What's New in Python

출시 버전 ***3.7.17***

A. M. Kuchling

6월 28, 2023

Contents

편집자 Elvis Pranskevichus <elvis@magic.io>

이 기사에서는 파이썬 3.6과 비교하여 3.7의 새로운 기능에 관해 설명합니다. 파이썬 3.7은 2018년 6월 27일에 배포되었습니다. 자세한 내용은 changelog 를 참조하세요.

1 요약 – 배포 주요 사항

새로운 문법 기능:

- *PEP 563*, 형 어노테이션의 지연된 평가.

이전 버전과 호환되지 않는 문법 변경:

- `async`와 `await` 는 이제 예약 키워드입니다.

새 라이브러리 모듈:

- `contextvars`: *PEP 567 – 컨텍스트 변수*
- `dataclasses`: *PEP 557 – 데이터 클래스*
- *importlib.resources*

새로운 내장 기능:

- *PEP 553*, 새로운 `breakpoint()` 함수.

파이썬 데이터 모델 개선:

- *PEP 562*, 모듈 어트리뷰트에 대한 액세스의 사용자 정의.
- *PEP 560*, typing 모듈과 제네릭 형에 대한 코어 지원.
- dict 객체의 삽입 순서 보존 특성을 파이썬 언어 규격의 공식적인 일부로 선언했습니다.

표준 라이브러리의 현저한 개선:

- `asyncio` 모듈에 새 기능과 현저한 사용성 및 성능 개선 이 추가되었습니다.
- `time` 모듈은 나노초 해상도의 함수 지원을 얻었습니다.

CPython 구현 개선:

- ASCII를 기본 텍스트 인코딩으로 사용하지 않기:
  - *PEP 538*, 레거시 C 로케일 강제 변경
  - *PEP 540*, 강제 UTF-8 실행시간 모드
- *PEP 552*, 결정적 .pycs
- 새로운 개발 실행시간 모드
- *PEP 565*, 개선된 `DeprecationWarning` 처리

C API 개선 사항:

- *PEP 539*, 스레드-로컬 저장소를 위한 새로운 C API

설명서 개선 사항:

- *PEP 545*, 파이썬 설명서 번역
- 새로운 설명서 번역: 일본어, 프랑스어, 한국어.

이 배포는 많은 영역에서 주목할만한 성능 향상을 제공합니다. 최적화 섹션에 자세히 나와 있습니다.

이전 파이썬 배포와의 호환성에 영향을 줄 수 있는 변경 사항 목록은 파이썬 *3.7*로 이식하기 섹션을 참조하십시오.

2 새로운 기능

2.1 PEP 563 : 어노테이션의 지연된 평가

파이썬에서 형 힌트의 출현은 **PEP 3107**에서 추가되고 **PEP 526**에서 더욱 다듬어진 어노테이션의 두 가지 사용성 문제를 드러냈습니다:

- 어노테이션은 현재 스코프에서 이미 사용 가능한 이름 만 사용할 수 있습니다. 즉, 어떤 종류의 전방 참조도 지원하지 않았습니다; 그리고
- 소스 코드에 어노테이션을 붙이는 것은 파이썬 프로그램의 시작 시간에 악영향을 미쳤습니다.

이 두 가지 문제는 어노테이션 평가를 지연시키는 것으로 해결됩니다. 정의 시간에 어노테이션의 표현식을 실행하는 코드를 컴파일하는 대신, 컴파일러는 해당 표현식의 AST와 동등한 문자열 형식으로 어노테이션을 저장합니다. 필요하다면, 실행시간에 `typing.get_type_hints()` 를 사용하여 어노테이션을 해석 할 수 있습니다. 이것이 필수적이지 않은 일반적인 경우에는, 어노테이션을 싸게 저장할 수 있고 (짧은 문자열은 인터프리터에 의해 한 번만 만들어지기 때문입니다), 시작 시간을 더 빠르게 할 수 있습니다.

사용성 측면에서, 이제 어노테이션이 전방 참조를 지원하므로 다음 문법이 유효합니다:

```
class C:
    @classmethod
    def from_string(cls, source: str) -> C:
        ...

    def validate_b(self, obj: B) -> bool:
        ...

class B:
    ...
```

이 변경으로 인해 호환성이 깨지기 때문에, 파이썬 3.7에서 새 동작은 `__future__` 임포트를 사용하여 모듈별로 새로운 동작을 활성화해야 합니다:

```
from __future__ import annotations
```

It will become the default in Python 3.10.

더 보기:

PEP 563 – 어노테이션의 지연된 평가 Łukasz Langa가 작성하고 구현한 PEP.

2.2 PEP 538: 레거시 C 로케일 강제 변경

파이썬 3시리즈에서 진행 중인 과제는, 윈도우 이외의 플랫폼에서 기본 C 또는 POSIX 로케일 사용으로 인해 묵시적으로 가정되는 "7-비트 ASCII" 텍스트 인코딩을 처리하기 위한 적절한 기본 전략을 결정하는 것입니다.

새 `PYTHONCOERCECLOCALE` 환경 변수의 설명서에 설명되어있듯이, **PEP 538** 은 기본 인터프리터 명령행 인터페이스를 갱신하여, 사용 가능한 UTF-8 기반 로케일로 자동으로 강제 변경합니다. 이런 식으로 `LC_CTYPE` 을 자동 설정하는 것은, 핵심 인터프리터와 로케일을 인식하는 C 확장 (가령 `readline`) 모두 기본 텍스트 인코딩으로 ASCII 대신 UTF-8을 가정하게 된다는 뜻입니다.

