Installing Python Modules

릴리스 *3.13.1*

Guido van Rossum and the Python development team

1월 05, 2025

Contents

1 핵심 용어 **3**

2 기본 사용법 **5**

3 어떻게 …? **7**
3.1 … 파이썬 3.4 이전 버전의 파이썬에서 `pip` 를 설치합니까? 7
3.2 … 현재 사용자만을 위해 패키지를 설치합니까? 7
3.3 … 과학 계산용 파이썬 패키지를 설치합니까? 7
3.4 … 병렬로 설치된 여러 버전의 파이썬으로 작업합니까? 7

4 흔히 만나는 설치 문제 **9**
4.1 리눅스에 시스템 파이썬 설치하기 9
4.2 Pip가 설치되지 않았습니다. 9
4.3 바이너리 확장 설치하기 9

A 용어집 **11**

B About this documentation **29**
B.1 Contributors to the Python documentation 29

C 역사와 라이센스 **31**
C.1 소프트웨어의 역사 31
C.2 파이썬에 액세스하거나 사용하기 위한 이용 약관 32
C.3 포함된 소프트웨어에 대한 라이센스 및 승인 35

D 저작권 **51**

색인 **53**

전자 우편
distutils-sig@python.org

널리 사용되는 공개 소스 개발 프로젝트로서, 파이썬에는 적극적으로 지원하는 기여자와 사용자의 커뮤니티가 있어, 자신들의 소프트웨어를 공개 소스 라이센스 계약에 따라 다른 파이썬 개발자가 사용할 수 있도록 합니다.

이를 통해 파이썬 사용자는 효과적으로 공유하고 공동 작업할 수 있으며, 다른 사람들이 공통 (심지어 때로는 희귀한) 문제에 대해 이미 만든 솔루션의 이점을 누릴 수 있을 뿐 아니라, 자체 솔루션을 공동 풀에 잠재적으로 제공할 수 있습니다.

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

> **참고**
>
> 기업 및 기타 기관 사용자의 경우, 많은 조직에서 공개 소스 소프트웨어를 사용하고 공헌하는 데 대한 자체 정책을 시행하고 있음을 알고 있어야 합니다. 파이썬과 함께 제공되는 배포 및 설치 도구를 사용할 때 이러한 정책을 고려하십시오.

CHAPTER 1

핵심 용어

- `pip` 는 선호되는 설치 프로그램입니다. 파이썬 3.4부터, 기본적으로 파이썬 바이너리 설치 프로그램에 포함되어 있습니다.
- 가상 환경 은 패키지가 시스템 전체에 설치되는 것이 아니라, 특정 응용 프로그램에 사용되도록 설치될 수 있도록 하는 반 격리 된 파이썬 환경입니다.
- `venv` 는 가상 환경을 만들기 위한 표준 도구이며, 파이썬 3.3 이후로 파이썬의 일부입니다. 파이썬 3.4부터, 생성되는 모든 가상 환경에 기본적으로 `pip` 를 설치합니다.
- `virtualenv` 는 `venv` 에 대한 제삼자 대안(그리고 전임자) 입니다. `venv` 를 아예 제공하지 않거나, 생성되는 환경에 `pip` 를 자동으로 설치할 수 없는, 3.4 이전의 파이썬 버전에서 가상 환경을 사용할 수 있도록 합니다.
- The Python Package Index is a public repository of open source licensed packages made available for use by other Python users.
- the Python Packaging Authority is the group of developers and documentation authors responsible for the maintenance and evolution of the standard packaging tools and the associated metadata and file format standards. They maintain a variety of tools, documentation, and issue trackers on GitHub.
- `distutils` 는 1998년에 파이썬 표준 라이브러리에 처음 추가된 최초의 빌드 및 배포 시스템입니다. `distutils` 의 직접 사용이 단계적으로 폐지되고 있지만, 여전히 현재의 패키징 및 배포 기반 시설의 기초를 이루고 있고, 표준 라이브러리의 일부로 남아있을 뿐만 아니라, 그 이름은 다른 방식으로도 (가령 파이썬 패키징 표준 개발을 조정하는 데 사용되는 메일링 리스트의 이름으로) 남아있습니다.

버전 3.5에서 변경: 이제 가상 환경을 만드는데 `venv` 사용을 권장합니다.

더 보기

파이썬 패키징 사용자 지침서: 가상 환경 생성과 사용

CHAPTER 2

기본 사용법

표준 패키징 도구는 모두 명령 줄에서 사용하도록 설계되었습니다.

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

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

> **참고**
>
> For POSIX users (including macOS and Linux users), the examples in this guide assume the use of a *virtual environment*.
>
> 윈도우 사용자의 경우, 이 지침서의 예제는 파이썬을 설치할 때 시스템 PATH 환경 변수를 조정하는 옵션이 선택되었다고 가정합니다.

정확한 버전이나 최소 버전을 명령 줄에서 직접 지정할 수도 있습니다. >, < 와 같은 비교 연산자나 셸에서 해석되는 다른 특수 문자를 사용할 때는, 패키지 이름과 버전을 큰따옴표로 묶어야 합니다:

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

일반적으로, 적절한 모듈이 이미 설치되어있으면, 다시 설치하려고 해도 아무런 효과가 없습니다. 기존 모듈을 업그레이드하려면 명시적으로 요청해야 합니다:

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

`pip` 와 그 기능에 관한 더 많은 정보와 자원은 파이썬 패키징 사용자 지침서 에서 찾을 수 있습니다.

가상 환경의 생성은 `venv` 모듈을 통해 수행됩니다. 활성 가상 환경에 패키지를 설치하려면 위에 표시된 명령을 사용합니다.

> **더 보기**
>
> 파이썬 패키징 사용자 지침서: 파이썬 배포 패키지 설치하기

CHAPTER 3

어떻게 …?

이것들은 몇 가지 일반적인 작업에 대한 빠른 답변 또는 링크입니다.

3.1 … 파이썬 3.4 이전 버전의 파이썬에서 pip 를 설치합니까?

파이썬은 pip 를 파이썬 3.4에 와서야 포함하기 시작했습니다. 이전 버전의 경우, pip 는 파이썬 패키징 사용자 지침서에 설명된 대로 "부트스트랩(bootstrapped)" 될 필요가 있습니다.

> **더 보기**
>
> 파이썬 패키징 사용자 지침서: 패키지 설치 요구 사항

3.2 … 현재 사용자만을 위해 패키지를 설치합니까?

--user 옵션을 python -m pip install 에 넘겨주면 시스템의 모든 사용자가 아닌 현재 사용자를 위해서만 패키지를 설치합니다.

3.3 … 과학 계산용 파이썬 패키지를 설치합니까?

많은 과학 계산용 파이썬 패키지는 복잡한 바이너리 의존성을 가지고 있고, 현재 pip 를 사용하여 직접 설치하기가 쉽지 않습니다. 이 시점에서, 사용자가 이 패키지들을 pip 로 설치하려고 시도하는 것보다 다른 방법 으로 설치하는 것이 더 쉬운 경우가 종종 있습니다.

> **더 보기**
>
> 파이썬 패키징 사용자 지침서: 과학 계산용 패키지 설치하기

3.4 … 병렬로 설치된 여러 버전의 파이썬으로 작업합니까?

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

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

적절히 버전이 지정된 `pip` 명령도 제공될 수 있습니다.

윈도우에서는 `py` 파이썬 실행기를 `-m` 스위치와 조합해서 사용하십시오:

```
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
```

CHAPTER 4

흔히 만나는 설치 문제

4.1 리눅스에 시스템 파이썬 설치하기

리눅스 시스템에서, 보통 파이썬 설치가 배포 일부로 포함됩니다. 이 파이썬 설치에 설치하려면 시스템에 대한 루트 액세스가 필요하며, `pip` 를 사용하여 구성 요소가 예기치 않게 업그레이드되면 시스템 패키지 관리자와 시스템의 다른 구성 요소의 작동을 방해할 수 있습니다.

이러한 시스템에서는, `pip` 로 패키지를 설치할 때 가상 환경이나 사용자별 설치를 사용하는 것이 더 좋습니다.

4.2 Pip가 설치되지 않았습니다.

`pip` 가 기본적으로 설치되지 않았을 수도 있습니다. 한가지 가능한 해결법은 다음과 같습니다:

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

There are also additional resources for installing pip.

