Installing Python Modules

リリース *3.13.5*

Guido van Rossum and the Python development team

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 3.1 3.2 3.3 3.4 第4章	… 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? よくあるインストールに関する問題	7 7 7 8 9
4.1 4.2 4.3	Linux で、システムの Python 内にインストールする pip がインストールされていない バイナリの拡張のインストール	9 9 9
付録 A 章	用語集	11
	このドキュメントについて	11 35 35
付録 B 章 B.1	このドキュメントについて Python ドキュメントへの貢献者	35
付録 B 章 B.1 付録 C 章 C.1 C.2 C.3	アython ドキュメントへの貢献者	35 35 37 37 38
付録 B 章 B.1 付録 C 章 C.1 C.2 C.3	R このドキュメントについて Python ドキュメントへの貢献者	35 35 37 37 38 43

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、共有するためのガイドについては Python packaging user guide を参照してください。

1 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

ONE

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。
- Python Package 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 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 の使用が今は推奨されています。

→ 参考

Python Packaging User Guide: Creating and using virtual environments

第

TWO

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 Package Index からインストールします。

python -m pip install SomePackage

① 注釈

POSIX ユーザ (macOS 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。

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

第

THREE

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 を並行して使う には?

Linux や macOS、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

FOUR

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 macOS についての wheels の公開が Python Package 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

Python Packaging User Guide: Binary Extensions

付録

Α

用語集

>>>

対

話型 シェルにおけるデフォルトの Python プロンプトです。インタープリターで対話的に実行されるコード例でよく見られます。

. . .

次

のものが考えられます:

- 対話型 (*interactive*) 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。

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。

Annotations of local variables cannot be accessed at runtime, but annotations of global variables, class attributes, and functions are stored in the __annotations__ special attribute of modules, classes, and functions, respectively.

See variable annotation, function annotation, PEP 484 and PEP 526, which describe this functionality. Also see annotations-how to for best practices on working with annotations.

引数 (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?、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 *asynchronous iterator* で、__anext__() メソッドを使って呼ばれると awaitable オブジェクトを返します。この awaitable オブジェクトは、次の yield 式まで非同期ジェネレータ関数の本体を実行します。

Each yield temporarily suspends processing, remembering the execution state (including local variables and pending try-statements). When the *asynchronous generator iterator* effectively resumes with another awaitable returned by <code>__anext__()</code>, it picks up where it left off. See **PEP 492** and **PEP 525**.

asynchronous iterable

12

__(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。

属性

(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通常参照される値です。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。

オブジェクトには、identifiers で定義される識別子ではない名前の属性を与えることができます。たとえば setattr()を使い、オブジェクトがそれを許可している場合に行えます。このような属性はドット表記式ではアクセスできず、代わりに getattr()を使って取る必要があります。

awaitable

(待機可能) await 式で使用する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

悲深き終身独裁者 (Benevolent Dictator For Life) の略です。Python の作者、Guido van Rossum の

慈

binary file

ことです。

(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。

borrowed reference

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

Calling Py_INCREF() on the *borrowed reference* is recommended to convert it to a *strong reference* in-place, except when the object cannot be destroyed before the last usage of the borrowed reference. The Py_NewRef() function can be used to create a new *strong reference*.

bytes-like object

bufferobjects をサポートしていて、C 言語の意味で 連続した バッファーを提供可能なオブジェクト。bytes, bytearray, 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ーオブジェクトには、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。

callable

callable is an object that can be called, possibly with a set of arguments (see *argument*), with the following syntax:

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

A function, and by extension a method, is a callable. An instance of a class that implements the __call__() method is also a callable.

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

クラス

(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しに) 変更されることを目的としている変数です。

closure variable A

free variable referenced from a nested scope that is defined in an outer scope rather than being resolved at runtime from the globals or builtin namespaces. May be explicitly defined with the nonlocal keyword to allow write access, or implicitly defined if the variable is only being read.

For example, in the inner function in the following code, both x and print are *free variables*, but only x is a *closure variable*:

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

(前のページからの続き)

print(x)
return inner

Due to the codeobject.co_freevars attribute (which, despite its name, only includes the names of closure variables rather than listing all referenced free variables), the more general *free variable* term is sometimes used even when the intended meaning is to refer specifically to closure variables.