PEP 11 의 플랫폼 지원 정의 역시 전체 텍스트 처리 지원을 적절히 구성된 비 ASCII 기반 로케일로 제한하도록 갱신되었습니다.

이 변경의 일부로, 이제 정의된 강제 변경 대상 로케일(현재 `C.UTF-8`, `C.utf8`, `UTF-8`)을 사용할 때, `stdin` 과 `stdout` 의 기본 에러 처리기는 (`strict` 대신) `surrogateescape` 입니다. `stderr` 의 기본 에러 처리기는 로케일에 관계없이 계속 `backslashreplace` 입니다.

로케일 강제 변경은 기본적으로 조용히 일어나지만, 로케일 관련 통합 문제를 디버깅하는 데 도움을 주기 위해 `PYTHONCOERCECLOCALE=warn` 를 설정해서 (`stderr` 로 직접 출력되는) 명시적 경고를 요청할 수 있습니다. 또한, 이 설정은 핵심 인터프리터가 초기화될 때 레거시 C 로케일이 활성 상태로 남아 있으면 파이썬 런타임이 경고를 하도록 만듭니다.

PEP 538 의 로케일 강제 변환이 (비 파이썬 응용 프로그램과 이전 버전의 파이썬을 실행하는 경우를 포함하는) 자식 프로세스뿐만 아니라 (GNU `readline` 같은) 확장 모듈에도 영향을 주는 장점이 있지만, 실행 중인 시스템에 적절한 대상 로케일이 있어야 한다는 단점이 있습니다. 적절한 대상 로케일을 사용할 수 없는 경우 (예를 들어, RHEL/CentOS 7에서 발생하듯이)를 더 잘 처리하기 위해, 파이썬 3.7은 *PEP 540: 강제 UTF-8 실행시간 모드* 또한 구현합니다.

더 보기:

PEP 538 – 레거시 C 로케일을 UTF-8 기반 로케일로 강제 변경 Nick Coghlan 이 작성하고 구현한 PEP.

2.3 PEP 540: 강제 UTF-8 실행시간 모드

새로운 `-X utf8` 명령행 옵션과 `PYTHONUTF8` 환경 변수를 사용하여 CPython *UTF-8* 모드 를 활성화할 수 있습니다.

UTF-8 모드에서, CPython은 로케일 설정을 무시하고 기본적으로 UTF-8 인코딩을 사용합니다. `sys.stdin` 과 `sys.stdout` 스트림의 에러 처리기는 `surrogateescape` 로 설정됩니다.

강제 UTF-8 모드는 임베디드 응용 프로그램의 로케일 설정을 변경하지 않고 임베디드 파이썬 인터프리터의 텍스트 처리 동작을 변경하는 데 사용할 수 있습니다.

PEP 540 의 UTF-8 모드는 실행 중인 시스템에서 사용할 수 있는 로케일에 관계없이 작동하는 이점이 있지만, (GNU `readline` 과 같은) 확장 모듈, 비 파이썬 응용 프로그램을 실행하는 자식 프로세스, 이전 버전의 파이썬을 실행하는 자식 프로세스에 영향을 주지 못하는 단점이 있습니다. 이러한 구성 요소와 통신 할 때 텍스트 데이터가 손상될 위험을 줄이기 위해 파이썬 3.7은 *PEP 540: 강제 UTF-8 실행시간 모드* 또한 구현합니다.

UTF-8 모드는 로케일이 `C` 또는 `POSIX` 이고, **PEP 538** 로케일 강제 변환이 UTF-8 기반 대안으로의 변경에 실패할 때 (그 실패가 `PYTHONCOERCECLOCALE=0` 설정 때문이든, `LC_ALL` 설정 때문이든, 적절한 대상 로케일이 없기 때문이든 무관하게) 기본적으로 활성화됩니다.

더 보기:

PEP 540 – 새로운 UTF-8 모드 추가 Victor Stinner가 작성하고 구현한 PEP

2.4 PEP 553: 내장 breakpoint()

파이썬 3.7에는 파이썬 디버거에 진입하는 쉽고 일관된 방식을 제공하는 새로운 내장 `breakpoint()` 함수가 포함되어 있습니다.

내장 `breakpoint()` 는 `sys.breakpointhook()` 을 호출합니다. 기본적으로, 후자는 `pdb`를 임포트 한 다음 `pdb.set_trace()` 를 호출합니다. 하지만, `sys.breakpointhook()` 을 여러분이 선택한 함수에 연결하면, `breakpoint()` 는 임의의 디버거에 진입할 수 있습니다. 또한, 환경 변수 `PYTHONBREAKPOINT` 를 여러분이 선택한 디버거의 콜러블로 설정할 수 있습니다. 내장 *breakpoint()*' 를 완전히 비활성화하려면 `PYTHONBREAKPOINT=0` 를 설정하십시오.

