
Installing Python Modules

출시 버전 **3.7.16**

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

2월 08, 2023

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?	8
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	이 설명서에 관하여	25
B.1	파이썬 설명서의 공헌자들	25
C	역사와 라이선스	27
C.1	소프트웨어의 역사	27
C.2	파이썬에 액세스하거나 사용하기 위한 이용 약관	28
C.3	포함된 소프트웨어에 대한 라이선스 및 승인	31
D	저작권	45
	색인	47

이메일 distutils-sig@python.org

널리 사용되는 공개 소스 개발 프로젝트로서, 파이썬에는 적극적으로 지원하는 기여자와 사용자의 커뮤니티가 있어, 자신들의 소프트웨어를 공개 소스 라이선스 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.

이 지침서는 이 절차의 설치 부분을 다룹니다. 자신의 파이썬 프로젝트를 만들고 공유하는 지침은 배포 지침서를 참조하십시오.

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.
- **파이썬 패키지 색인** 은 다른 파이썬 사용자가 사용할 수 있도록 만들어진 공개 소스 라이선스 패키지의 공용 저장소입니다.
- 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 both [GitHub](#) and [Bitbucket](#).
- `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 3.6부터 폐지: `pyvenv` 는 파이썬 3.3과 3.4에서 가상 환경을 만드는 데 추천되는 도구였으며 파이썬 3.6에서 폐지되었습니다.

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.

다음 명령은 파이썬 패키지 색인에서 최신 버전의 모듈과 그 의존성을 설치합니다:

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

참고: POSIX 사용자(맥 OS X 와 리눅스 사용자를 포함합니다)의 경우, 이 지침서의 예제는 **가상 환경** 을 사용한다고 가정합니다.

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 셀에서 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](#)

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 계산을 위한 파이썬 패키지를 설치합니까?

많은 과학 계산을 위한 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 방법으로 설치하는 것이 더 쉬운 경우가 종종 있습니다.

더 보기:

파이썬 패키징 사용자 지침서: 과학 계산을 위한 패키지 설치하기

3.4 … 병렬로 설치된 여러 버전의 파이썬으로 작업합니까?

리눅스, 맥 OS X 및 다른 POSIX 시스템에서는, 버전이 지정된 파이썬 명령을 `-m` 스위치와 조합하여 적절한 `pip` 사본을 실행하십시오:

```
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
```