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

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

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

コンテキスト管理プロトコル

The __enter__() and __exit__() methods called by the with statement. See PEP 343.

context manager

An object which implements the *context management protocol* and controls the environment seen in a with statement. See PEP 343.

context variable A

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

contiguous

(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。

コルーチン

(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有る場合に利用されます。

current context

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

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)

@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

Any object which defines the methods $__get__()$, $__set__()$, or $__delete__()$. When a class attribute is a descriptor, its special binding behavior is triggered upon attribute lookup. Normally, using a.b to get, set or delete an attribute looks up the object named b in the class dictionary for a, but if b is a descriptor, the respective descriptor method gets called. Understanding descriptors is a key to a deep understanding of Python because they are the basis for many features including functions, methods, properties, class methods, static methods, and reference to super classes.

デスクリプタのメソッドに関しての詳細は、descriptors や Descriptor How To Guide を参照してください。

dictionary

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

dictionary comprehension

(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 = $\{n: n ** 2 \text{ for } n \text{ in range}(10)\}$ とすると、キー n ** 2 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

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

duck-typing 5

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 $\mathbf x$ 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

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

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。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。

filesystem encoding and error handler

Encoding and error handler used by Python to decode bytes from the operating system and encode Unicode to the operating system.

ファイルシステムのエンコーディングでは、すべてが 128 バイト以下に正常にデコードされることが 保証されなくてはなりません。ファイルシステムのエンコーディングでこれが保証されなかった場合 は、API 関数が UnicodeError を送出することがあります。

The sys.getfilesystemencoding() and sys.getfilesystemencodeerrors() functions can be used to get the filesystem encoding and error handler.

The *filesystem encoding and error handler* are configured at Python startup by the PyConfig_Read() function: see filesystem_encoding and filesystem_errors members of PyConfig.

See also the locale encoding.

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。

There are two types of finder: *meta path finders* for use with sys.meta_path, and *path entry finders* for use with sys.path_hooks.

See finders-and-loaders and importlib for much more detail.

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。

free threading

A threading model where multiple threads can run Python bytecode simultaneously within the

same interpreter. This is in contrast to the *global interpreter lock* which allows only one thread to execute Python bytecode at a time. See **PEP 703**.

free variable

Formally, as defined in the language execution model, a free variable is any variable used in a namespace which is not a local variable in that namespace. See *closure variable* for an example. Pragmatically, due to the name of the <code>codeobject.co_freevars</code> attribute, the term is also sometimes used as a synonym for *closure variable*.

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。

function annotation

(関数アノテーション) 関数のパラメータや返り値の 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 型の返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*, **PEP 484**, を参照してください。また、アノテーションを利用するベストプラクティスとして annotations-howto も参照してください。

___future__

from __future__ import <feature> という future 文 は、コンパイラーに将来の Python リリースで標準となる構文や意味を使用して現在のモジュールをコンパイルするよう指示します。__future__ モジュールでは、feature のとりうる値をドキュメント化しています。このモジュールをインポートし、その変数を評価することで、新機能が最初に言語に追加されたのはいつかや、いつデフォルトになるか(またはなったか)を見ることができます:

```
>>> 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) 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。

ジェネレータ

(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。

Each yield temporarily suspends processing, remembering the execution state (including local variables and pending try-statements). When the *generator iterator* resumes, it picks up where it left off (in contrast to functions which start fresh on every invocation).

generator expression

An *expression* that returns an *iterator*. It looks like a normal expression followed by a for clause defining a loop variable, range, and an optional if clause. The combined expression generates values for an enclosing function:

```
>>> 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

Α

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。

As of Python 3.13, the GIL can be disabled using the --disable-gil build configuration. After building Python with this option, code must be run with -X gil=0 or after setting the PYTHON_GIL=0 environment variable. This feature enables improved performance for

multi-threaded applications and makes it easier to use multi-core CPUs efficiently. For more details, see PEP 703.

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。pyc-invalidation を参照してください。

hashable

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

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) 及び学習環境 (Learning 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。

永続オブジェクト (immortal)

Immortal objects are a CPython implementation detail introduced in PEP 683.

If an object is immortal, its *reference count* is never modified, and therefore it is never deallocated while the interpreter is running. For example, True and None are immortal in CPython.