더 보기:

PEP 553 – **내장 breakpoint()** Barry Warsaw가 작성하고 구현한 PEP

2.5 PEP 539: 스레드-로컬 저장소를 위한 새로운 C API

파이썬은 스레드 로컬 저장소 지원을 위한 C API를 제공하지만; 기존 스레드 로컬 저장소 (TLS) API 는 모든 플랫폼에서 TLS 키로 `int`를 사용합니다. 이것은 공식적으로 지원되는 플랫폼에서는 일반적으로 문제가 되지 않지만, POSIX를 준수하지도 실용적인 의미에서 이식성이 있지도 않습니다.

PEP 539 는 CPython에 새로운 스레드 특정 저장소 (TSS) API 를 제공해서 이를 변경하는데, CPython 인터프리터 내에서 기존 TLS API의 사용을 대체하는 동시에 기존 API를 폐지합니다. TSS API는 TSS 키를 나타내는데 `int` 대신 `Py_tss_t`라는 새로운 형을 사용합니다. 이 형은 하부 TLS 구현에 따라 달라질 수 있는 불투명 한 형입니다. 그래서 네이티브 TLS 키가 `int`로 안전하게 캐스팅될 수 없는 방식으로 정의된 플랫폼에서 CPython을 빌드 할 수 있게 합니다.

네이티브 TLS 키가 `int`로 안전하게 캐스팅될 수 없는 방식으로 정의된 플랫폼에서는, 기존 TLS API의 모든 함수는 작동하지 않고 즉시 실패를 반환합니다. 이는 이전 API가 신뢰성 있게 사용될 수 없는 플랫폼에서 지원되지 않으며, 이러한 지원을 추가하기 위한 노력이 없을 것을 분명하게 나타냅니다.

더 보기:

PEP 539 – **CPython의 스레드-로컬 저장소를 위한 새로운 C-API** Erik M. Bray가 작성하고 Masayuki Yamamoto가 구현한 PEP.

2.6 PEP 562: 모듈 어트리뷰트에 대한 액세스 사용자 정의

파이썬 3.7은 모듈에, 발견되지 않는 어트리뷰트마다 호출되는 `__getattr__()` 을 정의할 수 있도록 합니다. 이제 모듈에 `__dir__()` 도 정의할 수 있게 되었습니다.

이것이 유용한 전형적인 예는 모듈 어트리뷰트 폐지와 지연 로딩입니다.

더 보기:

PEP 562 – **모듈 `__getattr__` 과 `__dir__`** Ivan Levkivskyi가 작성하고 구현한 PEP

2.7 PEP 564: 나노초 해상도의 새로운 시간 함수

현대 시스템의 시계 해상도는 `time.time()` 함수와 그 변형이 반환하는 부동 소수점 숫자의 제한된 정밀도를 초과 할 수 있습니다. **PEP 564** 는 기존 타이머 함수의 새로운 "나노 초" 변형을 `time` 모듈에 추가합니다:

- `time.clock_gettime_ns()`
- `time.clock_settime_ns()`
- `time.monotonic_ns()`
- `time.perf_counter_ns()`
- `time.process_time_ns()`
- `time.time_ns()`

새로운 함수는 나노초의 수를 정숫값으로 반환합니다.

<https://www.python.org/dev/peps/pep-0564/#annex-clocks-resolution-in-python>'_ 의 측정에 의하면 리눅스와 윈도우에서 `time.time_ns()` 의 해상도는 `time.time()` 보다 약 3배 정밀합니다.

더 보기:

PEP 564 – 나노초 해상도의 새로운 시간 함수 추가 Victor Stinner가 작성하고 구현한 PEP

2.8 PEP 565: `__main__` 의 DeprecationWarning 표시

`DeprecationWarning` 의 기본 처리 방식이 변경되어, 이러한 경고가 다시 한번 기본적으로 표시됩니다. 하지만, 이를 발생시킨 코드가 `__main__` 모듈에서 직접 실행될 때만 표시됩니다. 결과적으로, 단일 파일 스크립트 개발자와 파이썬을 대화식으로 사용하는 개발자는 사용하는 API에 대한 폐지 경고를 다시 보게 되지만, 임포트되는 응용 프로그램, 라이브러리, 프레임웍 모듈에서 발생하는 폐지 경고는 계속 기본적으로 숨겨집니다.

이 변경으로 인해, 이제 표준 라이브러리는 개발자가 세 가지 다른 폐지 경고 동작 중 하나를 선택할 수 있도록 합니다:

- `FutureWarning`: 기본적으로 항상 표시됩니다. 응용 프로그램 최종 사용자를 대상으로 하는 경고로 권장됩니다 (예, 폐지된 응용 프로그램 구성 설정).
- `DeprecationWarning`: 기본적으로 `__main__` 과 테스트 실행 시에 표시됩니다. 버전 업그레이드가 동작 변경이나 에러를 일으킬 수 있어서, 다른 파이썬 개발자를 대상으로 하는 경고로 권장됩니다.
- `PendingDeprecationWarning`: 기본적으로 테스트 실행 시에만 표시됩니다. 향후 버전 업그레이드가 경고 범주를 `DeprecationWarning` 이나 `FutureWarning`으로 변경하게 될 경우를 위한 것입니다.

이전에는 `DeprecationWarning` 과 `PendingDeprecationWarning` 둘 다 테스트를 실행할 때만 볼 수 있었습니다. 주로 단일 파일 스크립트를 작성하거나 대화식으로 파이썬을 사용하는 개발자는 사용된 API가 호환되지 않는 방식으로 변경된 것을 보고 놀랄 수 있었습니다.