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
```

pip 설치 에 대한 추가 자원도 있습니다.

4.3 바이너리 확장 설치하기

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

바이너리 wheel 형식에 대한 지원의 도입과 파이썬 패키지 색인을 통해 적어도 윈도우와 맥 OS X 용의 휠을 게시할 수 있게 됨에 따라, 사용자가 스스로 빌드하기 보다는 점차 사전 빌드된 확장들을 설치할 수 있게 되어, 이 문제는 시간이 지날수록 줄어들 것으로 예상합니다.

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

>>> 대화형 셸의 기본 파이썬 프롬프트. 인터프리터에서 대화형으로 실행될 수 있는 코드 예에서 자주 볼 수 있습니다.

... 들여쓰기 된 코드 블록의 코드를 입력할 때, 쌍을 이루는 구분자(괄호, 대괄호, 중괄호) 안에 코드를 입력할 때, 데코레이터 지정 후의 대화형 셸의 기본 파이썬 프롬프트.

2to3 파이썬 2.x 코드를 파이썬 3.x 코드로 변환하려고 시도하는 도구인데, 소스를 구문 분석하고 구문 분석 트리를 탐색해서 감지할 수 있는 대부분의 비호환성을 다룹니다.

2to3 는 표준 라이브러리에서 lib2to3 로 제공됩니다; 독립적으로 실행할 수 있는 스크립트는 Tools/scripts/2to3 로 제공됩니다. 2to3-reference 을 보세요.

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.

이 기능을 설명하는 **변수 어노테이션**, **함수 어노테이션**, **PEP 484**, **PEP 526**을 참조하세요.

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 차이와 **PEP 362**도 보세요.

asynchronous context manager (비동기 컨텍스트 관리자) `__aenter__()` 와 `__aexit__()` 메서드를 정의함으로써 `async with` 문에서 보이는 환경을 제어하는 객체. **PEP 492**로 도입되었습니다.

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.

비동기 *이터레이터* 인데 `__anext__()` 를 호출하면 어웨이터블 객체를 돌려주고, 이것은 다음 `yield` 표현식 까지 비동기 제너레이터 함수의 바디를 실행합니다.

각 `yield`는 일시적으로 처리를 중단하고, 그 위치의 (지역 변수들과 대기 중인 `try`-문들을 포함하는) 실행 상태를 기억합니다. 비동기 제너레이터 *이터레이터* 가 `__anext__()` 가 돌려주는 또 하나의 어웨이터블로 재개되면, 떠난 곳으로 복귀합니다. **PEP 492**와 **PEP 525**를 보세요.

asynchronous iterable (비동기 *이터러블*) `async for` 문에서 사용될 수 있는 객체. `__aiter__()` 메서드는 비동기 *이터레이터* 를 돌려줘야 합니다. **PEP 492**로 도입되었습니다.

asynchronous iterator (비동기 *이터레이터*) `__aiter__()` 와 `__anext__()` 메서드를 구현하는 객체. `__anext__` 는 어웨이터블 객체를 돌려줘야 합니다. `async for`는 `StopAsyncIteration` 예외가 발생할 때까지 비동기 *이터레이터*의 `__anext__()` 메서드가 돌려주는 어웨이터블을 팝니다. **PEP 492**로 도입되었습니다.

attribute (어트리뷰트) 점표현식을 사용하는 이름으로 참조되는 객체와 결합한 값. 예를 들어, 객체 *o*가 어트리뷰트 *a*를 가지면, *o.a*처럼 참조됩니다.

awaitable (어웨이터블) `await` 표현식에 사용할 수 있는 객체. 코루틴 이나 `__await__()` 메서드를 가진 객체가 될 수 있습니다. **PEP 492**를 보세요.

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

binary file (바이너리 파일) 바이트열류 객체들을 읽고 쓸 수 있는 파일 객체. 바이너리 파일의 예로는 바이너리 모드 ('rb', 'wb' 또는 'rb+') 로 열린 파일, `sys.stdin.buffer`, `sys.stdout.buffer`, `io.BytesIO` 와 `gzip.GzipFile` 의 인스턴스를 들 수 있습니다.

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

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.

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.

coercion (코어션) 같은 형의 두 인자를 수반하는 연산이 일어나는 동안, 한 형의 인스턴스를 다른 형으로 묵시적으로 변환하는 것. 예를 들어, `int(3.15)`는 실수를 정수 3으로 변환합니다. 하지만, `3+4.5`에서, 각 인자는 다른 형이고 (하나는 `int`, 다른 하나는 `float`), 둘을 더하기 전에 같은 형으로 변환해야 합니다. 그렇지 않으면 `TypeError`를 일으킵니다. 코어션 없이는, 호환되는 형들조차도 프로그래머가 같은 형으로 정규화해주어야 합니다, 예를 들어, 그냥 `3+4.5` 하는 대신 `float(3)+4.5`.

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 필요하다면, `cmath`를 사용합니다. 복소수의 활용은 꽤 수준 높은 수학적 기능입니다. 필요하다고 느끼지 못한다면, 거의 확실히 무시해도 좋습니다.

