。

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** も参照してください。

named tuple ”名前付きタプル”という用語は、タプルを継承していて、インデックスが付く要素に対し属性を使つてのアクセスもできる任意の型やクラスに应用されています。その型やクラスは他の機能も持っていることもあります。

`time.localtime()` や `os.stat()` の戻り値を含むいくつかの組み込み型は名前付きタプルです。他の例は `sys.float_info` です:

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

(上の例のように) いくつかの名前付きタプルは組み込み型になっています。その他にも名前付きタプルは、通常のクラス定義で `tuple` を継承し、名前のフィールドを定義して作成できます。そのようなクラスは手動で書いたり、`collections.namedtuple()` ファクトリ関数で作成したりできます。後者の方法は、手動で書いた名前付きタプルや組み込みの名前付きタプルには無い付加的なメソッドを追加できます。

namespace (名前空間) 変数が格納される場所です。名前空間は辞書として実装されます。名前空間にはオブジェクトの (メソッドの) 入れ子になったものだけでなく、局所的なもの、大域的なもの、そして組み込みのものがあります。名前空間は名前の衝突を防ぐことによってモジュール性をサポートする。例えば関数 `builtins.open` と `os.open()` は名前空間で区別されています。また、どのモジュールが関数を実装しているか明示することによって名前空間は可読性と保守性を支援します。例えば、`random.seed()` や `itertools.islice()` と書くと、それぞれモジュール `random` や `itertools` で実装されていることが明らかです。

namespace package (名前空間パッケージ) サブパッケージのコンテナとしてのみ提供される **PEP 420** で定義された *package* です。名前空間パッケージは物理的な表現を持たないことができ、`__init__.py` ファイルを持たないため、*regular package* とは異なります。

module を参照してください。

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です。例えば、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はデフォルトでは変数の参照だけができ、変数の代入はできないので注意してください。ローカル変数は、最も内側のスコープで変数を読み書きします。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。`nonlocal` で外側の変数に書き込めます。

new-style class (新スタイルクラス) 今では全てのクラスオブジェクトに使われている味付けの古い名前です。以前の Python のバージョンでは、新スタイルクラスのみが `__slots__`、デスクリプタ、`__getattr__()`、クラスメソッド、そして静的メソッド等の Python の新しい、多様な機能を利用できました。

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** 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。

regular package と *namespace package* を参照してください。

parameter (仮引数) 名前付の実体で **関数** (や **メソッド**) の定義において関数が受ける **実引数** を指定します。仮引数には5種類あります:

- **位置またはキーワード:** **位置** であるいは **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。位置専用の引数は 関数定義の引数のリストの中でそれらの後ろに `/` を含めることで定義できます。例えば下記の `posonly1` と `posonly2` は位置専用引数になります:

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

- **キーワード専用:** キーワードによってのみ与えられる引数を指定します。キーワード専用の引数を定義できる場所は、例えば以下の `kw_only1` や `kw_only2` のように、関数定義の仮引数リストに

含めた可変長位置引数または裸の `*` の後です:

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

- **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。

仮引数、FAQ の 実引数と仮引数の違いは何か?、`inspect.Parameter` クラス、`function` セクション、**PEP 362** を参照してください。

path entry *path based finder* が `import` するモジュールを探す *import path* 上の 1 つの場所です。

path entry finder `sys.path_hooks` にある callable (つまり *path entry hook*) が返した *finder* です。与えられた *path entry* にあるモジュールを見つける方法を知っています。

パスエントリーファインダが実装するメソッドについては `importlib.abc.PathEntryFinder` を参照してください。

path entry hook `sys.path_hook` リストにある callable で、指定された *path entry* にあるモジュールを見つける方法を知っている場合に *path entry finder* を返します。

path based finder デフォルトの *meta path finder* の 1 つは、モジュールの *import path* を検索します。

path-like object (path-like オブジェクト) ファイルシステムパスを表します。path-like オブジェクトは、パスを表す `str` オブジェクトや `bytes` オブジェクト、または `os.PathLike` プロトコルを実装したオブジェクトのどれかです。`os.PathLike` プロトコルをサポートしているオブジェクトは `os.fspath()` を呼び出すことで `str` または `bytes` のファイルシステムパスに変換できます。`os.fsdecode()` と `os.fsencode()` はそれぞれ `str` あるいは `bytes` になるのを保証するのに使えます。**PEP 519** で導入されました。