4.3 바이너리 확장 설치하기

파이썬은 일반적으로 소스 기반 배포에 크게 의존하고 있으며, 최종 사용자가 설치 프로세스의 일부로 소스로부터 확장 모듈을 컴파일할 것으로 기대합니다.

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

`wheel` 파일로 아직 사용할 수 없는 과학 계산용 소프트웨어 설치를 위한 어떤 해결책들은 로컬에서 빌드할 필요 없이 다른 바이너리 확장을 얻는 데 도움이 될 수 있습니다.

더 보기

파이썬 패키징 사용자 지침서: 바이너리 확장

APPENDIX A

용어집

>>>

The default Python prompt of the *interactive* shell. Often seen for code examples which can be executed interactively in the interpreter.

...

다음과 같은 것들을 가리킬 수 있습니다:

- The default Python prompt of the *interactive* shell when entering the code for an indented code block, when within a pair of matching left and right delimiters (parentheses, square brackets, curly braces or triple quotes), or after specifying a decorator.
- `Ellipsis` 내장 상수.

abstract base class (추상 베이스 클래스)

추상 베이스 클래스는 `hasattr()` 같은 다른 테크닉들이 불편하거나 미묘하게 잘못된 (예를 들어, 매직 메서드) 경우, 인터페이스를 정의하는 방법을 제공함으로써 덕 타이핑 을 보완합니다. ABC는 가상 서브 클래스를 도입하는데, 클래스를 계승하지 않으면서도 `isinstance()` 와 `issubclass()` 에 의해 감지될 수 있는 클래스들입니다; `abc` 모듈 설명서를 보세요. 파이썬에는 많은 내장 ABC 들이 따라오는데 다음과 같은 것들이 있습니다: 자료 구조 (`collections.abc` 모듈에서), 숫자 (`numbers` 모듈에서), 스트림 (`io` 모듈에서), 임포트 파인더와 로더 (`importlib.abc` 모듈에서). `abc` 모듈을 사용해서 자신만의 ABC를 만들 수도 있습니다.

annotation (어노테이션)

관습에 따라 형 힌트 로 사용되는 변수, 클래스 어트리뷰트 또는 함수 매개변수 나 반환 값과 연결된 레이블입니다.

지역 변수의 어노테이션은 실행 시간에 액세스할 수 없지만, 전역 변수, 클래스 속성 및 함수의 어노테이션은 각각 모듈, 클래스, 함수의 `__annotations__` 특수 어트리뷰트에 저장됩니다.

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

argument (인자)

함수를 호출할 때 함수 (또는 메서드) 로 전달되는 값. 두 종류의 인자가 있습니다:

- 키워드 인자 (*keyword argument*): 함수 호출 때 식별자가 앞에 붙은 인자 (예를 들어, `name=`) 또는 `**` 를 앞에 붙인 딕셔너리로 전달되는 인자. 예를 들어, 다음과 같은 `complex()` 호출에서 `3` 과 `5` 는 모두 키워드 인자입니다:

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

- 위치 인자 (*positional argument*): 키워드 인자가 아닌 인자. 위치 인자들은 인자 목록의 처음에 나오거나 이터러블 의 앞에 `*` 를 붙여 전달할 수 있습니다. 예를 들어, 다음과 같은 호출에서 `3` 과 `5` 는 모두 위치 인자입니다.

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

인자는 함수 바디의 이름 붙은 지역 변수에 대입됩니다. 이 대입에 적용되는 규칙들에 대해서는 calls 절을 보세요. 문법적으로, 어떤 표현식이건 인자로 사용될 수 있습니다; 구해진 값이 지역 변수에 대입됩니다.

용어집의 매개변수 항목과 FAQ 질문 인자와 매개변수의 차이 와 **PEP 362**도 보세요.

asynchronous context manager (비동기 컨텍스트 관리자)
An object which controls the environment seen in an `async with` statement by defining `__aenter__()` and `__aexit__()` methods. Introduced by **PEP 492**.

asynchronous generator (비동기 제너레이터)
비동기 제너레이터 이터레이터 를 돌려주는 함수. `async def` 로 정의되는 코루틴 함수처럼 보이는데, `async for` 루프가 사용할 수 있는 일련의 값들을 만드는 `yield` 표현식을 포함한다는 점이 다릅니다.

보통 비동기 제너레이터 함수를 가리키지만, 어떤 문맥에서는 비동기 제너레이터 이터레이터 를 가리킵니다. 의도하는 의미가 명확하지 않은 경우는, 완전한 용어를 써서 모호함을 없앱니다.

비동기 제너레이터 함수는 `await` 표현식과, `async for` 문과, `async with` 문을 포함할 수 있습니다.

asynchronous generator iterator (비동기 제너레이터 이터레이터)
비동기 제너레이터 함수가 만드는 객체.

This is an *asynchronous iterator* which when called using the `__anext__()` method returns an awaitable object which will execute the body of the asynchronous generator function until the next `yield` expression.

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

asynchronous iterable (비동기 이터러블)
An object, that can be used in an `async for` statement. Must return an *asynchronous iterator* from its `__aiter__()` method. Introduced by **PEP 492**.

asynchronous iterator (비동기 이터레이터)
An object that implements the `__aiter__()` and `__anext__()` methods. `__anext__()` must return an *awaitable* object. `async for` resolves the awaitables returned by an asynchronous iterator' s `__anext__()` method until it raises a `StopAsyncIteration` exception. Introduced by **PEP 492**.

attribute (어트리뷰트)
A value associated with an object which is usually referenced by name using dotted expressions. For example, if an object *o* has an attribute *a* it would be referenced as *o.a*.

It is possible to give an object an attribute whose name is not an identifier as defined by identifiers, for example using `setattr()`, if the object allows it. Such an attribute will not be accessible using a dotted expression, and would instead need to be retrieved with `getattr()`.

awaitable (어웨이터블)
An object that can be used in an `await` expression. Can be a *coroutine* or an object with an `__await__()` method. See also **PEP 492**.

BDFL
자비로운 종신 독재자 (Benevolent Dictator For Life), 즉 Guido van Rossum, 파이썬의 창시자.

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

`str` 객체를 읽고 쓸 수 있는 파일 객체에 대해서는 텍스트 파일 도 참조하세요.

borrowed reference

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

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 (바이트열류 객체)

bufferobjects 를 지원하고 C-연속 버퍼를 익스포트 할 수 있습니다. 여러 공통 `memoryview` 객체들은 물론이고 `bytes`, `bytearray`, `array.array` 객체들을 포함합니다. 바이트열류 객체들은 바이너리 데이터를 다루는 여러 가지 연산들에 사용될 수 있습니다; 압축, 바이너리 파일로 저장, 소켓을 통한 전송 같은 것들이 있습니다.

어떤 연산들은 바이너리 데이터가 가변적일 필요가 있습니다. 이런 경우에 설명서는 종종 "읽고-쓰기 바이트열류 객체"라고 표현합니다. 가변 버퍼 객체의 예로는 `bytearray` 와 `bytearray` 의 `memoryview` 가 있습니다. 다른 연산들은 바이너리 데이터가 불변 객체 ("읽기 전용 바이트열류 객체")에 저장되도록 요구합니다; 이런 것들의 예로는 `bytes`와 `bytes` 객체의 `memoryview` 가 있습니다.

bytecode (바이트 코드)

파이썬 소스 코드는 바이트 코드로 컴파일되는데, CPython 인터프리터에서 파이썬 프로그램의 내부 표현입니다. 바이트 코드는 `.pyc` 파일에 캐시 되어, 같은 파일을 두 번째 실행할 때 더 빨라지게 만듭니다 (소스에서 바이트 코드로의 재컴파일을 피할 수 있습니다). 이 "중간 언어" 는 각 바이트 코드에 대응하는 기계를 실행하는 가상 기계 에서 실행된다고 말합니다. 바이트 코드는 서로 다른 파이썬 가상 기계에서 작동할 것으로 기대하지도, 파이썬 배포 간에 안정적이지도 않다는 것에 주의해야 합니다.

바이트 코드 명령어들의 목록은 dis 모듈 설명서에 나옵니다.

callable

A 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 (콜백)

인자로 전달되는 미래의 어느 시점에서 실행될 서브 루틴 함수.

class (클래스)

사용자 정의 객체들을 만들기 위한 주형. 클래스 정의는 보통 클래스의 인스턴스를 대상으로 연산하는 메서드 정의들을 포함합니다.

class variable (클래스 변수)

클래스에서 정의되고 클래스 수준 (즉, 클래스의 인스턴스에서가 아니라) 에서만 수정되는 변수.