더 보기:

PEP 565 – `__main__` 의 DeprecationWarning 표시 Nick Coghlan이 작성하고 구현한 PEP

2.9 PEP 560: typing 모듈과 제네릭 형에 대한 코어 지원

처음에는 PEP 484 가 핵심 CPython 인터프리터의 어떤 변경도 도입하지 않도록 설계되었습니다. 이제 형 힌트와 typing 모듈이 커뮤니티에서 광범위하게 사용되므로, 이 제한이 제거됩니다. PEP는 두 개의 특수 메서드 __class_getitem__() 과 __mro_entries__ 를 소개합니다. 이 메서드는 이제 typing 에 있는 대부분 클래스와 특수 구조물에서 사용됩니다. 그 결과로, 형과 관련된 여러 연산의 속도가 최대 7배까지 증가했고, 제네릭 형은 메타 클래스 충돌 없이 사용할 수 있으며, typing 모듈의 몇 가지 오랜 버그가 해결되었습니다.

더 보기:

PEP 560 – typing 모듈과 제네릭 형에 대한 코어 지원 Ivan Levkivskyi가 작성하고 구현한 PEP

2.10 PEP 552: 해시 기반 .pyc 파일

파이썬은 전통적으로 바이트 코드 캐시 파일(즉, .pyc 파일)의 최신성을 검사하기 위해, 소스 메타 데이터(최종 수정 타임스탬프와 크기)를 캐시 파일이 만들어질 때 헤더에 저장된 소스 메타 데이터와 비교했습니다. 효과적이지만, 이 무효화 방법에는 단점이 있습니다. 파일 시스템 타임스탬프가 너무 거친 경우, 파이썬은 소스 변경을 놓칠 수 있어 사용자 혼란을 낳을 수 있습니다. 또한, 캐시 파일에 타임스탬프를 갖는 것은 빌드 재현성 과 콘텐츠 기반 빌드 시스템에서 문제가 됩니다.

PEP 552 는 소스 타임스탬프 대신 소스 파일의 해시가 소스 타임스탬프 대신 무효화에 사용될 수 있도록 pyc 형식을 확장합니다. 이러한 .pyc 파일을 "해시 기반"이라고 합니다. 기본적으로, 파이썬은 여전히 타임스탬프 기반 무효화를 사용하며 실행 시간에 해시 기반 .pyc 파일을 생성하지 않습니다. 해시 기반 .pyc 파일은 py_compile 또는 compileall로 만들 수 있습니다.

해시 기반 .pyc 파일에는 두 가지 변형이 있습니다: 검사형(checked)과 비검사형(unchecked). 파이썬은 검사형 해시 기반 .pyc 파일을 실행시간에 해당 소스 파일에 대해 유효성을 검사하지만, 비검사형 해시 기반 pyc에 대해서는 확인하지 않습니다. 비검사형 해시 기반 .pyc 파일은 파이썬 외부의 시스템(가령 빌드 시스템)이 .pyc 파일을 최신 상태로 유지하는 책임을 지는 환경에서 유용한 성능 최적화입니다.

자세한 정보는 pyc-invalidation 를 보십시오.

더 보기:

PEP 552 – 결정적 pycs Benjamin Peterson이 작성하고 구현한 PEP

2.11 PEP 545: 파이썬 설명서 번역

PEP 545 는 파이썬 설명서 번역을 만들고 유지하는 과정을 설명합니다.

세 가지 새로운 번역이 추가되었습니다:

- 일본어: https://docs.python.org/ja/
- 프랑스어: https://docs.python.org/fr/
- 한국어: https://docs.python.org/ko/

더 보기:

PEP 545 – 파이썬 설명서 번역 Julien Palard, Inada Naoki 및 Victor Stinner가 작성하고 구현한 PEP.

2.12 개발 실행시간 모드: -X dev

새 `-X dev` 명령행 옵션이나 새 `PYTHONDEVMODE` 환경 변수를 사용하여 CPython의 개발 모드 를 활성화할 수 있습니다. 개발 모드에 있을 때, CPython은 기본적으로 활성화되기에는 너무 비싼 추가적인 실행 시간 검사를 수행합니다. 이 모드의 효과에 대한 자세한 설명은 `-X dev` 설명서를 보십시오.