context manager (컨텍스트 관리자) `__enter__()`와 `__exit__()` 메서드를 정의함으로써 `with` 문에서 보이는 환경을 제어하는 객체. **PEP 343**으로 도입되었습니다.

context variable (컨텍스트 변수) 컨텍스트에 따라 다른 값을 가질 수 있는 변수. 이는 각 실행 스레드가 변수에 대해 다른 값을 가질 수 있는 스레드-로컬 저장소와 비슷합니다. 그러나, 컨텍스트 변수를 통해, 하나의 실행 스레드에 여러 컨텍스트가 있을 수 있으며 컨텍스트 변수의 주 용도는 동시성 비동기 태스크에서 변수를 추적하는 것입니다. `contextvars`를 참조하십시오.

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) Coroutines are a more generalized form of subroutines. Subroutines are entered at one point and exited at another point. Coroutines can be entered, exited, and resumed at many different points. They can be implemented with the `async def` statement. See also **PEP 492**.

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.

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(...):
    ...
f = staticmethod(f)

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

같은 개념이 클래스에도 존재하지만, 덜 자주 쓰입니다. 데코레이터에 대한 더 자세한 내용은 함수 정의와 클래스 정의의 설명서를 보면 됩니다.

descriptor (디스크립터) 메서드 `__get__()` 이나 `__set__()` 이나 `__delete__()` 를 정의하는 객체. 클래스 어트리뷰트가 디스크립터일 때, 어트리뷰트 조회는 특별한 연결 작용을 일으킵니다. 보통, `a.b`를 읽거나, 쓰거나, 삭제하는데 사용할 때, `a`의 클래스 디렉터리에서 `b`라고 이름 붙여진 객체를 찾습니다. 하지만 `b`가 디스크립터면, 해당하는 디스크립터 메서드가 호출됩니다. 디스크립터를 이해하는 것은 파이썬에 대한 깊은 이해의 열쇠인데, 함수, 메서드, 프로퍼티, 클래스 메서드, 스태틱 메서드, 슈퍼클래스 참조 등의 많은 기능의 기초를 이루고 있기 때문입니다.

디스크립터의 메서드들에 대한 자세한 내용은 `descriptors`에 나옵니다.

dictionary (딕셔너리) 임의의 키를 값에 대응시키는 연관 배열 (associative array). 키는 `__hash__()` 와 `__eq__()` 메서드를 갖는 모든 객체가 될 수 있습니다. 펄에서 해시라고 부릅니다.

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-views`를 보세요.

docstring (독스트링) 클래스, 함수, 모듈에서 첫 번째 표현식으로 나타나는 문자열 리터럴. 스위트가 실행될 때는 무시되지만, 컴파일러에 의해 인지되어 돌려쏜 클래스, 함수, 모듈의 `__doc__` 어트리뷰트로 삽입됩니다. 인트로스펙션을 통해 사용할 수 있으므로, 객체의 설명서를 위한 규범적인 장소입니다.

duck-typing (덕 타이핑) 올바른 인터페이스를 가졌는지 판단하는데 객체의 형을 보지 않는 프로그래밍 스타일; 대신, 단순히 메서드나 어트리뷰트가 호출되거나 사용됩니다 (“오리처럼 보이고 오리처럼 꺾꺾댄다면, 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 클래스로 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 문장들이 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) 하부 자원에 대해 파일 지향적 API(`read()` 나 `write()` 같은 메서드들)를 드러내는 객체. 만들어진 방법에 따라, 파일 객체는 실제 디스크 상의 파일이나 다른 저장 장치나 통신 장치 (예를 들어, 표준 입출력, 인-메모리 버퍼, 소켓, 파이프, 등등)에 대한 액세스를 중계할 수 있습니다. 파일 객체는 파일류 객체 (*file-like objects*)나 스트림 (*streams*) 이라고도 불립니다.

실제로는 세 종류의 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 객체의 비슷한 말.

finder (파인더) 임포트될 모듈을 위한 **로더**를 찾으려고 시도하는 객체.

파이썬 3.3. 이후로, 두 종류의 파인더가 있습니다: `sys.meta_path`와 함께 사용하는 **메타 경로 파인더**와 `sys.path_hooks`와 함께 사용하는 **경로 엔트리 파인더**.

더 자세한 내용은 **PEP 302**, **PEP 420**, **PEP 451**에 나옵니다.

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

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 받아 들일 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.

이 기능을 설명하는 **변수 어노테이션**과 **PEP 484**를 참조하세요.

__future__ 프로그래머가 현재 인터프리터와 호환되지 않는 새 언어 기능들을 활성화할 수 있도록 하는 가상 모듈.

`__future__` 모듈을 임포트하고 그 변수들의 값들을 구해서, 새 기능이 언제 처음으로 언어에 추가되었고, 언제부터 그것이 기본이 되는지 볼 수 있습니다:

```
>>> 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 이터레이터를 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 이터레이터가 재개되면, 떠난 곳으로 복귀합니다 (호출마다 새로 시작하는 함수와 대비됩니다).

generator expression (제너레이터 표현식) 이터레이터를 돌려주는 표현식. 루프 변수와 범위를 정의하는 `for` 절과 생략 가능한 `if` 절이 뒤에 붙는 일반 표현식처럼 보입니다. 결합한 표현식은 둘러싼 함수를 위한 값들을 만들어냅니다:

```
>>> 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.

GIL 전역 인터프리터 록 을 보세요.

global interpreter lock (전역 인터프리터 록) 한 번에 오직 하나의 스레드가 파이썬 바이트 코드를 실행하도록 보장하기 위해 CPython 인터프리터가 사용하는 메커니즘. (dict와 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.

하지만, 어떤 확장 모듈들은, 표준이나 제삼자 모두, 압축이나 해싱 같은 계산 집약적인 작업을 수행할 때는 GIL을 반납하도록 설계되었습니다. 또한, I/O를 할 때는 항상 GIL을 반납합니다.

(훨씬 더 미세하게 공유 데이터를 잠그는) “스레드에 자유로운(free-threaded)” 인터프리터를 만들고자 하는 과거의 노력은 성공적이지 못했는데, 혼란 단일 프로세서 경우의 성능 저하가 심하기 때문입니다. 이 성능 이슈를 극복하는 것은 구현을 훨씬 복잡하게 만들어서 유지 비용이 더 들어갈 것으로 여겨지고 있습니다.

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

hashable (해시 가능) 객체가 일생 그 값이 변하지 않는 해시값을 갖고 (`__hash__()` 메서드가 필요합니다), 다른 객체와 비교될 수 있으면 (`__eq__()` 메서드가 필요합니다), 해시 가능하다고 합니다. 같다고 비교되는 해시 가능한 객체들의 해시값은 같아야 합니다.

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 파이썬을 위한 통합 개발 환경 (Integrated Development Environment). IDLE은 파이썬의 표준 배포판에 따라오는 기초적인 편집기와 인터프리터 환경입니다.

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 (임포트) 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) 파이썬은 대화형 인터프리터를 갖고 있는데, 인터프리터 프롬프트에서 문장과 표현식을 입력할 수 있고, 즉각 실행된 결과를 볼 수 있다는 뜻입니다. 인자 없이 단지 `python`을 실행하세요 (컴퓨터의 주메뉴에서 선택하는 것도 가능할 수 있습니다). 새 아이디어를 검사하거나 모듈과 패키지를 들여다보는 매우 강력한 방법입니다 (`help(x)`를 기억하세요).

interpreted (인터프리티드) 바이트 코드 컴파일러의 존재 때문에 그 부분이 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 종료를 주된 원인은 실행되는 `__main__` 모듈이나 스크립트가 실행을 끝내는 것입니다.

iterable (이터러블) 멤버들을 한 번에 하나씩 돌려줄 수 있는 객체. 이터러블의 예로는 모든 (`list`, `str`, `tuple` 같은) 시퀀스 형들, `dict` 같은 몇몇 비 시퀀스 형들, [파일 객체들](#), `__iter__()` 나 [시퀀스](#) 개념을 구현하는 `__getitem__()` 메서드를 써서 정의한 모든 클래스의 객체들이 있습니다.

이터러블은 `for` 루프에 사용될 수 있고, 시퀀스를 필요로 하는 다른 많은 곳 (`zip()`, `map()`, ...) 에 사용될 수 있습니다. 이터러블 객체가 내장 함수 `iter()` 에 인자로 전달되면, 그 객체의 이터레이터를 돌려줍니다. 이 이터레이터는 값들의 집합을 한 번 거치는 동안 유효합니다. 이터러블을 사용할 때, 보통은 `iter()` 를 호출하거나, 이터레이터 객체를 직접 다룰 필요는 없습니다. `for` 문은 이것들을 여러 분을 대신해서 자동으로 해주는데, 루프를 도는 동안 이터레이터를 잡아들 이름 없는 변수를 만듭니다. [이터레이터](#), [시퀀스](#), [제너레이터](#)도 보세요.

iterator (이터레이터) 데이터의 스트림을 표현하는 객체. 이터레이터의 `__next__()` 메서드를 반복적으로 호출하면 (또는 내장 함수 `next()` 로 전달하면) 스트림에 있는 항목들을 차례대로 돌려줍니다. 더 이상의 데이터가 없을 때는 대신 `StopIteration` 예외를 일으킵니다. 이 지점에서, 이터레이터 객체는 소진되고, 이후의 모든 `__next__()` 메서드 호출은 `StopIteration` 예외를 다시 일으키기만 합니다. 이터레이터는 이터레이터 객체 자신을 돌려주는 `__iter__()` 메서드를 가질 것이 요구되기 때문에, 이터레이터는 이터러블이기도 하고 다른 이터러블들을 받아들이는 대부분의 곳에서 사용될 수 있습니다. 중요한 예외는 여러 번의 이터레이션을 시도하는 코드입니다. (`list` 같은) 컨테이너 객체는 `iter()` 함수로 전달하거나 `for` 루프에 사용할 때마다 새 이터레이터를 만듭니다. 이런 것을 이터레이터에 대해서 수행하려고 하면, 지난 이터레이션에 사용된 이미 소진된 이터레이터를 돌려줘서, 빈 컨테이너처럼 보이게 만듭니다.

`typeiter` 에 더 자세한 내용이 있습니다.

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.

키 함수를 만드는 데는 여러 방법이 있습니다. 예를 들어, `str.lower()` 메서드는 케이스 구분 없는 정렬을 위한 키 함수로 사용될 수 있습니다. 대안적으로, 키 함수는 `lambda` 표현식으로 만들 수도 있는데, 이런 식입니다: `lambda r: (r[0], r[2])`. 또한, `operator` 모듈은 세 개의 키 함수 생성자를 제공합니다: `attrgetter()`, `itemgetter()`, `methodcaller()`. 키 함수를 만들고 사용하는 법에 대한 예로 [Sorting HOW TO](#) 를 보세요.

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) 내장 파이썬 *시퀀스*. 그 이름에도 불구하고, 원소에 대한 액세스가 $O(1)$ 이기 때문에, 연결 리스트 (linked list) 보다는 다른 언어의 배열과 유사합니다.

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) 모듈을 로드하는 객체. `load_module()` 이라는 이름의 메서드를 정의해야 합니다. 로더는 보통 *파인더* 가 돌려줍니다. 자세한 내용은 [PEP 302](#) 를, 추상 베이스 클래스는 `importlib.abc.Loader` 를 보세요.

magic method (매직 메서드) 특수 메서드 의 비공식적인 비슷한 말.

mapping (매핑) 임의의 키 조회를 지원하고 Mapping 이나 MutableMapping 추상 베이스 클래스 에 지정된 메서드들을 구현하는 컨테이너 객체. 예로는 `dict`, `collections.defaultdict`, `collections.OrderedDict`, `collections.Counter` 를 들 수 있습니다.

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 디렉터리, 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 사용되었습니다.

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) 메서드 결정 순서는 조회하는 동안 멤버를 검색하는 베이스 클래스들의 순서입니다. 2.3 릴리스부터 파이썬 인터프리터에 사용된 알고리즘의 상세한 내용은 [The Python 2.3 Method Resolution Order](#) 를 보면 됩니다.

module (모듈) 파이썬 코드의 조직화 단위를 담당하는 객체. 모듈은 임의의 파이썬 객체들을 담는 이름 공간을 갖습니다. 모듈은 *임포트* 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.

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) The term “named tuple” applies to any type or class that inherits from tuple and whose indexable elements are also accessible using named attributes. The type or class may have other features as well.

Several built-in types are named tuples, including the values returned by `time.localtime()` and `os.stat()`. Another example is `sys.float_info`:

```

>>> 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 with the factory function `collections.namedtuple()`. The latter technique also adds 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(메서드에서) 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 유지보수성에 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 패키지와는 다릅니다.

모듈도 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) 지금은 모든 클래스 객체에 사용되고 있는 클래스 버전의 예전 이름. 초기의 파이썬 버전에서는, 오직 뉴스타일 클래스만 `__slots__`, 디스크립터, 프라퍼티, `__getattr__()`, 클래스 메서드, 스태틱 메서드와 같은 파이썬의 새롭고 다양한 기능들을 사용할 수 있었습니다.

object (객체) 상태(어트리뷰트나 값)를 갖고 동작(메서드)이 정의된 모든 데이터. 또한, 모든 뉴스타일 클래스의 최종적인 베이스 클래스입니다.

package (패키지) 서브 모듈들이나, 재귀적으로 서브 패키지들을 포함할 수 있는 파이썬 모듈. 기술적으로, 패키지는 `__path__` 어트리뷰트가 있는 파이썬 모듈입니다.

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 인자로 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. 파이썬은 위치-전용 매개변수를 정의하는 문법을 갖고 있지 않습니다. 하지만, 어떤 매개 함수들은 위치-전용 매개변수를 갖습니다(예를 들어, `abs()`).
- 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 파인더가 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](#) [혹](#)) 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 [혹](#)) `sys.path_hook` 리스트에 있는 콜러블인데, 특정 [경로 엔트리](#)에서 모듈을 찾는 법을 알고 있다면 [경로 엔트리 파인더](#)를 돌려줍니다.

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 포함 해서 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) 객체에 대한 참조의 개수. 객체의 참조 횟수가 0으로 떨어지면, 메모리가 반납됩니다. 참조 횟수 추적은 일반적으로 파이썬 코드에 노출되지는 않지만, *CPython* 구현의 핵심 요소입니다. `sys` 모듈은 특정 객체의 참조 횟수를 돌려주는 `getrefcount()` 을 정의합니다.

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

이름 공간 패키지 도 보세요.

__slots__ 클래스 내부의 선언인데, 인스턴스 어트리뷰트들을 위한 공간을 미리 선언하고 인스턴스 디렉터리를 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) `__getitem__()` 특수 메서드를 통해 정수 인덱스를 사용한 빠른 요소 액세스를 지원하고, 시퀀스의 길이를 돌려주는 `__len__()` 메서드를 정의하는 *이터러블*. 몇몇 내장 시퀀스들을 나열해보면, `list`, `str`, `tuple`, `bytes` 가 있습니다. `dict` 또한 `__getitem__()` 과 `__len__()` 을 지원하지만,

조회에 정수 대신 임의의 불변 키를 사용하기 때문에 시퀀스가 아니라 매핑으로 취급된다는 것에 주의해야 합니다.