PEP Python Enhancement Proposal。PEP は、Python コミュニティに対して情報を提供する、あるいは Python の新機能やその過程や環境について記述する設計文書です。PEP は、機能についての簡潔な技術的仕様と提案する機能の論拠 (理論) を伝えるべきです。

PEP は、新機能の提案にかかる、コミュニティによる問題提起の集積と Python になされる設計決断の文書化のための最上位の機構となることを意図しています。PEP の著者にはコミュニティ内の合意形成を行うこと、反対意見を文書化することの責務があります。

PEP 1 を参照してください。

portion **PEP 420** で定義されている、namespace package に属する、複数のファイルが (zip ファイルに格納されている場合もある) 1 つのディレクトリに格納されたもの。

positional argument (位置引数) **実引数** を参照してください。

provisional API (暫定 API) 標準ライブラリの後方互換性保証から計画的に除外されたものです。そのようなインターフェースへの大きな変更は、暫定であるとされている間は期待されていませんが、コア開発者によって必要とみなされれば、後方非互換な変更 (インターフェースの削除まで含まれる) が行われます。このような変更はむやみに行われるものではありません -- これは API を組み込む前には見落とされていた重大な欠陥が露呈したときにのみ行われます。

暫定 API についても、後方互換性のない変更は「最終手段」とみなされています。問題点が判明した場合でも後方互換な解決策を探すべきです。

このプロセスにより、標準ライブラリは問題となるデザインエラーに長い間閉じ込められることなく、時代を超えて進化を続けられます。詳細は **PEP 411** を参照してください。

provisional package *provisional API* 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)
```

qualified name (修飾名) モジュールのグローバルスコープから、そのモジュールで定義されたクラス、関数、メソッドへの、"パス" を表すドット名表記です。**PEP 3155** で定義されています。トップレベルの関数やクラスでは、修飾名はオブジェクトの名前と同じです:

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

モジュールへの参照で使われると、**完全修飾名** (*fully qualified name*) はすべての親パッケージを含む全体のドット名表記、例えば `email.mime.text` を意味します:

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

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。

regular package 伝統的な、`__init__.py` ファイルを含むディレクトリとしての *package*。

namespace package を参照してください。

__slots__ クラス内での宣言で、インスタンス属性の領域をあらかじめ定義しておき、インスタンス辞書を排除することで、メモリを節約します。これはよく使われるテクニックですが、正しく扱うには少しトリッキーなので、稀なケース、例えばメモリが死活問題となるアプリケーションでインスタンスが大量に存在する、といったときを除き、使わないのがベストです。

sequence (シーケンス) 整数インデクスによる効率的な要素アクセスを `__getitem__()` 特殊メソッドを通じてサポートし、長さを返す `__len__()` メソッドを定義した *iterable* です。組み込みシーケンス型には、`list`, `str`, `tuple`, `bytes` などがあります。`dict` は `__getitem__()` と `__len__()` もサポートしますが、検索の際に整数ではなく任意の *immutable* なキーを使うため、シーケンスではなくマッピング (mapping) とみなされているので注意してください。

The `collections.abc.Sequence` abstract base class defines a much richer interface that goes beyond just `__getitem__()` and `__len__()`, adding `count()`, `index()`, `__contains__()`, and `__reversed__()`. Types that implement this expanded interface can be registered explicitly using `register()`.

set comprehension A compact way to process all or part of the elements in an iterable and return a set with the results. `results = {c for c in 'abracadabra' if c not in 'abc'}` generates the set of strings `{'r', 'd'}`. See comprehensions.

single dispatch *generic function* の一種で実装は一つの引数の型により選択されます。

slice (スライス) 一般に *シーケンス* の一部を含むオブジェクト。スライスは、添字表記 `[]` で与えられた複数の数の間にコロンを書くことで作られます。例えば、`variable_name[1:3:5]` です。角括弧 (添字) 記号は `slice` オブジェクトを内部で利用しています。

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。

text encoding A string in Python is a sequence of Unicode code points (in range U+0000--U+10FFFF).

To store or transfer a string, it needs to be serialized as a sequence of bytes.

Serializing a string into a sequence of bytes is known as "encoding", and recreating the string from the sequence of bytes is known as "decoding".

There are a variety of different text serialization codecs, which are collectively referred to as "text encodings".

text file (テキストファイル) `str` オブジェクトを読み書きできる *file object* です。しばしば、テキストファイルは実際にバイト指向のデータストリームにアクセスし、**テキストエンコーディング** を自動的に行います。テキストファイルの例は、`sys.stdin`, `sys.stdout`, `io.StringIO` インスタンスなどをテキストモード ('r' or 'w') で開いたファイルです。

bytes-like **オブジェクト** を読み書きできるファイルオブジェクトについては、**バイナリファイル** も参照してください。

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ため、ドキュメンテーション文字列を書く時に特に便利です。

type (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。

type alias (型エイリアス) 型の別名で、型を識別子に代入して作成します。

型エイリアスは **型ヒント** を単純化するのに有用です。例えば:

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

これは次のように読みやすくなります:

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

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

機能の説明がある `typing` と **PEP 484** を参照してください。

type hint (型ヒント) 変数、クラス属性、関数のパラメータや戻り値の期待される型を指定する *annotation* です。

型ヒントは必須ではなく Python では強制ではありませんが、静的型解析ツールにとって有用であり、IDE のコード補完とリファクタリングの手助けになります。

グローバル変数、クラス属性、関数で、ローカル変数でないものの型ヒントは `typing.get_type_hints()` で取得できます。

機能の説明がある `typing` と [PEP 484](#) を参照してください。

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 `bytes.splitlines()` も参照してください。

variable annotation (変数アノテーション) 変数あるいはクラス属性の *annotation*。

変数あるいはクラス属性に注釈を付けたときは、代入部分は任意です:

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

変数アノテーションは通常は **型ヒント** のために使われます: 例えば、この変数は `int` の値を取ることを期待されています:

```
count: int = 0
```

変数アノテーションの構文については [annassign](#) 節で解説しています。

この機能について解説している [function annotation](#), [PEP 484](#), [PEP 526](#) を参照してください。

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。

`venv` を参照してください。

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** 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。

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群の開発は、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！（訳注：日本語訳の問題については、GitHub 上の [Issue Tracker](#) で報告をお願いします。）

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 for his Alternative Python Reference project from which Sphinx got many good ideas.

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で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！

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 Reserch 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, <https://www.zope.org/> 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

Python software and documentation are licensed under the *PSF License Agreement*.

Starting with Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF License Agreement and the *Zero-Clause BSD license*.

Some software incorporated into Python is under different licenses. The licenses are listed with code falling under that license. See *Licenses and Acknowledgements for Incorporated Software* for an incomplete list of these licenses.

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 3.9.21

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 3.9.21 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 3.9.21 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2023 Python Software Foundation; All Rights Reserved" are retained in Python 3.9.21 alone or in any derivative version prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 3.9.21 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 3.9.21.
4. PSF is making Python 3.9.21 available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR

WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 3.9.21 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.9.21 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.9.21, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 3.9.21, 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.

(次のページに続く)

(前のページからの続き)

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 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.

(次のページに続く)

(前のページからの続き)

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 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 3.9.21 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 Licenses and Acknowledgements for Incorporated Software

This section is an incomplete, but growing list of licenses and acknowledgements for third-party software incorporated in the Python distribution.

C.3.1 Mersenne Twister

The `_random` module includes code based on a download from <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html>. The following are the verbatim comments from the original code:

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

(次のページに続く)

(前のページからの続き)

```
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

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

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

All rights reserved.

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

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

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

(次のページに続く)

(前のページからの続き)

SUCH DAMAGE.

C.3.3 Asynchronous socket services

The `asyncchat` and `asyncore` modules contain the following notice:

Copyright 1996 by Sam Rushing

All Rights Reserved

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

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

C.3.4 Cookie management

The `http.cookies` module contains the following notice:

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

(次のページに続く)

(前のページからの続き)

```
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 Execution tracing

The `trace` module contains the following notice:

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

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

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

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

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

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

C.3.6 UUencode and UUdecode functions

The `uu` module contains the following notice:

```
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
```

(次のページに続く)

(前のページからの続き)

```
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Elinghouse
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 Remote Procedure Calls

The `xmlrpc.client` module contains the following notice:

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.3.8 test_epoll

test_epoll モジュール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 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 モジュールは 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

(次のページに続く)

(前のページからの続き)

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

C.3.10 SipHash24

The file `Python/pyhash.c` contains Marek Majkowski' implementation of Dan Bernstein's SipHash24 algorithm. It contains the following note:

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

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

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

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

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

C.3.11 strtod 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
```

(次のページに続く)

(前のページからの続き)

```
* 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

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

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
 *    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
```

(次のページに続く)

(前のページからの続き)

```

*   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.
*
* 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

```

(次のページに続く)

(前のページからの続き)