3 기타 언어 변경

- An `await` expression and comprehensions containing an `async for` clause were illegal in the expressions in formatted string literals due to a problem with the implementation. In Python 3.7 this restriction was lifted.
- 이제 함수에 255개 이상의 인자를 전달할 수 있고, 함수는 255개 이상의 매개변수를 가질 수 있습니다. (Contributed by Serhiy Storchaka in bpo-12844 and bpo-18896.)
- 이제 `bytes.fromhex()` 와 `bytearray.fromhex()` 는 스페이스뿐만 아니라 모든 ASCII 공백을 무시합니다. (Contributed by Robert Xiao in bpo-28927.)
- `str`, `bytes`, `bytearray` 는 새 `isascii()` 메서드에 대한 지원을 얻었습니다. 이 메서드를 사용하여 문자열이나 바이트열에 오직 ASCII 문자만 들어 있는지 검사할 수 있습니다. (Contributed by INADA Naoki in bpo-32677.)
- 이제 `ImportError` 는 `from ... import ...` 가 실패할 때 모듈 이름과 모듈 `__file__` 경로를 출력합니다. (Contributed by Matthias Bussonnier in bpo-29546.)
- 이제 하위 모듈을 이름에 연결하는 절대 임포트가 수반되는 순환 임포트가 지원됩니다. (Contributed by Serhiy Storchaka in bpo-30024.)
- 이제 `object.__format__(x, '')` 는 `format(str(self), '')` 대신에 `str(x)` 과 동등합니다. (Contributed by Serhiy Storchaka in bpo-28974.)
- 이제 스택 트레이스의 동적 생성을 더욱 잘 지원하기 위해, 파이썬 코드에서 `types.TracebackType` 의 인스턴스를 만들 수 있고, 트레이스백 의 `tb_next` 어트리뷰트에 쓸 수 있습니다. (Contributed by Nathaniel J. Smith in bpo-30579.)
- `-m` 스위치를 사용할 때, 이제 `sys.path[0]` 를 빈 디렉터리로 남겨두기보다는 전체 시작 디렉터리 경로로 확장됩니다 (빈 디렉터리를 남겨두면, 임포트가 일어날 때의 현재 작업 디렉터리로부터의 임포트가 가능해집니다) (Contributed by Nick Coghlan in bpo-33053.)
- 새 `-X importtime` 옵션 또는 `PYTHONPROFILEIMPORTTIME` 환경 변수는 각 모듈 임포트의 타이밍을 보여주기 위해 사용될 수 있습니다. (Contributed by Victor Stinner in bpo-31415.)

4 새 모듈

4.1 contextvars

새 `contextvars` 모듈과 새 C API 집합은 컨텍스트 변수 에 대한 지원을 도입합니다. 컨텍스트 변수는 개념적으로 스레드-로컬 변수와 유사합니다. TLS와 달리, 컨텍스트 변수는 비동기 코드를 올바르게 지원합니다.

`asyncio`와 `decimal` 모듈은 바로 사용할 수 있도록 컨텍스트 변수를 사용하고 지원하도록 업데이트되었습니다. 특히 활성 십진 컨텍스트는 이제 컨텍스트 변수에 저장되므로, 십진수 연산이 비동기 코드에서 올바른 컨텍스트에서 작동 할 수 있습니다.

더 보기:

PEP 567 – 컨텍스트 변수 Yury Selivanov가 작성하고 구현한 PEP

4.2 dataclasses

새 `dataclass()` 데코레이터는 데이터 클래스 를 선언하는 방법을 제공합니다. 데이터 클래스는 클래스 변수 어노테이션을 사용하여 어트리뷰트를 기술합니다. 생성자와 `__repr__()`, `__eq__()`, `__hash__()` 와 같은 다른 매직 메서드가 자동으로 생성됩니다.

예:

```
@dataclass
class Point:
    x: float
    y: float
    z: float = 0.0

p = Point(1.5, 2.5)
print(p)   # produces "Point(x=1.5, y=2.5, z=0.0)"
```

더 보기:

PEP 557 – 데이터 클래스 Eric V. Smith가 작성하고 구현한 PEP

4.3 importlib.resources

새 `importlib.resources` 모듈은 여러 개의 새로운 API와 하나의 새로운 ABC를 제공하여, 패키지 내부의 리소스 에 접근하고 읽는 것을 지원합니다. 리소스는 대략 패키지 안의 파일과 비슷하지만, 물리적인 파일 시스템에 있는 실제 파일 일 필요는 없습니다. 모듈 로더는 이 새로운 API를 지원하기 위해 `importlib.abc.ResourceReader` 인스턴스를 반환하는 `get_resource_reader()` 함수를 제공 할 수 있습니다. 내장 파일 경로 로더와 zip 파일 로더는 모두 이것을 지원합니다.

Contributed by Barry Warsaw and Brett Cannon in bpo-32248.

더 보기:

importlib_resources – 이전 파이썬 버전을 위한 PyPI 백 포트.

5 개선된 모듈

5.1 argparse

새로운 `ArgumentParser.parse_intermixed_args()` 메서드는 옵션과 위치 인자의 혼합을 허용합니다. (Contributed by paul.j3 in bpo-14191.)

5.2 asyncio

`asyncio` 모듈은 많은 새로운 기능과 사용성 및 성능 개선이 추가되었습니다. 주목할만한 변경 사항은 다음과 같습니다:

- 새로운 잠정 `asyncio.run()` 함수는 자동으로 이벤트 루프를 생성하고 파괴함으로써 동기 코드에서 코루틴을 실행하는 데 사용될 수 있습니다. (Contributed by Yury Selivanov in bpo-32314.)
- asyncio 는 `contextvars` 에 대한 지원을 얻었습니다. `loop.call_soon()`, `loop.call_soon_threadsafe()`, `loop.call_later()`, `loop.call_at()` 및 `Future.add_done_callback()` 은 새로운 선택적 키워드-전용 *context* 매개 변수를 갖습니다. `Tasks`

는 이제 자신의 컨텍스트를 자동으로 추적합니다. 자세한 정보는 **PEP 567**을 보십시오. (Contributed by Yury Selivanov in bpo-32436.)