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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
        x += 1
        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 (복소수)
익숙한 실수 시스템의 확장인데, 모든 숫자가 실수부와 허수부의 합으로 표현됩니다. 허수부는 실수에 허수 단위(`-1`의 제곱근)를 곱한 것인데, 종종 수학에서는 `i`로, 공학에서는 `j`로 표기합니다. 파이썬은 후자의 표기법을 쓰는 복소수를 기본 지원합니다; 허수부는 `j` 접미사를 붙여서 표기합니다, 예를 들어, `3+1j`. `math` 모듈의 복소수 버전이 필요하면, `cmath`를 사용합니다. 복소수의 활용은 꽤 수준 높은 수학적 기능입니다. 필요하다고 느끼지 못한다면, 거의 확실히 무시해도 좋습니다.

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*.

context management protocol
The `__enter__()` and `__exit__()` methods called by the `with` statement. See **PEP 343**.

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

context variable (컨텍스트 변수)
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 (연속)
버퍼는 정확히 *C*-연속(*C-contiguous*)이거나 포트란 연속(*Fortran contiguous*)일 때 연속이라고 여겨집니다. 영차원 버퍼는 C-연속이면서 포트란 연속입니다. 일차원 배열에서, 항목들은 서로에 인접하고, 0에서 시작하는 오름차순 인덱스의 순서대로 메모리에 배치되어야 합니다. 다차원 C-연속 배열에서, 메모리 주소의 순서대로 항목들을 방문할 때 마지막 인덱스가 가장 빨리 변합니다. 하지만, 포트란 연속 배열에서는, 첫 번째 인덱스가 가장 빨리 변합니다.

coroutine (코루틴)
코루틴은 서브루틴의 더 일반화된 형태입니다. 서브루틴은 한 지점에서 진입하고 다른 지점에서 탈출합니다. 코루틴은 여러 다른 지점에서 진입하고, 탈출하고, 재개할 수 있습니다. 이것들은 `async def` 문으로 구현할 수 있습니다. **PEP 492**를 보세요.

coroutine function (코루틴 함수)
코루틴 객체를 돌려주는 함수. 코루틴 함수는 `async def` 문으로 정의될 수 있고, `await` 와 `async for`와 `async with` 키워드를 포함할 수 있습니다. 이것들은 **PEP 492** 에 의해 도입되었습니다.

CPython
파이썬 프로그래밍 언어의 규범적인 구현인데, python.org에서 배포됩니다. 이 구현을 Jython 이나 IronPython 과 같은 다른 것들과 구별할 필요가 있을 때 용어 "CPython" 이 사용됩니다.

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 (데코레이터)
다른 함수를 돌려주는 함수인데, 보통 `@wrapper` 문법을 사용한 함수 변환으로 적용됩니다. 데코레이터의 흔한 예는 `classmethod()` 과 `staticmethod()` 입니다.

데코레이터 문법은 단지 편의 문법일 뿐입니다. 다음 두 함수 정의는 의미상으로 동등합니다:

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

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

같은 개념이 클래스에도 존재하지만, 덜 자주 쓰입니다. 데코레이터에 대한 더 자세한 내용은 함수 정의 와 클래스 정의 의 설명서를 보면 됩니다.

descriptor (디스크립터)
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.

디스크립터의 메서드들에 대한 자세한 내용은 descriptors나 디스크립터 사용법 안내서에 나옵니다.

dictionary (딕셔너리)
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 (딕셔너리 컴프리헨션)
이터러블에 있는 요소 전체나 일부를 처리하고 결과를 담은 딕셔너리를 반환하는 간결한 방법. `results = {n: n ** 2 for n in range(10)}`은 값 `n ** 2`에 매핑된 키 `n`을 포함하는 딕셔너리를 생성합니다. comprehensions을 참조하십시오.

dictionary view (딕셔너리 뷰)
`dict.keys()`, `dict.values()`, `dict.items()` 메서드가 돌려주는 객체들을 딕셔너리 뷰라고 부릅니다. 이것들은 딕셔너리 항목들에 대한 동적인 뷰를 제공하는데, 딕셔너리가 변경될 때, 뷰가 이 변화를 반영한다는 뜻입니다. 딕셔너리 뷰를 완전한 리스트로 바꾸려면 `list(dictview)`를 사용하면 됩니다. dict-views를 보세요.

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

duck-typing (덕 타이핑)
올바른 인터페이스를 가졌는지 판단하는데 객체의 형을 보지 않는 프로그래밍 스타일; 대신, 단순히 메서드나 어트리뷰트가 호출되거나 사용됩니다 ("오리처럼 보이고 오리처럼 꽥꽥댄다면, 그것은 오리다.") 특정한 형 대신에 인터페이스를 강조함으로써, 잘 설계된 코드는 다형적인 치환을 허락함으로써 유연성을 개선할 수 있습니다. 덕 타이핑은 `type()` 이나 `isinstance()` 을 사용한 검사를 피합니다. (하지만, 덕 타이핑이 추상 베이스 클래스 로 보완될 수 있음에 유의해야 합니다.) 대신에, `hasattr()` 검사나 *EAFP* 프로그래밍을 씁니다.

EAFP
허락보다는 용서를 구하기가 쉽다 (Easier to ask for forgiveness than permission). 이 흔히 볼 수 있는 파이썬 코딩 스타일은, 올바른 키나 어트리뷰트의 존재를 가정하고, 그 가정이 틀리면 예외를 잡습니다. 이 깔끔하고 빠른 스타일은 많은 `try`와 `except` 문의 존재로 특징지어집니다. 이 테크닉은 C와 같은 다른 많은 언어에서 자주 사용되는 *LBYL* 스타일과 대비됩니다.

expression (표현식)
어떤 값으로 구해질 수 있는 문법적인 조각. 다른 말로 표현하면, 표현식은 리터럴, 이름, 어트리뷰트 액세스, 연산자, 함수들과 같은 값을 돌려주는 표현 요소들을 쌓아 올린 것입니다. 다른 많은 언어와 대조적으로, 모든 언어 구성물들이 표현식인 것은 아닙니다. `while`처럼, 표현식으로 사용할 수 없는 문장 들이 있습니다. 대입 또한 문장이고, 표현식이 아닙니다.

extension module (확장 모듈)
C 나 C++로 작성된 모듈인데, 파이썬의 C API를 사용해서 핵심이나 사용자 코드와 상호 작용합니다.

f-string (f-문자열)
`'f'` 나 `'F'` 를 앞에 붙인 문자열 리터럴들을 흔히 "f-문자열" 이라고 부르는데, 포맷 문자열 리터럴의 줄임말입니다. **PEP 498** 을 보세요.

file object (파일 객체)
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*.

실제로는 세 부류의 파일 객체들이 있습니다. 날(raw) 바이너리 파일, 버퍼드(buffered) 바이너리 파일, 텍스트 파일. 이들의 인터페이스는 `io` 모듈에서 정의됩니다. 파일 객체를 만드는 규범적인 방법은 `open()` 함수를 쓰는 것입니다.

file-like object (파일류 객체)
파일 객체 의 비슷한 말.

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.

The filesystem encoding must guarantee to successfully decode all bytes below 128. If the file system encoding fails to provide this guarantee, API functions can raise `UnicodeError`.

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 (파인더)
임포트될 모듈을 위한 로더 를 찾으려고 시도하는 객체.

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 (정수 나눗셈)
가장 가까운 정수로 내림하는 수학적 나눗셈. 정수 나눗셈 연산자는 `//` 다. 예를 들어, 표현식 `11 // 4` 의 값은 `2`가 되지만, 실수 나눗셈은 `2.75`를 돌려줍니다. `(-11) // 4` 가 `-2.75`를 내림 한 `-3`이 됨에 유의해야 합니다. **PEP 238**을 보세요.

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 `codeobject.co_freevars` attribute, the term is also sometimes used as a synonym for *closure variable*.

function (함수)
호출자에게 어떤 값을 돌려주는 일련의 문장들. 없거나 그 이상의 인자 가 전달될 수 있는데, 바디의 실행에 사용될 수 있습니다. 매개변수 와 메서드 와 function 섹션도 보세요.

function annotation (함수 어노테이션)
함수 매개변수나 반환 값의 어노테이션.