`collections.abc.Sequence` 추상 베이스 클래스는 `__getitem__()` 과 `__len__()` 을 넘어서 훨씬 풍부한 인터페이스를 정의하는데, `count()`, `index()`, `__contains__()`, `__reversed__()` 를 추가합니다. 이 확장된 인터페이스를 구현한 형을 `register()` 를 사용해서 명시적으로 등록할 수 있습니다.

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

slice (슬라이스) 보통 시퀀스의 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.

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`.

text encoding (텍스트 인코딩) 유니코드 문자열을 바이트열로 인코딩하는 코덱.

text file (텍스트 파일) `str` 객체를 읽고 쓸 수 있는 파일 객체. 종종, 텍스트 파일은 실제로는 바이트 지향 데이터 스트림을 액세스하고 텍스트 인코딩을 자동 처리합니다. 텍스트 파일의 예로는 텍스트 모드 ('r' 또는 'w') 로 열린 파일, `sys.stdin`, `sys.stdout`, `io.StringIO` 의 인스턴스를 들 수 있습니다.

바이트열 객체를 읽고 쓸 수 있는 파일 객체에 대해서는 바이너리 파일도 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 걸쳐 쓸 수 있는데, 독스트링을 쓸 때 특히 쓸모 있습니다.

type (형) 파이썬 객체의 형은 그것이 어떤 종류의 객체인지를 결정합니다; 모든 객체는 형이 있습니다. 객체의 형은 `__class__` 어트리뷰트로 액세스할 수 있거나 `type(obj)` 로 얻을 수 있습니다.

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

형 에일리어스는 형 힌트를 단순화하는 데 유용합니다. 예를 들면:

```
from typing import List, Tuple

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:

```
from typing import List, Tuple

Color = Tuple[int, int, int]

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

이 기능을 설명하는 `typing`과 **PEP 484**를 참조하세요.

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

형 힌트는 선택 사항이며 파이썬에서 강제되지는 않습니다. 하지만, 정적 형 분석 도구에 유용하며 IDE의 코드 완성 및 리팩토링을 돕습니다.

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

이 기능을 설명하는 `typing`과 [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 힌트**로 사용됩니다: 예를 들어, 이 변수는 `int` 값을 가질 것으로 기대됩니다:

```
count: int = 0
```

변수 어노테이션 문법은 섹션 [annassign](#)에서 설명합니다.

이 기능을 설명하는 **함수 어노테이션**, [PEP 484](#) 및 [PEP 526](#)을 참조하세요.

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 코드**를 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

이 설명서에 관하여

이 설명서는 `reStructuredText` 소스에서 만들어진 것으로, 파이썬 설명서를 위해 특별히 제작된 문서 처리기인 `Sphinx` 를 사용했습니다.

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., 원래 파이썬 설명서 도구 집합의 작성자이자 많은 콘텐츠의 작가;
- `reStructuredText`와 `Docutils` 스위트를 만드는 `Docutils` 프로젝트.
- Fredrik Lundh, 그의 `Alternative Python Reference` 프로젝트에서 `Sphinx`가 많은 아이디어를 얻었습니다.

B.1 파이썬 설명서의 공헌자들

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!

역사와 라이선스

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.

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

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 라이선스는 GPL과 달리 여러분의 변경을 공개 소스로 만들지 않고 수정된 버전을 배포할 수 있게 합니다. GPL 호환 라이선스는 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

C.2.1 PSF LICENSE AGREEMENT FOR PYTHON 3.7.16

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),
→and
the Individual or Organization ("Licensee") accessing and otherwise using
→Python
3.7.16 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 3.7.16 alone or in any derivative
version, provided, however, that PSF's License Agreement and PSF's notice
→of
copyright, i.e., "Copyright © 2001-2023 Python Software Foundation; All
→Rights
Reserved" are retained in Python 3.7.16 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 3.7.16 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
3.7.16.
4. PSF is making Python 3.7.16 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 3.7.16 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.7.16
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT
→OF
MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.7.16, 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 3.7.16, 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

(다음 페이지에 계속)

(이전 페이지에서 계속)

third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.3 CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: <http://hdl.handle.net/1895.22/1013>."
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python

(다음 페이지에 계속)

(이전 페이지에서 계속)

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.3 포함된 소프트웨어에 대한 라이선스 및 승인

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

C.3.1 메르센 트위스터

_random 모듈은 <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> 에서 내려 받은 코드에 기반한 코드를 포함합니다. 다음은 원래 코드의 주석을 그대로 옮긴 것입니다:

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).