- 새로운 `asyncio.create_task()` 함수가 `asyncio.get_event_loop().create_task()` 의 손쉬운 방법으로 추가되었습니다. (Contributed by Andrew Svetlov in bpo-32311.)
- 새로운 `loop.start_tls()` 메서드를 사용하여 기존 연결을 TLS로 업그레이드할 수 있습니다. (Contributed by Yury Selivanov in bpo-23749.)
- 새로운 `loop.sock_recv_into()` 메서드는 데이터를 직접 소켓에서 제공된 버퍼로 읽어 들여 데이터 복사를 줄일 수 있게 합니다. (Contributed by Antoine Pitrou in bpo-31819.)
- 새로운 `asyncio.current_task()` 함수는 현재 실행 중인 `Task` 인스턴스를 반환하고, 새 `asyncio.all_tasks()` 함수는 주어진 루프에 있는 모든 기존 `Task` 인스턴스 집합을 반환합니다. `Task.current_task()` 와 `Task.all_tasks()` 메서드는 폐지되었습니다. (Contributed by Andrew Svetlov in bpo-32250.)
- 새로운 잠정적 `BufferedProtocol` 클래스는 수신 버퍼를 수동으로 제어하여 스트리밍 프로토콜을 구현할 수 있게 합니다. (Contributed by Yury Selivanov in bpo-32251.)
- 새로운 `asyncio.get_running_loop()` 함수는 현재 실행 중인 루프를 반환하고, 루프가 실행되고 있지 않으면 `RuntimeError` 를 발생시킵니다. 이는 아무것도 실행되고 있지 않으면 새로 만드는 `asyncio.get_event_loop()` 과는 대조적입니다. (Contributed by Yury Selivanov in bpo-32269.)
- 새로운 `StreamWriter.wait_closed()` 코루틴 메서드는 스트림 작성자가 닫힐 때까지 기다릴 수 있습니다. 새로운 `StreamWriter.is_closing()` 메서드를 사용하여 작성기가 닫히고 있는지 판별할 수 있습니다. (Contributed by Andrew Svetlov in bpo-32391.)
- 새로운 `loop.sock_sendfile()` 코루틴 메서드는 가능한 경우 `os.sendfile` 을 사용하여 파일을 전송할 수 있도록 합니다. (Contributed by Andrew Svetlov in bpo-32410.)
- 새로운 `Future.get_loop()` 와 `Task.get_loop()` 메서드는 태스크 나 퓨처가 만들어진 루프의 인스턴스를 돌려줍니다. `Server.get_loop()` 는 `asyncio.Server` 객체에 대해서도 같은 일을 할 수 있도록 합니다. (Contributed by Yury Selivanov in bpo-32415 and Srinivas Reddy Thatiparthy in bpo-32418.)
- 이제 `asyncio.Server` 의 인스턴스가 어떻게 시작되는지 제어할 수 있습니다. 이전에는 서버를 만들 때 즉시 서버가 시작되었습니다. 새로운 *start_serving* 키워드 인자는 `loop.create_server()` 와 `loop.create_unix_server()` 뿐만 아니라 `Server.start_serving()` 와 `Server.serve_forever()` 에도 사용되어 서버 인스턴스를 만드는 것과 시작시키는 것을 분리할 수 있습니다. 새로운 `Server.is_serving()` 메서드는 서버가 실행 중이면 `True` 를 돌려줍니다. `Server` 객체는 이제 비동기 컨텍스트 관리자입니다:

```
srv = await loop.create_server(...)

async with srv:
    # some code

# At this point, srv is closed and no longer accepts new connections.
```

(Contributed by Yury Selivanov in bpo-32662.)

- `loop.call_later()` 가 반환하는 콜백 객체는 예약된 절대 콜백 타임스탬프를 돌려주는 새 `when()` 메서드를 얻었습니다. (Contributed by Andrew Svetlov in bpo-32741.)
- `loop.create_datagram_endpoint()` 메서드는 유닉스 소켓 지원을 얻었습니다. (Contributed by Quentin Dawans in bpo-31245.)
- `asyncio.open_connection()`, `asyncio.start_server()` 함수, `loop.create_connection()`, `loop.create_server()`, `loop.create_accepted_socket()` 메서드와 해당 유닉스 소켓 변형은 이제 *ssl_handshake_timeout* 키워드 인자를 받아들입니다. (Contributed by Neil Aspinall in bpo-29970.)

- 새로운 `Handle.cancelled()` 메서드는 콜백이 취소된 경우 `True` 를 돌려줍니다. (Contributed by Marat Sharafutdinov in bpo-31943.)
- asyncio 소스는 `async/await` 구문을 사용하도록 변환되었습니다. (Contributed by Andrew Svetlov in bpo-32193.)
- 새로운 `ReadTransport.is_reading()` 메서드를 사용하여 전송계층의 읽기 상태를 판별 할 수 있습니다. 또한 `ReadTransport.resume_reading()` 과 `ReadTransport.pause_reading()` 은 이제 멱등적(idempotent) 입니다. (Contributed by Yury Selivanov in bpo-32356.)
- 소켓 경로를 받아들이는 루프 메서드에는 이제 경로류 객체 를 전달할 수 있습니다. (Contributed by Yury Selivanov in bpo-32066.)
- `asyncio` 에서 리눅스의 TCP 소켓은 이제 기본적으로 `TCP_NODELAY` 플래그가 설정된 상태로 만들어 집니다. (Contributed by Yury Selivanov and Victor Stinner in bpo-27456.)
- 취소된 작업에서 발생하는 예외는 더 로깅 되지 않습니다. (Contributed by Yury Selivanov in bpo-30508.)
- 새로운 `WindowsSelectorEventLoopPolicy` 와 `WindowsProactorEventLoopPolicy` 클래스. (Contributed by Yury Selivanov in bpo-33792.)