함수 어노테이션은 일반적으로 형 힌트 로 사용됩니다: 예를 들어, 이 함수는 두 개의 `int` 인자를 받아들일 것으로 기대되고, 동시에 `int` 반환 값을 줄 것으로 기대됩니다:

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

함수 어노테이션 문법은 function 절에서 설명합니다.

See *variable annotation* and **PEP 484**, which describe this functionality. Also see annotations-howto for best practices on working with annotations.

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

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

garbage collection (가비지 수거)
더 사용되지 않는 메모리를 반납하는 절차. 파이썬은 참조 횟수 추적과 참조 순환을 감지하고 끊을 수 있는 순환 가비지 수거기를 통해 가비지 수거를 수행합니다. 가비지 수거기는 `gc` 모듈을 사용해서 제어할 수 있습니다.

generator (제너레이터)
제너레이터 이터레이터 를 돌려주는 함수. 일반 함수처럼 보이는데, 일련의 값들을 만드는 `yield` 표현식을 포함한다는 점이 다릅니다. 이 값들은 for-루프로 사용하거나 `next()` 함수로 한 번에 하나씩 꺼낼 수 있습니다.

보통 제너레이터 함수를 가리키지만, 어떤 문맥에서는 제너레이터 이터레이터 를 가리킵니다. 의도하는 의미가 명확하지 않은 경우는, 완전한 용어를 써서 모호함을 없앱니다.

generator iterator (제너레이터 이터레이터)
제너레이터 함수가 만드는 객체.

각 `yield`는 일시적으로 처리를 중단하고, 그 위치의 (지역 변수들과 대기 중인 try-문들을 포함하는) 실행 상태를 기억합니다. 제너레이터 이터레이터 가 재개되면, 떠난 곳으로 복귀합니다 (호출마다 새로 시작하는 함수와 대비됩니다).

generator expression (제너레이터 표현식)
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 (제네릭 함수)
같은 연산을 서로 다른 형들에 대해 구현한 여러 함수로 구성된 함수. 호출 때 어떤 구현이 사용될지는 디스패치 알고리즘에 의해 결정됩니다.

싱글 디스패치 용어집 항목과 `functools.singledispatch()` 데코레이터와 **PEP 443**도 보세요.

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

annotations.

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

GIL
전역 인터프리터 록 을 보세요.

global interpreter lock (전역 인터프리터 록)
한 번에 오직 하나의 스레드가 파이썬 바이트 코드 를 실행하도록 보장하기 위해 *CPython* 인터프리터가 사용하는 메커니즘. (`dict`와 같은 중요한 내장형들을 포함하는) 객체 모델이 묵시적으로 동시 액세스에 대해 안전하도록 만들어서 CPython 구현을 단순하게 만듭니다. 인터프리터 전체를 잠그는 것은 인터프리터를 다중스레드화하기 쉽게 만드는 대신, 다중 프로세서 기계가 제공하는 병렬성의 많은 부분을 희생합니다.

However, some extension modules, either standard or third-party, are designed so as to release the GIL when doing computationally intensive tasks such as compression or hashing. Also, the GIL is always released when doing I/O.

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 (해시 기반 pyc)
유효성을 판별하기 위해 해당 소스 파일의 최종 수정 시간이 아닌 해시를 사용하는 바이트 코드 캐시 파일. pyc-invalidation을 참조하세요.

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

해시 가능성은 객체를 딕셔너리의 키나 집합의 멤버로 사용할 수 있게 하는데, 이 자료 구조들이 내부적으로 해시값을 사용하기 때문입니다.

대부분 파이썬의 불변 내장 객체들은 해시 가능합니다; (리스트나 딕셔너리 같은) 가변 컨테이너들은 그렇지 않습니다; (튜플이나 frozenset 같은) 불변 컨테이너들은 그들의 요소들이 해시 가능할 때만 해시 가능합니다. 사용자 정의 클래스의 인스턴스 객체들은 기본적으로 해시 가능합니다. (자기 자신을 제외하고는) 모두 다르다고 비교되고, 해시값은 `id()`로 부터 만들어집니다.

IDLE
An Integrated Development and Learning Environment for Python. idle is a basic editor and interpreter environment which ships with the standard distribution of Python.

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 (불변)
고정된 값을 갖는 객체. 불변 객체는 숫자, 문자열, 튜플을 포함합니다. 이런 객체들은 변경될 수 없습니다. 새 값을 저장하려면 새 객체를 만들어야 합니다. 변하지 않는 해시값이 있어야 하는 곳에서 중요한 역할을 합니다, 예를 들어, 딕셔너리의 키.

import path (임포트 경로)
경로 기반 파인더 가 임포트 할 모듈을 찾기 위해 검색하는 장소들 (또는 경로 엔트리) 의 목록. 임포트 하는 동안, 이 장소들의 목록은 보통 `sys.path` 로부터 옵니다, 하지만 서브 패키지의 경우 부모 패키지의 `__path__` 어트리뷰트로부터 올 수도 있습니다.

importing (임포팅)
한 모듈의 파이썬 코드가 다른 모듈의 파이썬 코드에서 사용될 수 있도록 하는 절차.

importer (임포터)
모듈을 찾기도 하고 로드 하기도 하는 객체; 동시에 파인더 이자 로더 객체입니다.

interactive (대화형)

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 (인터프리티드)

바이트 코드 컴파일러의 존재 때문에 그 구분이 흐릿해지기는 하지만, 파이썬은 컴파일 언어가 아니라 인터프리터 언어입니다. 이것은 명시적으로 실행 파일을 만들지 않고도, 소스 파일을 직접 실행할 수 있다는 뜻입니다. 그 프로그램이 좀 더 천천히 실행되기는 하지만, 인터프리터 언어는 보통 컴파일 언어보다 짧은 개발/디버깅 주기를 갖습니다. 대화형 도 보세요.

interpreter shutdown (인터프리터 종료)

종료하라는 요청을 받을 때, 파이썬 인터프리터는 특별한 시기에 진입하는데, 모듈이나 여러 가지 중요한 내부 구조들과 같은 모든 할당된 자원들을 단계적으로 반납합니다. 또한, 가비지 수거기 를 여러 번 호출합니다. 사용자 정의 파괴자나 weakref 콜백에 있는 코드들의 실행을 시작시킬 수 있습니다. 종료 시기 동안 실행되는 코드는 다양한 예외들을 만날 수 있는데, 그것이 의존하는 자원들이 더 기능하지 않을 수 있기 때문입니다 (흔한 예는 라이브러리 모듈이나 경고 장치들입니다).

인터프리터 종료의 주된 원인은 실행되는 `__main__` 모듈이나 스크립트가 실행을 끝내는 것입니다.

iterable (이터러블)

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 (이터레이터)

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

typeiter 에 더 자세한 내용이 있습니다.

CPython 구현 상세: 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 (키 함수)

키 함수 또는 콜레이션(collation) 함수는 정렬(sorting)이나 배열(ordering)에 사용되는 값을 돌려주는 콜러블입니다. 예를 들어, `locale.strxfrm()` 은 로케일 특정 방식을 따르는 정렬 키를 만드는 데 사용됩니다.

파이썬의 많은 도구가 요소들이 어떻게 순서 지어지고 묶이는지를 제어하기 위해 키 함수를 받아들입니다. 이런 것들에는 `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()`, `itertools.groupby()` 이 있습니다.

There are several ways to create a key function. For example. the `str.lower()` method can serve as a key function for case insensitive sorts. Alternatively, a key function can be built from a `lambda` expression such as `lambda r: (r[0], r[2])`. Also, `operator.attrgetter()`, `operator.itemgetter()`, and

`operator.methodcaller()` are three key function constructors. See the Sorting HOW TO for examples of how to create and use key functions.

keyword argument (키워드 인자)
인자 를 보세요.

lambda (람다)
호출될 때 값이 구해지는 하나의 표현식 으로 구성된 이름 없는 인라인 함수. 람다 함수를 만드는 문법은 `lambda [parameters]: expression` 입니다.

LBYL
뛰기 전에 보라 (Look before you leap). 이 코딩 스타일은 호출이나 조회를 하기 전에 명시적으로 사전 조건들을 검사합니다. 이 스타일은 *EAFP* 접근법과 대비되고, 많은 `if` 문의 존재로 특징지어집니다.