(다음 페이지에 계속)

(이전 페이지에서 계속)

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

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

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote
products derived from this software without specific prior written
permission.

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

Any feedback is very welcome.

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

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

C.3.2 소켓

socket 모듈은 `getaddrinfo()` 와 `getnameinfo()` 함수를 사용합니다. 이들은 WIDE Project, <http://www.wide.ad.jp/>, 에서 온 별도 소스 파일로 코딩되어 있습니다.

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.

(다음 페이지에 계속)

(이전 페이지에서 계속)

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

C.3.3 비동기 소켓 서비스

asynchat과 asyncore 모듈은 다음과 같은 주의 사항을 포함합니다:

```
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
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

prior permission.

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

C.3.5 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

uu 모듈은 다음과 같은 주의 사항을 포함합니다:

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

(다음 페이지에 계속)

(이전 페이지에서 계속)

supporting documentation, and that the name of Lance Ellinghouse not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:

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

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

C.3.8 test_epoll

test_epoll 모듈은 다음과 같은 주의 사항을 포함합니다:

```
Copyright (c) 2001-2006 Twisted Matrix Laboratories.

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

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

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

C.3.9 Select kqueue

select 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/

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

C double과 문자열 간의 변환을 위한 C 함수 dtoa 와 strtod 를 제공하는 파일 Python/dtoa.c 는 현재 <http://www.netlib.org/fp/> 에서 얻을 수 있는 David M. Gay의 같은 이름의 파일에서 파생되었습니다. 2009년 3월 16일에 받은 원본 파일에는 다음과 같은 저작권 및 라이선스 공지가 포함되어 있습니다:

```
/*****
 *
 * 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

모듈 `hashlib`, `posix`, `ssl`, `crypt` 는 운영 체제가 사용할 수 있게 하면 추가의 성능을 위해 OpenSSL 라이브러리를 사용합니다. 또한, 윈도우와 맥 OS X 파이썬 설치 프로그램은 OpenSSL 라이브러리 사본을 포함할 수 있으므로, 여기에 OpenSSL 라이선스 사본을 포함합니다:

LICENSE ISSUES

=====

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

OpenSSL License

```
/* =====
 * Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 * software must display the following acknowledgment:
 * "This product includes software developed by the OpenSSL Project
 * for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 * endorse or promote products derived from this software without
 * prior written permission. For written permission, please contact
 * openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 * nor may "OpenSSL" appear in their names without prior written
 * permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 * acknowledgment:
 * "This product includes software developed by the OpenSSL Project
 * for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```

* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to. The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code. The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 * notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in the
 * documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 * must display the following acknowledgement:
 * "This product includes cryptographic software written by
 * Eric Young (eay@cryptsoft.com)"
 * The word 'cryptographic' can be left out if the rouines from the library
 * being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 * the apps directory (application code) you must include an acknowledgement:
 * "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *

```

(다음 페이지에 계속)

(이전 페이지에서 계속)

```
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

C.3.13 expat

pyexpat 확장은 빌드를 `--with-system-expat` 로 구성하지 않는 한, 포함된 expat 소스 사본을 사용하여 빌드됩니다:

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

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

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

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

C.3.14 libffi

`_ctypes` 확장은 빌드를 `--with-system-libffi` 로 구성하지 않는 한, 포함된 `libffi` 소스 사본을 사용하여 빌드됩니다:

```
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.

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

_decimal 모듈은 빌드를 --with-system-libmpdec 로 구성하지 않는 한, 포함된 libmpdec 소스 사본을 사용하여 빌드됩니다:

```
Copyright (c) 2008-2016 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.

(다음 페이지에 계속)

(이전 페이지에서 계속)

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.

APPENDIX D

저작권

파이썬과 이 설명서는:

Copyright © 2001-2023 Python Software Foundation. All rights reserved.

Copyright © 2000 BeOpen.com. All rights reserved.

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

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

전체 라이선스 및 사용 권한 정보는 [역사](#)와 [라이선스](#) 에서 제공합니다.