몇몇 `asyncio` API는 페지되었습니다.

5.3 binascii

`b2a_uu()` 함수는 선택적 *backtick* 키워드 인자를 받아들입니다. 참일 때, 0은 스페이스 대신에 `'`'` 로 표현됩니다. (Contributed by Xiang Zhang in bpo-30103.)

5.4 calendar

`HTMLCalendar` 클래스는 생성된 HTML 달력에서 CSS 클래스의 개별화를 쉽게 해주는 새로운 클래스 어트리뷰트를 갖습니다. (Contributed by Oz Tiram in bpo-30095.)

5.5 collections

`collections.namedtuple()` 은 이제 기본값을 지원합니다. (Contributed by Raymond Hettinger in bpo-32320.)

5.6 compileall

`compileall.compile_dir()` 는 새로운 *invalidation_mode* 매개 변수를 갖습니다. 이 매개 변수는 해시 기반 *.pyc* 무효화 를 활성화하는데 사용할 수 있습니다. 무효화 모드는 새로운 `--invalidation-mode` 인자를 사용하여 명령행에서 지정할 수도 있습니다. (Contributed by Benjamin Peterson in bpo-31650.)

5.7 concurrent.futures

`ProcessPoolExecutor` 와 `ThreadPoolExecutor` 는 이제 새로운 *initializer* 및 *initargs* 생성자 인자를 지원합니다. (Contributed by Antoine Pitrou in bpo-21423.)

`ProcessPoolExecutor` 는 새로운 *mp_context* 인자를 통해 multiprocessing 컨텍스트를 받아들일 수 있습니다. (Contributed by Thomas Moreau in bpo-31540.)

5.8 contextlib

새로운 `nullcontext()` 는 `ExitStack` 보다 더 간단하고 빠른 no-op 컨텍스트 관리자입니다. (Contributed by Jesse-Bakker in bpo-10049.)

새로운 `asynccontextmanager()`, `AbstractAsyncContextManager`, `AsyncExitStack` 가 추가되어 동기 대응물을 보완합니다. (Contributed by Jelle Zijlstra in bpo-29679 and bpo-30241, and by Alexander Mohr and Ilya Kulakov in bpo-29302.)

5.9 cProfile

`cProfile` 명령행은 이제 스크립트 경로의 대안으로 `-m module_name` 을 허용합니다. (Contributed by Sanyam Khurana in bpo-21862.)

5.10 crypt

`crypt` 모듈은 이제 블로피시 해싱 방법을 지원합니다. (Contributed by Serhiy Storchaka in bpo-31664.)

`mksalt()` 함수는 이제 해싱을 위한 라운드 수를 지정할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-31702.)

5.11 datetime

새로운 `datetime.fromisoformat()` 메서드는 `datetime.isoformat()` 이 출력하는 형식의 문자열로부터 `datetime` 객체를 생성합니다. (Contributed by Paul Ganssle in bpo-15873.)

`tzinfo` 클래스는 이제 분보다 작은 오프셋을 지원합니다. (Contributed by Alexander Belopolsky in bpo-5288.)

5.12 dbm

`dbm.dumb` 는 이제 읽기 전용 파일 읽기를 지원하며 변경되지 않았을 때 더는 색인 파일을 쓰지 않습니다.

5.13 decimal

`decimal` 모듈은 이제 십진 컨텍스트를 저장하는데 컨텍스트 변수 를 사용한다. (Contributed by Yury Selivanov in bpo-32630.)

5.14 dis

`dis()` 함수는 이제 중첩된 코드 객체 (컴프리헨션, 제너레이터 표현식 및 중첩된 함수의 코드와 중첩된 클래스를 만드는데 사용되는 코드)를 역 어셈블 할 수 있습니다. 해체 재귀의 최대 깊이는 새로운 * depth * 매개변수에 의해 제어됩니다. (Contributed by Serhiy Storchaka in bpo-11822.)

5.15 distutils

`README.rst` 는 이제 distutils 표준 README 목록에 포함됩니다. 그래서 소스 배포판에 포함됩니다. (Contributed by Ryan Gonzalez in bpo-11913.)

5.16 enum

`Enum` 은 새로운 `_ignore_` 클래스 프로퍼티를 얻었는데, 열거형 멤버가 되어서는 안 되는 프로퍼티의 이름들을 나열할 수 있도록 합니다. (Contributed by Ethan Furman in bpo-31801.)

파이썬 3.8에서 `Enum` 클래스에서 열거 이외의 객체를 포함 검사하려고 하면 `TypeError` 가 발생합니다 (예 `1 in Color`). 마찬가지로 `Flag` 멤버에서 플래그가 아닌 객체를 포함 검사하려고 하면 `TypeError` 를 일으킵니다 (예 `1 in Perm.RW`); 현재는, 두 연산 모두 대신 `False` 가 반환되며, 폐지되었습니다. (Contributed by Ethan Furman in bpo-33217.)

5.17 functools

`functools.singledispatch()` 는 이제 형 어노테이션을 사용해서 구현을 등록할 수 있습니다. (Contributed by Łukasz Langa in bpo-32227.)