다중 스레드 환경에서, LBYL 접근법은 "보기" 와 "뛰기" 간에 경쟁 조건을 만들게 될 위험이 있습니다. 예를 들어, 코드 `if key in mapping: return mapping[key]` 는 검사 후에, 하지만 조회 전에, 다른 스레드가 *key* 를 *mapping* 에서 제거하면 실패할 수 있습니다. 이런 이슈는 록이나 EAFP 접근법을 사용함으로써 해결될 수 있습니다.

list (리스트)
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 (리스트 컴프리헨션)
시퀀스의 요소들 전부 또는 일부를 처리하고 그 결과를 리스트로 돌려주는 간결한 방법. `result = ['{:#04x}'.format(x) for x in range(256) if x % 2 == 0]` 는 0에서 255 사이에 있는 짝수들의 16진수 (0x..) 들을 포함하는 문자열의 리스트를 만듭니다. `if` 절은 생략할 수 있습니다. 생략하면, `range(256)` 에 있는 모든 요소가 처리됩니다.

loader (로더)
An object that loads a module. It must define a method named `load_module()`. A loader is typically returned by a *finder*. See also:

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

locale encoding
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 (매직 메서드)
특수 메서드 의 비공식적인 비슷한 말.

mapping (매핑)
A container object that supports arbitrary key lookups and implements the methods specified in the `collections.abc.Mapping` or `collections.abc.MutableMapping` abstract base classes. Examples include `dict`, `collections.defaultdict`, `collections.OrderedDict` and `collections.Counter`.

meta path finder (메타 경로 파인더)
`sys.meta_path` 의 검색이 돌려주는 파인더. 메타 경로 파인더는 경로 엔트리 파인더 와 관련되어 있기는 하지만 다릅니다.

메타 경로 파인더가 구현하는 메서드들에 대해서는 `importlib.abc.MetaPathFinder` 를 보면 됩니다.

metaclass (메타 클래스)
클래스의 클래스. 클래스 정의는 클래스 이름, 클래스 딕셔너리, 베이스 클래스들의 목록을 만듭니다. 메타 클래스는 이 세 인자를 받아서 클래스를 만드는 책임을 집니다. 대부분의 객체 지향형 프로그래밍 언어들은 기본 구현을 제공합니다. 파이썬을 특별하게 만드는 것은 커스텀 메타 클래스를 만들 수 있다는 것입니다. 대부분 사용자에게는 이 도구가 전혀 필요 없지만, 필요가 생길 때, 메타 클래스는 강력하고 우아한 해법을 제공합니다. 어트리뷰트 액세스의 로깅(logging), 스레드 안전성의 추가, 객체 생성 추적, 싱글톤 구현과 많은 다른 작업에 사용됐습니다.

metaclasses 에서 더 자세한 내용을 찾을 수 있습니다.

method (메서드)
클래스 바디 안에서 정의되는 함수. 그 클래스의 인스턴스의 어트리뷰트로서 호출되면, 그 메서드는 첫 번째 인자 (보통 `self` 라고 불린다) 로 인스턴스 객체를 받습니다. 함수 와 중첩된 스코프 를 보세요.

method resolution order (메서드 결정 순서)
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 (모듈)
파이썬 코드의 조직화 단위를 담당하는 객체. 모듈은 임의의 파이썬 객체들을 담는 이름 공간을 갖습니다. 모듈은 임포팅 절차에 의해 파이썬으로 로드됩니다.

패키지 도 보세요.

module spec (모듈 스펙)
모듈을 로드하는데 사용되는 임포트 관련 정보들을 담고 있는 이름 공간. `importlib.machinery.ModuleSpec` 의 인스턴스.

See also module-specs.

MRO
메서드 결정 순서 를 보세요.

mutable (가변)
가변 객체는 값이 변할 수 있지만 `id()` 는 일정하게 유지합니다. 불변 도 보세요.

named tuple (네임드 튜플)
"named tuple(네임드 튜플)" 이라는 용어는 튜플에서 상속하고 이름 붙은 어트리뷰트를 사용하여 인덱스 할 수 있는 요소에 액세스 할 수 있는 모든 형이나 클래스에 적용됩니다. 형이나 클래스에는 다른 기능도 있을 수 있습니다.

`time.localtime()` 과 `os.stat()` 가 반환한 값을 포함하여, 여러 내장형이 네임드 튜플입니다. 또 다른 예는 `sys.float_info` 입니다:

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

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

namespace (이름 공간)
변수가 저장되는 장소. 이름 공간은 딕셔너리로 구현됩니다. 객체에 중첩된 이름 공간 (메서드 에서) 뿐만 아니라 지역, 전역, 내장 이름 공간이 있습니다. 이름 공간은 이름 충돌을 방지해서 모듈성을 지원합니다. 예를 들어, 함수 `builtins.open` 과 `os.open()` 은 그들의 이름 공간에 의해 구별됩니다. 또한, 이름 공간은 어떤 모듈이 함수를 구현하는지를 분명하게 만들어서 가독성과 유지 보수성에

도움을 줍니다. 예를 들어, `random.seed()` 또는 `itertools.islice()` 라고 쓰면 그 함수들이 각각 `random` 과 `itertools` 모듈에 의해 구현되었음이 명확해집니다.

namespace package (이름 공간 패키지)
오직 서브 패키지들의 컨테이너로만 기능하는 **PEP 420** 패키지. 이름 공간 패키지는 물리적인 실체가 없을 수도 있고, 특히 `__init__.py` 파일이 없으므로 정규 패키지 와는 다릅니다.

모듈 도 보세요.

nested scope (중첩된 스코프)
둘러싼 정의에서 변수를 참조하는 능력. 예를 들어, 다른 함수 내부에서 정의된 함수는 바깥 함수에 있는 변수들을 참조할 수 있습니다. 중첩된 스코프는 기본적으로는 참조만 가능할 뿐, 대입은 되지 않는다는 것에 주의해야 합니다. 지역 변수들은 가장 내부의 스코프에서 읽고 씁니다. 마찬가지로, 전역 변수들은 전역 이름 공간에서 읽고 씁니다. `nonlocal` 은 바깥 스코프에 쓰는 것을 허락합니다.

new-style class (뉴스타일 클래스)
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 (객체)
상태 (어트리뷰트나 값) 를 갖고 동작 (메서드) 이 정의된 모든 데이터. 또한, 모든 뉴스타일 클래스 의 최종적인 베이스 클래스입니다.

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 (패키지)
A Python *module* which can contain submodules or recursively, subpackages. Technically, a package is a Python module with a `__path__` attribute.

정규 패키지 와 이름 공간 패키지 도 보세요.

parameter (매개변수)
함수 (또는 메서드) 정의에서 함수가 받을 수 있는 인자 (또는 어떤 경우 인자들) 를 지정하는 이름 붙은 엔티티. 다섯 종류의 매개변수가 있습니다:

- 위치-키워드 (*positional-or-keyword*): 위치 인자 나 키워드 인자 로 전달될 수 있는 인자를 지정합니다. 이것이 기본 형태의 매개변수입니다, 예를 들어 다음에서 *foo* 와 *bar*:

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

- 위치-전용 (*positional-only*): 위치로만 제공될 수 있는 인자를 지정합니다. 위치 전용 매개변수는 함수 정의의 매개변수 목록에 `/` 문자를 포함하고 그 뒤에 정의할 수 있습니다, 예를 들어 다음에서 *posonly1* 과 *posonly2*:

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

- 키워드-전용 (*keyword-only*): 키워드로만 제공될 수 있는 인자를 지정합니다. 키워드-전용 매개변수는 함수 정의의 매개변수 목록에서 앞에 하나의 가변-위치 매개변수나 `*`를 그대로 포함해서 정의할 수 있습니다. 예를 들어, 다음에서 *kw_only1* 와 *kw_only2*:

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

- 가변-위치 (*var-positional*): (다른 매개변수들에 의해서 이미 받아들여진 위치 인자들에 더해) 제공될 수 있는 위치 인자들의 임의의 시퀀스를 지정합니다. 이런 매개변수는 매개변수 이름에 `*` 를 앞에 붙여서 정의될 수 있습니다, 예를 들어 다음에서 *args*:

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

- 가변-키워드 (*var-keyword*): (다른 매개변수들에 의해서 이미 받아들여진 키워드 인자들에 더해) 제공될 수 있는 임의의 개수 키워드 인자들을 지정합니다. 이런 매개변수는 매개변수 이름에 `**`를 앞에 붙여서 정의될 수 있습니다, 예를 들어 위의 예에서 *kwargs*.