Non-alphabetical

..., **11**
 2to3, **11**
 >>>, **11**
 __future__, **15**
 __slots__, **21**

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**
 bytecode (바이트 코드), **13**
 bytes-like object (바이트열류 객체), **12**

C

C-contiguous, **13**
 class (클래스), **13**
 class variable (클래스 변수), **13**
 coercion (코어션), **13**
 complex number (복소수), **13**
 context manager (컨텍스트 관리자), **13**
 context variable (컨텍스트 변수), **13**
 contiguous (연속), **13**
 coroutine (코루틴), **13**
 coroutine function (코루틴 함수), **13**

CPython, **13**

D

decorator (데코레이터), **13**
 descriptor (디스크립터), **14**
 dictionary (딕셔너리), **14**
 dictionary view (딕셔너리 뷰), **14**
 docstring (독스트링), **14**
 duck-typing (덕 타이핑), **14**

E

EAFP, **14**
 expression (표현식), **14**
 extension module (확장 모듈), **14**

F

f-string (f-문자열), **14**
 file object (파일 객체), **14**
 file-like object (파일류 객체), **15**
 finder (파인더), **15**
 floor division (정수 나눗셈), **15**
 Fortran contiguous, **13**
 function (함수), **15**
 function annotation (함수 어노테이션), **15**

G

garbage collection (가비지 수거), **15**
 generator, **15**
 generator (제너레이터), **15**
 generator expression, **15**
 generator expression (제너레이터 표현식), **15**
 generator iterator (제너레이터 이터레이터), **15**
 generic function (제네릭 함수), **16**
 GIL, **16**
 global interpreter lock (전역 인터프리터 락), **16**

H

hash-based pyc (해시 기반 pyc), **16**

hashable (해시 가능), 16

I

IDLE, 16

immutable (불변), 16

import path (임포트 경로), 16

importer (임porter), 16

importing (임포트), 16

interactive (대화형), 16

interpreted (인터프리터드), 16

interpreter shutdown (인터프리터 종료), 17

iterable (이터러블), 17

iterator (이터레이터), 17

K

key function (키 함수), 17

keyword argument (키워드 인자), 17

L

lambda (람다), 17

LBYL, 17

list (리스트), 18

list comprehension (리스트 컴프리헨션), 18

loader (로더), 18

M

magic

 method, 18

magic method (매직 메서드), 18

mapping (매핑), 18

meta path finder (메타 경로 파인더), 18

metaclass (메타 클래스), 18

method

 magic, 18

 special, 22

method (메서드), 18

method resolution order (메서드 결정 순서), 18

module (모듈), 18

module spec (모듈 스펙), 18

MRO, 18

mutable (가변), 18

N

named tuple (네임드 튜플), 18

namespace (이름 공간), 19

namespace package (이름 공간 패키지), 19

nested scope (중첩된 스코프), 19

new-style class (뉴스타일 클래스), 19

O

object (객체), 19

P

package (패키지), 19

parameter (매개변수), 19

path based finder (경로 기반 파인더), 20

path entry (경로 엔트리), 20

path entry finder (경로 엔트리 파인더), 20

path entry hook (경로 엔트리 훅), 20

path-like object (경로류 객체), 20

PEP, 20

portion (포션), 20

positional argument (위치 인자), 20

provisional API (잠정 API), 20

provisional package (잠정 패키지), 21

Python 3000 (파이썬 3000), 21

Pythonic (파이썬다운), 21

Q

qualified name (정규화된 이름), 21

R

reference count (참조 횟수), 21

regular package (정규 패키지), 21

S

sequence (시퀀스), 21

single dispatch (싱글 디스패치), 22

slice (슬라이스), 22

special

 method, 22

special method (특수 메서드), 22

statement (문장), 22

T

text encoding (텍스트 인코딩), 22

text file (텍스트 파일), 22

triple-quoted string (삼중 따옴표 된 문자열), 22

type (형), 22

type alias (형 에일리어스), 22

type hint (형 힌트), 22

U

universal newlines (유니버설 줄 넘김), 23

V

variable annotation (변수 어노테이션), 23

virtual environment (가상 환경), 23

virtual machine (가상 기계), 23

Y

파이썬 향상 제안

 PEP 1, 20

 PEP 238, 15

PEP 278, 23
PEP 302, 15, 18
PEP 343, 13
PEP 362, 12, 20
PEP 411, 21
PEP 420, 15, 19, 20
PEP 443, 16
PEP 451, 15
PEP 484, 11, 15, 22, 23
PEP 492, 12, 13
PEP 498, 14
PEP 519, 20
PEP 525, 12
PEP 526, 11, 23
PEP 3116, 23
PEP 3155, 21

Z

Zen of Python (파이썬 젠), 23