```

* 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.3.13 expat

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

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

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

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

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

C.3.14 libffi

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.3.15 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) 1995-2011 Jean-loup Gailly and Mark Adler

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

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

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

Jean-loup Gailly
jloup@gzip.org

Mark Adler
madler@alumni.caltech.edu

C.3.16 cfuhash

The implementation of the hash table used by the `tracemalloc` で使用しているハッシュテーブルの実装は、`cfuhash` プロジェクトのものに基づきます:

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

(次のページに続く)

(前のページからの続き)

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

The `_decimal` module is built using an included copy of the libmpdec library unless the build is configured `--with-system-libmpdec`:

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

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

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

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)

(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.3.18 W3C C14N test suite

The C14N 2.0 test suite in the `test` package (`Lib/test/xmltestdata/c14n-20/`) was retrieved from the W3C website at <https://www.w3.org/TR/xml-c14n2-testcases/> and is distributed under the 3-clause BSD license:

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.

付録

D

COPYRIGHT

Python and this documentation is:

Copyright © 2001-2023 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

アルファベット以外

..., 11
 2to3, 11
 >>>, 11
 __future__, 17
 __slots__, 25

A

abstract base class, 11
 annotation, 11
 argument, 11
 asynchronous context manager, 12
 asynchronous generator, 12
 asynchronous generator iterator, 12
 asynchronous iterable, 12
 asynchronous iterator, 13
 attribute, 13
 awaitable, 13

B

BDFL, 13
 binary file, 13
 bytecode, 13
 bytes-like object, 13

C

callback, 13
 C-contiguous, 14
 class, 13
 class variable, 13
 coercion, 14
 complex number, 14
 context manager, 14
 context variable, 14
 contiguous, 14
 coroutine, 14
 coroutine function, 14
 CPython, 14

D

decorator, 14
 descriptor, 15
 dictionary, 15
 dictionary comprehension, 15
 dictionary view, 15
 docstring, 15
 duck-typing, 15

E

EAFFP, 15
 expression, 16
 extension module, 16

F

f-string, 16
 file object, 16
 file-like object, 16
 finder, 16
 floor division, 16
 Fortran contiguous, 14
 function, 16
 function annotation, 16

G

garbage collection, 17
 generator, 17
 generator expression, 17
 generator iterator, 17
 generic function, 17
 generic type, 17
 GIL, 17
 global interpreter lock, 17

H

hash-based pyc, 18
 hashable, 18

I

IDLE, 18
 immutable, 18
 import path, 18
 importer, 18
 importing, 18
 interactive, 18
 interpreted, 19
 interpreter shutdown, 19
 iterable, 19
 iterator, 19

K

key function, 19
 keyword argument, 20

L

lambda, 20
 LBYL, 20
 list, 20
 list comprehension, 20
 loader, 20

M

magic
 method, 20
 magic method, 20
 mapping, 20
 meta path finder, 20
 metaclass, 20
 method, 21

- magic, 20
- special, 25

method resolution order, 21

module, 21

module spec, 21

MRO, 21

mutable, 21

N

named tuple, 21

namespace, 22

namespace package, 22

nested scope, 22

new-style class, 22

O

object, 22

P

package, 22

parameter, 22

path based finder, 23

path entry, 23

path entry finder, 23

path entry hook, 23

path-like object, 23

PEP, 23

portion, 24

positional argument (位置引数), 24

provisional API, 24

provisional package, 24

Python 3000, 24

Python Enhancement Proposals

- PEP 1, 23
- PEP 238, 16
- PEP 278, 27
- PEP 302, 16, 20
- PEP 343, 14
- PEP 362, 12, 23
- PEP 411, 24
- PEP 420, 16, 22, 24
- PEP 443, 17
- PEP 451, 16
- PEP 483, 17
- PEP 484, 11, 16, 17, 26, 27
- PEP 492, 1214
- PEP 498, 16
- PEP 519, 23
- PEP 525, 12
- PEP 526, 11, 27
- PEP 585, 17
- PEP 3116, 27
- PEP 3155, 24

Pythonic, 24

Q

qualified name, 24

R

reference count, 25

regular package, 25

S

sequence, 25

set comprehension, 25

single dispatch, 25

slice, 25

special

- method, 25

special method, 25

statement, 25

T

text encoding, 26

text file, 26

triple-quoted string, 26

type, 26

type alias, 26

type hint, 26

U

universal newlines, 27

V

variable annotation, 27

virtual environment, 27

virtual machine, 27

Z

Zen of Python, 27