매개변수는 선택적 인자들을 위한 기본값뿐만 아니라 선택적이거나 필수 인자들을 지정할 수 있습니다.

인자 용어집 항목, 인자와 매개변수의 차이에 나오는 FAQ 질문, `inspect.Parameter` 클래스, function 절, **PEP 362**도 보세요.

path entry (경로 엔트리)
경로 기반 파인더 가 임포트 할 모듈들을 찾기 위해 참고하는 임포트 경로 상의 하나의 장소.

path entry finder (경로 엔트리 파인더)
`sys.path_hooks` 에 있는 콜러블 (즉, 경로 엔트리 훅) 이 돌려주는 파인더 인데, 주어진 경로 엔트리 로 모듈을 찾는 방법을 알고 있습니다.

경로 엔트리 파인더들이 구현하는 메서드들은 `importlib.abc.PathEntryFinder` 에 나옵니다.

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

path based finder (경로 기반 파인더)
기본 메타 경로 파인더들 중 하나인데, 임포트 경로 에서 모듈을 찾습니다.

path-like object (경로류 객체)
파일 시스템 경로를 나타내는 객체. 경로류 객체는 경로를 나타내는 `str` 나 `bytes` 객체이거나 `os.PathLike` 프로토콜을 구현하는 객체입니다. `os.PathLike` 프로토콜을 지원하는 객체는 `os.fspath()` 함수를 호출해서 `str` 나 `bytes` 파일 시스템 경로로 변환될 수 있습니다; 대신 `os.fsdecode()` 와 `os.fsencode()` 는 각각 `str` 나 `bytes` 결과를 보장하는데 사용될 수 있습니다. **PEP 519**로 도입되었습니다.

PEP
파이썬 개선 제안. PEP는 파이썬 커뮤니티에 정보를 제공하거나 파이썬 또는 그 프로세스 또는 환경에 대한 새로운 기능을 설명하는 설계 문서입니다. PEP는 제안된 기능에 대한 간결한 기술 사양 및 근거를 제공해야 합니다.

PEP는 주요 새로운 기능을 제안하고 문제에 대한 커뮤니티 입력을 수집하며 파이썬에 들어간 설계 결정을 문서로 만들기 위한 기본 메커니즘입니다. PEP 작성자는 커뮤니티 내에서 합의를 구축하고 반대 의견을 문서화 할 책임이 있습니다.

PEP 1 참조하세요.

portion (포션)
PEP 420 에서 정의한 것처럼, 이름 공간 패키지에 이바지하는 하나의 디렉터리에 들어있는 파일들의 집합 (zip 파일에 저장되는 것도 가능합니다).

positional argument (위치 인자)
인자 를 보세요.

provisional API (잠정 API)
잠정 API는 표준 라이브러리의 과거 호환성 보장으로부터 신중히 제외된 것입니다. 인터페이스의 큰 변화가 예상되지는 않지만, 잠정적이라고 표시되는 한, 코어 개발자들이 필요하다고 생각한다면 과거 호환성이 유지되지 않는 변경이 일어날 수 있습니다. 그런 변경은 불필요한 방식으로 일어나지는 않을 것입니다 — API를 포함하기 전에 놓친 중대하고 근본적인 결함이 발견된 경우에만 일어날 것입니다.

잠정 API에서조차도, 과거 호환성이 유지되지 않는 변경은 "최후의 수단" 으로 여겨집니다 - 모든 식별된 문제들에 대해 과거 호환성을 유지하는 해법을 찾으려는 모든 시도가 선행됩니다.

이 절차는 표준 라이브러리가 오랜 시간 동안 잘못된 설계 오류에 발목 잡히지 않고 발전할 수 있도록 만듭니다. 더 자세한 내용은 **PEP 411**을 보면 됩니다.

provisional package (잠정 패키지)
잠정 *API* 를 보세요.

Python 3000 (파이썬 3000)
파이썬 3.x 배포 라인의 별명 (버전 3의 배포가 먼 미래의 이야기던 시절에 만들어진 이름이다.) 이것을 "Py3k" 로 줄여 쓰기도 합니다.

Pythonic (파이썬다운)
다른 언어들에서 일반적인 개념들을 사용해서 코드를 구현하는 대신, 파이썬 언어에서 가장 자주 사용되는 이디엄들을 가까이 따르는 아이디어나 코드 조각. 예를 들어, 파이썬에서 자주 쓰는 이디엄은 `for` 문을 사용해서 이터러블의 모든 요소로 루핑하는 것입니다. 다른 많은 언어에는 이런 종류의 구성물이 없으므로, 파이썬에 익숙하지 않은 사람들은 대신에 숫자 카운터를 사용하기도 합니다:

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

더 깔끔한, 파이썬다운 방법은 이렇습니다:

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

qualified name (정규화된 이름)
모듈의 전역 스코프에서 모듈에 정의된 클래스, 함수, 메서드에 이르는 "경로"를 보여주는 점으로 구분된 이름. **PEP 3155** 에서 정의됩니다. 최상위 함수와 클래스의 경우에, 정규화된 이름은 객체의 이름과 같습니다:

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

모듈을 가리키는데 사용될 때, 완전히 정규화된 이름(*fully qualified name*)은 모든 부모 패키지들을 포함해서 모듈로 가는 점으로 분리된 이름을 의미합니다, 예를 들어, `email.mime.text`:

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

reference count (참조 횟수)
The number of references to an object. When the reference count of an object drops to zero, it is deallocated. Some objects are *immortal* and have reference counts that are never modified, and therefore the objects are never deallocated. Reference counting is generally not visible to Python code, but it is a key element of the *CPython* implementation. Programmers can call the `sys.getrefcount()` function to return the reference count for a particular object.

regular package (정규 패키지)
`__init__.py` 파일을 포함하는 디렉터리와 같은 전통적인 패키지.

이름 공간 패키지 도 보세요.

REPL
An acronym for the "read–eval–print loop", another name for the *interactive* interpreter shell.

__slots__
클래스 내부의 선언인데, 인스턴스 어트리뷰트들을 위한 공간을 미리 선언하고 인스턴스 딕셔너리를 제거함으로써 메모리를 절감하는 효과를 줍니다. 인기 있기는 하지만, 이 테크닉은 올바르게 사용하기가 좀 까다로운 편이라서, 메모리에 민감한 응용 프로그램에서 많은 수의 인스턴스가 있는 특별한 경우로 한정하는 것이 좋습니다.

sequence (시퀀스)
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 (집합 컴프리헨션)
이터러블에 있는 요소 전체나 일부를 처리하고 결과를 담은 집합을 반환하는 간결한 방법. `results = {c for c in 'abracadabra' if c not in 'abc'}`는 문자열의 집합 `{'r', 'd'}`를 생성합니다. comprehensions을 참조하십시오.

single dispatch (싱글 디스패치)
구현이 하나의 인자의 형에 기초해서 결정되는 제네릭 함수 디스패치의 한 형태.

slice (슬라이스)
보통 시퀀스 의 일부를 포함하는 객체. 슬라이스는 서브 스크립트 표기법을 사용해서 만듭니다. `variable_name[1:3:5]` 처럼, `[]` 안에서 여러 개의 숫자를 콜론으로 분리합니다. 대괄호 (서브 스크립트) 표기법은 내부적으로 `slice` 객체를 사용합니다.

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 (특수 메서드)
파이썬이 형에 어떤 연산을, 덧셈 같은, 실행할 때 묵시적으로 호출되는 메서드. 이런 메서드는 두 개의 밑줄로 시작하고 끝나는 이름을 갖고 있습니다. 특수 메서드는 specialnames 에 문서로 만들어져 있습니다.

statement (문장)
문장은 스위트 (코드의 "블록(block)") 를 구성하는 부분입니다. 문장은 표현식 이거나 키워드를 사용하는 여러 가지 구조물 중의 하나입니다. 가령 `if`, `while`, `for`.