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 #

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

Python has an interactive interpreter which means you can enter statements and expressions at the interpreter prompt, immediately execute them and see their results. Just launch python with no arguments (possibly by selecting it from your computer's main menu). It is a very powerful way to test out new ideas or inspect modules and packages (remember help(x)). For more on interactive mode, see tut-interac.

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

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

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

iterator

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 を送出します。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します。

詳細な情報は typeiter にあります。

CPython 実装の詳細: CPython does not consistently apply the requirement that an iterator define __iter__(). And also please note that the free-threading CPython does not guarantee the thread-safety of iterator operations.

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.attrgetter(), operator.itemgetter(), operator.methodcaller() の 3 つのキー関数コンストラクタがあります。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 1 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。

lexical analyzer

Formal name for the tokenizer; see token.

list A

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

list comprehension

(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

An object that loads a module. It must define the exec_module() and create_module() methods to implement the Loader interface. A loader is typically returned by a *finder*. See also:

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

ロケールエンコーディング

On Unix, it is the encoding of the LC_CTYPE locale. It can be set with locale. setlocale(locale.LC_CTYPE, new_locale).

On Windows, it is the ANSI code page (ex: "cp1252").

On Android and VxWorks, Python uses "utf-8" as the locale encoding.

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

See also the filesystem encoding and error handler.

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、collections.abc.Mapping か collections.abc.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取って、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。

メソッド

(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

Method Resolution Order is the order in which base classes are searched for a member during lookup. See python_2.3_mro for details of the algorithm used by the Python interpreter since the 2.3 release.

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

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。

See also module-specs.

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ってのアクセスもできる任意の型やクラスに応用されています。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() の返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
```

モ

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

namespace

(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

A package which serves only as a container for subpackages. Namespace packages may have no physical representation, and specifically are not like a regular package because they have no <code>__init__.py</code> file.

Namespace packages allow several individually installable packages to have a common parent package. Otherwise, it is recommended to use a *regular package*.

For more information, see PEP 420 and reference-namespace-package.

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

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

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。

optimized scope

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

a known set of local and nonlocal variable names are restricted to optimized scopes.

package

(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ですか?、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

Α

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

path based finder

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。

位置引数 (positional argument)

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) が行われえます。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:

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 (次のページに続く)
```

(前のページからの続き)

```
>>> 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。 **永続** であり、参照カウントが決して変更されないために割り当てが解除されないオブジェクトもあります。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。プログラマーは、任意のオブジェクトの参照カウントを知るために sys.getrefcount() 関数を呼び出すことが出来ます。

In *CPython*, reference counts are not considered to be stable or well-defined values; the number of references to an object, and how that number is affected by Python code, may be different between versions.

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。

REPL

"read – eval – print loop" の頭字語で、**対話型** インタープリターシェルの別名。

___slots___

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

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

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

set comprehension

(集合内包表記) iterable 内の全てあるいは一部の要素を処理して、その結果からなる集合を返すコンパクトな書き方です。results = {c for c in 'abracadabra' if c not in 'abc'} とすると、{'r', 'd'} という文字列の辞書を生成します。comprehensions を参照してください。

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。

soft deprecated A

soft deprecated API should not be used in new code, but it is safe for already existing code to use it. The API remains documented and tested, but will not be enhanced further.

Soft deprecation, unlike normal deprecation, does not plan on removing the API and will not emit warnings.

See PEP 387: Soft Deprecation.

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。

standard library

The collection of *packages*, *modules* and *extension modules* distributed as a part of the official Python interpreter package. The exact membership of the collection may vary based on platform, available system libraries, or other criteria. Documentation can be found at library-index.

See also sys.stdlib_module_names for a list of all possible standard library module names.

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。

static type checker

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

stdlib

An abbreviation of *standard library*.

strong reference In

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

The Py_NewRef() function can be used to create a strong reference to an object. Usually, the Py_DECREF() function must be called on the strong reference before exiting the scope of the strong reference, to avoid leaking one reference.

See also borrowed reference.

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。

トークン

small unit of source code, generated by the lexical analyzer (also called the *tokenizer*). Names, numbers, strings, operators, newlines and similar are represented by tokens.

The tokenize module exposes Python's lexical analyzer. The token module contains information on the various types of tokens.

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ができるので、ドキュメンテーション文字列を書く時に特に便利です。

type

The type of a Python object determines what kind of object it is; every object has a type. An object's type is accessible as its __class__ attribute or can be retrieved with type(obj).

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:

これは次のようにより読みやすくできます:

```
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返り値の期待される型を指定する annotation です。

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

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

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。

機能の説明がある function annotation, PEP 484, PEP 526 を参照してください。また、アノテーションを利用するベストプラクティスとして annotations-howto 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。

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。

34 付録 A 章 用語集

付録

В

このドキュメントについて

Python のドキュメントは、Sphinx を使って、reStructuredText のソースから生成されています。Sphinx は もともと Python のために作られ、現在は独立したプロジェクトとして保守されています。

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 ページをご覧下さい。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 プロジェクトは、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 ドキュメントへの貢献者

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 ${
m Misc/ACKS}$ 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ト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 was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see https://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see https://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations, which became Zope Corporation. In 2001, the Python Software Foundation (PSF, see https://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see https://opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

リリース	ベース	西暦年	権利	GPL-compatible? (1)
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	${\bf Be Open.com}$	no
1.6.1	1.6	2001	CNRI	yes (2)
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

1 注釈

- (1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.
- (2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

Guido の指示の下、これらのリリースを可能にして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 Python Software Foundation License Version 2.

Starting with Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF License Version 2 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 PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

- 1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") in source or binary form and its associated documentation.
- 2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2024 Python Software Foundation; All Rights Reserved" are retained in Python alone or in any derivative version prepared by Licensee.
- 3. In the event Licensee prepares a derivative work that is based on or incorporates Python or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python.

- 4. PSF is making Python available to Licensee on an "AS IS" basis.

 PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF

 EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR

 WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE

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

C.2.2 BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

- 1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").
- 2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.
- 3. BeOpen is making the Software available to Licensee on an "AS IS" basis.
 BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF

EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

- 4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
- 5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
- 6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at http://www.pythonlabs.com/logos.html may be used according to the permissions granted on that web page.
- 7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.3 CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

- 1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
- 2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement,

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

- 3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
- 4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
- 5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
- 6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
- 7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions.

 Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
- 8. By clicking on the "ACCEPT" button where indicated, or by copying, installing

or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.4 CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

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

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

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

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 C extension underlying 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 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, https://www.wide.ad.jp/.

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

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

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

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

C.3.3 Asynchronous socket services

The test.support.asynchat and test.support.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 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 codec 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 both that copyright notice and this permission notice appear in supporting documentation, and that the name of Lance Ellinghouse not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

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

Modified by Jack Jansen, CWI, July 1995:

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

C.3.7 XML 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 MERCHANT-ABILITY AND FITNESS. IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3.8 test_epoll

The test.test_epoll module contains the following notice:

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

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

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

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

C.3.9 Select 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 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 https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c. The original file, as retrieved on March 16, 2009, contains the following copyright and licensing notice:

C.3.12 OpenSSL

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

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

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

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

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

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

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

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

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

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

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications

represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

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

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

- 2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
- 3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct

or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

- 4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
 - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
 - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use,

reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

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

License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

C.3.13 expat

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

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

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

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

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

C.3.14 libffi

The _ctypes C extension underlying the ctypes module 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, 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 Mark Adler

jloup@gzip.org 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 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 C extension underlying 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) 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.

C.3.19 mimalloc

MIT License:

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

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

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

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

C.3.20 asyncio

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

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

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

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

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

C.3.21 Global Unbounded Sequences (GUS)

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

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

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

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

付録

D

COPYRIGHT

Python and this documentation is:

Copyright © 2001-2024 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	current context, 16	interpreted, 22
, 11	,	interpreter shutdown, 22
>>>, 11	D	iterable, 22
•		iterator, 22
future, 19 slots, 30	decorator, 16	,
S10tS, 50 クラス, 14	descriptor, 16	K
ファス, 14 コルーチン, 15	dictionary, 16	IX.
	dictionary comprehension, 17	key function, 23
コンテキスト管理プロトコル, 15	dictionary view, 17	keyword argument, 23
ジェネレータ, 19	docstring, 17	
トークン、32	duck-typing, 17	
メソッド, 25		–
magic, 24	E	lambda, 23
特殊, 31	-	LBYL, 23
ロケールエンコーディング, 24	EAFP, 17	lexical analyzer, 23
位置引数 (positional argument), 28	expression, 17	list, 23
特殊	extension module, 17	list comprehension, 24
メソッド, 31		loader, 24
環境変数	F	
PYTHON_GIL, 20	'	M
関数, 19	f-string, 17	171
	file object, 18	magic
A	file-like object, 18	メソッド, 24
	filesystem encoding and error	magic method, 24
abstract base class, 11	handler, 18	mapping, 24
annotation, 11	finder, 18	meta path finder, 24
asynchronous context manager, 12	floor division, 18	metaclass, 24
asynchronous generator, 12	Fortran contiguous, 15	method resolution order, 25
asynchronous generator iterator, 12	free threading, 18	module, 25
asynchronous iterable, 12	free variable, 19	module spec, 25
asynchronous iterator, 13	function annotation, 19	MRO, 25
awaitable, 13	,	mutable, 25
	G	,
В	d	N
	garbage collection, 19	IN
BDFL, 13	generator expression, 20	named tuple, 25
binary file, 13	generator iterator, 20	namespace, 26
borrowed reference, 13	generic function, 20	namespace package, 26
bytecode, 14	generic type, 20	nested scope, 26
bytes-like object, 13	GIL, 20	new-style class, 26
	global interpreter lock, 20	
C		0
11.11.14	H	
callable, 14	''	object, 26
callback, 14	hash-based pyc, 21	optimized scope, 26
C-contiguous, 15	hashable, 21	
class variable, 14		P
closure variable, 14		•
complex number, 15		package, 27
context, 15	IDLE, 21	parameter, 27
context manager, 15	immutable, 21	path based finder, 28
context variable, 15	import path, 21	path entry, 27
contiguous, 15	importer, 21	path entry finder, 28
coroutine function, 16	importing, 21	path entry hook, 28
CPython, 16	interactive, 22	path-like object, 28

```
Т
PEP, 28
                                                         PEP 3155, 29
\mathtt{portion},\, \textcolor{red}{\mathbf{28}}
                                                    PYTHON_GIL, 20
                                                                                                         text encoding, 32
provisional API, 28
                                                    Pythonic, 29
                                                                                                         text file, 32
{\tt provisional\ package},\, {\tt 29}
                                                                                                         {\tt triple-quoted\ string},\, {\tt 32}
Python 3000, 29
                                                    Q
                                                                                                         type, 32
Python Enhancement Proposals
                                                                                                         type alias, 32
                                                    {\tt qualified\ name,\ 29}
    PEP 1, 28
                                                                                                         \mathtt{type}\ \mathtt{hint},\, \color{red} \mathbf{33}
    PEP 238, 18
    PEP 278, 33
                                                    R
    PEP 302, 24
                                                    reference count, 30
    PEP 343, 15
                                                                                                         universal newlines, 33
                                                    regular package, 30
    \mathtt{PEP} \ \ \mathbf{362}, \ 12, \ 27
                                                    REPL, 30
    PEP 411, 29
    PEP 420, 26, 28
                                                    S
                                                                                                         variable annotation, {\bf 33}
    PEP 443, 20
                                                                                                         virtual environment, 33
    PEP 483, 20
                                                    sequence, 30
                                                                                                         virtual machine, 34
    \mathtt{PEP}\ 484,\,11,\,19,\,20,\,33
                                                    set comprehension, 30
                                                                                                         属性, 13
                                                    single dispatch, 30
    PEP 492, 12, 13, 16
                                                                                                         引数 (argument), 11
    PEP 498, 18
                                                    {\tt slice},\, {\color{red} {\bf 31}}
    PEP 519, 28
                                                    soft deprecated, 31
                                                                                                         W
    PEP 525, 12
                                                    special method, 31
    PEP 526, 11, 33
                                                    standard library, \mathbf{31}
                                                                                                         永続オブジェクト (immortal), 21
    PEP 585, 20
                                                    statement, 31
    PEP 683, 21
                                                    static type checker, \mathbf{31}
                                                                                                         Z
    \mathtt{PEP} \ 703, \ 19, \ 21
                                                    {\tt stdlib},\, {\color{red} {\bf 31}}
    PEP 3116, 33
                                                    strong reference, \mathbf{31}
                                                                                                         Zen of Python, 34
```

66 索引