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

strong reference
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 (텍스트 인코딩)
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 (텍스트 파일)
str 객체를 읽고 쓸 수 있는 파일 객체. 종종, 텍스트 파일은 실제로는 바이트 지향 데이터스트림을 액세스하고 텍스트 인코딩 을 자동 처리합니다. 텍스트 파일의 예로는 텍스트 모드 ('r' 또는 'w') 로 열린 파일, sys.stdin, sys.stdout, io.StringIO 의 인스턴스를 들 수 있습니다.

바이트열류 객체 를 읽고 쓸 수 있는 파일 객체에 대해서는 바이너리 파일 도 참조하세요.

triple-quoted string (삼중 따옴표 된 문자열)
따옴표 (") 나 작은따옴표 (') 세 개로 둘러싸인 문자열. 그냥 따옴표 하나로 둘러싸인 문자열에 없는 기능을 제공하지는 않지만, 여러 가지 이유에서 쓸모가 있습니다. 이스케이프 되지 않은 작은따옴표나 큰따옴표를 문자열 안에 포함할 수 있도록 하고, 연결 문자를 쓰지 않고도 여러 줄에 걸칠 수 있는데, 독스트링을 쓸 때 특히 쓸모 있습니다.

type (형)
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 (형 에일리어스)
형을 식별자에 대입하여 만들어지는 형의 동의어.

형 에일리어스는 형 힌트를 단순화하는 데 유용합니다. 예를 들면:

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

는 다음과 같이 더 읽기 쉽게 만들 수 있습니다:

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

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

이 기능을 설명하는 typing과 **PEP 484**를 참조하세요.

type hint (형 힌트)
변수, 클래스 어트리뷰트 및 함수 매개변수 나 반환 값의 기대되는 형을 지정하는 어노테이션.

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.

지역 변수를 제외하고, 전역 변수, 클래스 어트리뷰트 및 함수의 형 힌트는 typing.get_type_hints() 를 사용하여 액세스할 수 있습니다.

이 기능을 설명하는 typing과 **PEP 484**를 참조하세요.

universal newlines (유니버설 줄 넘김)
다음과 같은 것들을 모두 줄의 끝으로 인식하는, 텍스트 스트림을 해석하는 태도: 유닉스 개행 문자 관례 '\n', 윈도우즈 관례 '\r\n', 예전의 매킨토시 관례 '\r'. 추가적인 사용에 관해서는 bytes.splitlines() 뿐만 아니라 **PEP 278** 와 **PEP 3116** 도 보세요.

variable annotation (변수 어노테이션)
변수 또는 클래스 어트리뷰트의 어노테이션.

변수 또는 클래스 어트리뷰트에 어노테이션을 달 때 대입은 선택 사항입니다:

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

변수 어노테이션은 일반적으로 형 힌트로 사용됩니다: 예를 들어, 이 변수는 int 값을 가질 것으로 기대됩니다:

```
count: int = 0
```

변수 어노테이션 문법은 섹션 annassign 에서 설명합니다.

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

virtual environment (가상 환경)
파이썬 사용자와 응용 프로그램이, 같은 시스템에서 실행되는 다른 파이썬 응용 프로그램들의 동작에 영향을 주지 않으면서, 파이썬 배포 패키지들을 설치하거나 업그레이드하는 것을 가능하게 하는, 협력적으로 격리된 실행 환경.

`venv` 도 보세요.

virtual machine (가상 기계)
소프트웨어만으로 정의된 컴퓨터. 파이썬의 가상 기계는 바이트 코드 컴파일러가 출력하는 바이트 코드를 실행합니다.

Zen of Python (파이썬 젠)
파이썬 디자인 원리와 철학들의 목록인데, 언어를 이해하고 사용하는 데 도움이 됩니다. 이 목록은 대화형 프롬프트에서 "`import this`" 를 입력하면 보입니다.

APPENDIX B

About this documentation

Python' s documentation is generated from reStructuredText sources using Sphinx, a documentation generator originally created for Python and now maintained as an independent project.

설명서와 이를 위한 툴체인 개발은 파이썬 자체와 마찬가지로 전적으로 자원봉사자의 노력입니다. 기여하고 싶다면, 참여 방법에 대한 정보는 reporting-bugs 페이지를 참고하십시오. 새로운 자원봉사자는 언제나 환영합니다!

다음 분들에게 많은 감사를 드립니다:

- Fred L. Drake, Jr., the creator of the original Python documentation toolset and author of much of the content;
- the Docutils project for creating reStructuredText and the Docutils suite;
- Fredrik Lundh for his Alternative Python Reference project from which Sphinx got many good ideas.

B.1 Contributors to the Python documentation

많은 사람이 파이썬 언어, 파이썬 표준 라이브러리 및 파이썬 설명서에 기여했습니다. 기여자의 부분적인 목록은 파이썬 소스 배포판의 Misc/ACKS 를 참조하십시오.

파이썬이 이런 멋진 설명서를 갖게 된 것은 파이썬 커뮤니티의 입력과 기여 때문입니다 – 감사합니다!

APPENDIX C

역사와 라이센스

C.1 소프트웨어의 역사

파이썬은 ABC라는 언어의 후계자로서 네덜란드의 Stichting Mathematisch Centrum (CWI, https://www.cwi.nl/ 참조)의 Guido van Rossum에 의해 1990년대 초반에 만들어졌습니다. 파이썬에는 다른 사람들의 많은 공헌이 포함되었지만, Guido는 파이썬의 주요 저자로 남아 있습니다.

1995년, Guido는 Virginia의 Reston에 있는 Corporation for National Research Initiatives(CNRI, https://www.cnri.reston.va.us/ 참조)에서 파이썬 작업을 계속했고, 이곳에서 여러 버전의 소프트웨어를 출시했습니다.

2000년 5월, Guido와 파이썬 핵심 개발팀은 BeOpen.com으로 옮겨서 BeOpen PythonLabs 팀을 구성했습니다. 같은 해 10월, PythonLabs 팀은 Digital Creations(현재 Zope Corporation; https://www.zope.org/ 참조)로 옮겼습니다. 2001 년, 파이썬 소프트웨어 재단(PSF, https://www.python.org/psf/ 참조)이 설립되었습니다. 이 단체는 파이썬 관련 지적 재산권을 소유하도록 특별히 설립된 비영리 조직입니다. Zope Corporation은 PSF의 후원 회원입니다.

모든 파이썬 배포판은 공개 소스입니다 (공개 소스 정의에 대해서는 https://opensource.org/를 참조하십시오). 역사적으로, 대부분 (하지만 전부는 아닙니다) 파이썬 배포판은 GPL과 호환됩니다; 아래의 표는 다양한 배포판을 요약한 것입니다.

배포판	파생된 곳	해	소유자	GPL 호환?
0.9.0 ~ 1.2	n/a	1991-1995	CWI	yes
1.3 ~ 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	no
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2 이상	2.1.1	2001-현재	PSF	yes

참고

GPL과 호환된다는 것은 우리가 GPL로 파이썬을 배포한다는 것을 의미하지는 않습니다. 모든 파이썬 라이센스는 GPL과 달리 여러분의 변경을 공개 소스로 만들지 않고 수정된 버전을 배포할 수 있게 합니다. GPL 호환 라이센스는 파이썬과 GPL 하에 발표된 다른 소프트웨어를 결합할 수 있게 합니다; 다른 것들은 그렇지 않습니다.

Guido의 지도하에 이 배포를 가능하게 만든 많은 외부 자원봉사자들에게 감사드립니다.

C.2 파이썬에 액세스하거나 사용하기 위한 이용 약관

파이썬 소프트웨어와 설명서는 *PSF License Agreement*에 따라 라이센스가 부여됩니다.

파이썬 3.8.6부터, 설명서의 예제, 조리법 및 기타 코드는 PSF License Agreement와 *Zero-Clause BSD license*에 따라 이중 라이센스가 부여됩니다.

파이썬에 통합된 일부 소프트웨어에는 다른 라이센스가 적용됩니다. 라이센스는 해당 라이센스에 해당하는 코드와 함께 나열됩니다. 이러한 라이센스의 불완전한 목록은 포함된 소프트웨어에 대한 라이센스 및 승인를 참조하십시오.

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 3.13.1

```
1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and
   the Individual or Organization ("Licensee") accessing and otherwise using Python
   3.13.1 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
   grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,
   analyze, test, perform and/or display publicly, prepare derivative works,
   distribute, and otherwise use Python 3.13.1 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 3.13.1 alone or in any derivative version
   prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or
   incorporates Python 3.13.1 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
   3.13.1.

4. PSF is making Python 3.13.1 available to Licensee on an "AS IS" basis.
   PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED.  BY WAY OF
   EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR
   WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE
   USE OF PYTHON 3.13.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.13.1
   FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF
   MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.13.1, 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. Nothing in this License Agreement shall be deemed to create any relationship
   of agency, partnership, or joint venture between PSF and Licensee.  This License
   Agreement does not grant permission to use PSF trademarks or trade name in a
   trademark sense to endorse or promote products or services of Licensee, or any
```

```
   third party.

8. By copying, installing or otherwise using Python 3.13.1, 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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   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 3.13.1 DOCUMENTATION

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

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

C.3 포함된 소프트웨어에 대한 라이센스 및 승인

이 섹션은 파이썬 배포판에 포함된 제삼자 소프트웨어에 대한 불완전하지만 늘어나고 있는 라이센스와 승인의 목록입니다.

C.3.1 메르센 트위스터

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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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 소켓

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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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 비동기 소켓 서비스

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 쿠키 관리

`http.cookies` 모듈은 다음과 같은 주의 사항을 포함합니다:

```
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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

C.3.5 실행 추적

`trace` 모듈은 다음과 같은 주의 사항을 포함합니다:

```
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 및 UUdecode 함수

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.
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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 원격 프로시저 호출

`xmlrpc.client` 모듈은 다음과 같은 주의 사항을 포함합니다:

```
    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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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` 모듈은 kqueue 인터페이스에 대해 다음과 같은 주의 사항을 포함합니다:

```
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

파일 `Python/pyhash.c` 에는 Dan Bernstein의 SipHash24 알고리즘의 Marek Majkowski의 구현이 포함되어 있습니다. 여기에는 다음과 같은 내용이 포함되어 있습니다:

```
<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/
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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 와 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:

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

C.3.12 OpenSSL

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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   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,
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   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,
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
   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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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

`zlib` 확장은 시스템에서 발견된 zlib 버전이 너무 오래되어서 빌드에 사용될 수 없으면, 포함된 zlib 소스 사본을 사용하여 빌드됩니다:

```
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.
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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

`tracemalloc` 에 의해 사용되는 해시 테이블의 구현은 cfuhash 프로젝트를 기반으로 합니다:

```
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 테스트 스위트

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,
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
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 subr_smr.c. The file is distributed under the 2-Clause BSD License:

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

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice unmodified, this list of conditions, and the following
   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.
```

APPENDIX D

저작권

파이썬과 이 설명서는:

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.

전체 라이센스 및 사용 권한 정보는 역사와 라이센스 에서 제공합니다.

색인

알파벳 이외

...., 11
>>>, 11
__future__, 17
__slots__, 24

A

abstract base class (추상 베이스 클래스), 11
annotation (어노테이션), 11
argument (인자), 11
asynchronous context manager (비동기 컨텍스트 관리자), 12
asynchronous generator (비동기 제너레이터), 12
asynchronous generator iterator (비동기 제너레이터 이터레이터), 12
asynchronous iterable (비동기 이터러블), 12
asynchronous iterator (비동기 이터레이터), 12
attribute (어트리뷰트), 12
awaitable (어웨이터블), 12

B

BDFL, 12
binary file (바이너리 파일), 12
borrowed reference, 13
bytecode (바이트 코드), 13
bytes-like object (바이트열류 객체), 13

C

callable, 13
callback (콜백), 13
C-contiguous, 14
class (클래스), 13
class variable (클래스 변수), 13
closure variable, 13
complex number (복소수), 14
context, 14
context management protocol, 14
context manager (컨텍스트 관리자), 14
context variable (컨텍스트 변수), 14
contiguous (연속), 14
coroutine (코루틴), 14
coroutine function (코루틴 함수), 14
CPython, 14
current context, 14

D

decorator (데코레이터), 15
descriptor (디스크립터), 15
dictionary (딕셔너리), 15
dictionary comprehension (딕셔너리 컴프리헨션), 15
dictionary view (딕셔너리 뷰), 15
docstring (독스트링), 15
duck-typing (덕 타이핑), 15

E

EAFP, 15
expression (표현식), 16
extension module (확장 모듈), 16

F

f-string (*f*-문자열), 16
file object (파일 객체), 16
file-like object (파일류 객체), 16
filesystem encoding and error handler, 16
finder (파인더), 16
floor division (정수 나눗셈), 16
Fortran contiguous, 14
free threading, 16
free variable, 16
function (함수), 17
function annotation (함수 어노테이션), 17

G

garbage collection (가비지 수거), 17
generator (제너레이터), 17
generator expression (제너레이터 표현식), 17
generator iterator (제너레이터 이터레이터), 17
generic function (제네릭 함수), 17
generic type (제네릭 형), 17
GIL, 18
global interpreter lock (전역 인터프리터 록), 18

H

hash-based pyc (해시 기반 *pyc*), 18

hashable (해시 가능), 18

I

IDLE, 18
immortal, 18
immutable (불변), 18
import path (임포트 경로), 18
importer (임포터), 18
importing (임포팅), 18
interactive (대화형), 19
interpreted (인터프리티드), 19
interpreter shutdown (인터프리터 종료), 19
iterable (이터러블), 19
iterator (이터레이터), 19

K

key function (키 함수), 19
keyword argument (키워드 인자), 20

L

lambda (람다), 20
LBYL, 20
list (리스트), 20
list comprehension (리스트 컴프리헨션), 20
loader (로더), 20
locale encoding, 20

M

magic
    method (메서드), 20
magic method (매직 메서드), 20
mapping (매핑), 20
meta path finder (메타 경로 파인더), 20
metaclass (메타 클래스), 21
method (메서드), 21
    magic, 20
    special, 25
method resolution order (메서드 결정 순서), 21
module (모듈), 21
module spec (모듈 스펙), 21
MRO, 21
mutable (가변), 21

N

named tuple (네임드 튜플), 21
namespace (이름 공간), 21
namespace package (이름 공간 패키지), 22
nested scope (중첩된 스코프), 22
new-style class (뉴스타일 클래스), 22

O

object (객체), 22
optimized scope, 22

P

package (패키지), 22
parameter (매개변수), 22
path based finder (경로 기반 파인더), 23
path entry (경로 엔트리), 23
path entry finder (경로 엔트리 파인더), 23
path entry hook (경로 엔트리 훅), 23
path-like object (경로류 객체), 23
PEP, 23
portion (포션), 23
positional argument (위치 인자), 23
provisional API (잠정 *API*), 23
provisional package (잠정 패키지), 23
Python 3000 (파이썬 *3000*), 24
Python 향상 제안
    PEP 1, 23
    PEP 238, 16
    PEP 278, 26
    PEP 302, 20
    PEP 343, 14
    PEP 362, 12, 23
    PEP 411, 23
    PEP 420, 22, 23
    PEP 443, 17
    PEP 483, 18
    PEP 484, 11, 17, 18, 26, 27
    PEP 492, 12, 14
    PEP 498, 16
    PEP 519, 23
    PEP 525, 12
    PEP 526, 11, 27
    PEP 585, 18
    PEP 683, 18
    PEP 703, 16, 18
    PEP 3116, 26
    PEP 3155, 24
PYTHON_GIL, 18
Pythonic (파이썬다운), 24

Q

qualified name (정규화된 이름), 24

R

reference count (참조 횟수), 24
regular package (정규 패키지), 24
REPL, 24

S

sequence (시퀀스), 25
set comprehension (집합 컴프리헨션), 25
single dispatch (싱글 디스패치), 25
slice (슬라이스), 25
soft deprecated, 25
special
    method (메서드), 25
special method (특수 메서드), 25
statement (문장), 25
static type checker, 25
strong reference, 25

T

text encoding (텍스트 인코딩), 25
text file (텍스트 파일), 26
triple-quoted string (삼중 따옴표 된 문자열), 26
type (형), 26
type alias (형 에일리어스), 26
type hint (형 힌트), 26

U

universal newlines (유니버설 줄 넘김), 26

V

variable annotation (변수 어노테이션), 26
virtual environment (가상 환경), 27
virtual machine (가상 기계), 27

Y

환경 변수
    PYTHON_GIL, 18

Z

Zen of Python (파이썬 젠), 27