5.18 gc

새로운 `gc.freeze()` 함수는 가비지 수집기가 추적 한 모든 객체를 고정하고 미래의 수집에서 그것들을 제외합니다. POSIX `fork()` 호출 전에 사용하면 GC를 쓰기 전에 복사(copy-on-write) 친화적으로 만들고 수집 속도를 높일 수 있습니다. 새로운 `gc.unfreeze()` 함수는 이 작업을 되돌립니다. 또한, `gc.get_freeze_count()` 를 사용하여 고정된 객체의 수를 얻을 수 있습니다. (Contributed by Li Zekun in bpo-31558.)

5.19 hmac

`hmac` 모듈은 이제 최적화된 단일 호출 `digest()` 함수를 제공합니다. `HMAC()` 보다 최대 3배 빠릅니다. (Contributed by Christian Heimes in bpo-32433.)

5.20 http.client

`HTTPConnection` 과 `HTTPSConnection` 은 이제 향상된 업로드 처리량을 위해 새로운 *blocksize* 인자를 지원합니다. (Contributed by Nir Soffer in bpo-31945.)

5.21 http.server

`SimpleHTTPRequestHandler` 는 이제 HTTP `If-Modified-Since` 헤더를 지원합니다. 헤더에 지정된 시간 이후에 대상 파일이 수정되지 않았으면 서버는 304 응답 상태를 돌려줍니다. (Contributed by Pierre Quentel in bpo-29654.)

`SimpleHTTPRequestHandler` 는 새로운 `--directory` 명령행 인자와 더불어 새로운 *directory* 인자를 받아들입니다. 이 매개 변수를 사용하면 서버는 지정된 디렉터리를 제공하는데, 기본적으로는 현재 작업 디렉터리를 사용합니다. (Contributed by Stéphane Wirtel and Julien Palard in bpo-28707.)

새로운 `ThreadingHTTPServer` 클래스는 `ThreadingMixin` 을 사용하여 요청을 처리하는 데 스레드를 사용합니다. `http.server` 가 `-m` 로 실행될 때 사용됩니다. (Contributed by Julien Palard in bpo-31639.)

5.22 idlelib 및 IDLE

자동 완성을 위한 여러 수정. (Contributed by Louie Lu in bpo-15786.)

Module Browser (File 메뉴에 있는데, 예전에는 Class Browser라고 불렸습니다)는 이제 최상위 함수와 클래스 외에도 중첩된 함수와 클래스도 표시합니다. (Contributed by Guilherme Polo, Cheryl Sabella, and Terry Jan Reedy in bpo-1612262.)

설정 대화 상자(Options, Configure IDLE)는 모양과 기능을 향상하기 위해 부분적으로 다시 작성되었습니다. (Contributed by Cheryl Sabella and Terry Jan Reedy in multiple issues.)

이제 글꼴 표본에 라틴 문자 이외의 문자가 포함되어 사용자가 특정 글꼴을 선택했을 때의 효과를 더 잘 볼 수 있습니다. (Contributed by Terry Jan Reedy in bpo-13802.) 표본을 편집하여 다른 문자를 포함할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-31860.)

이전에는 확장 기능으로 구현된 IDLE 기능이 일반 기능으로 다시 구현되었습니다. 이것들의 설정을 Extensions 탭에서 다른 대화 상자 탭으로 옮겼습니다. (Contributed by Charles Wohlganger and Terry Jan Reedy in bpo-27099.)

편집기 코드 컨텍스트 옵션이 개정되었습니다. 상자는 최대 행 수까지 모든 컨텍스트 행을 표시합니다. 컨텍스트 행을 클릭하면 편집기가 해당 행으로 이동합니다. 사용자 정의 테마의 컨텍스트 색상이 설정 대화 상자의 강조 표시 탭에 추가되었습니다. (Contributed by Cheryl Sabella and Terry Jan Reedy in bpo-33642, bpo-33768, and bpo-33679.)

윈도우에서, 새로운 API 호출이 tk가 DPI에 맞게 조정된다는 것을 윈도우에 알려줍니다. Windows 8.1+ 또는 10에서, 파이썬 바이너리의 DPI 호환성 속성을 변경하지 않고도, 96DPI보다 큰 모니터 해상도를 사용하면, 텍스트와 선이 더 선명해집니다. 그렇지 않으면 아무런 효과도 일으키지 않습니다. (Contributed by Terry Jan Reedy in bpo-33656.)

3.7.1의 새로운 기능:

N 줄(기본값은 50)을 초과하는 출력은 버튼으로 축소됩니다. N은 설정 대화 상자의 General 페이지의 PyShell 섹션에서 변경할 수 있습니다. 저 작은, 하지만 아주 길 수도 있는, 줄은 출력을 마우스 오른쪽 버튼으로 클릭하면 축소할 수 있습니다. 축소된 출력은 버튼을 더블 클릭해서 재자리에서 확대하거나, 버튼을 마우스 오른쪽 단추로 클릭하여 클립 보드나 별도의 창으로 확대할 수 있습니다. (Contributed by Tal Einat in bpo-1529353.).

위의 변경 사항은 3.6 유지 보수 배포로 역 이식되었습니다.

NEW in 3.7.4:

Add “Run Customized” to the Run menu to run a module with customized settings. Any command line arguments entered are added to sys.argv. They re-appear in the box for the next customized run. One can also suppress the normal Shell main module restart. (Contributed by Cheryl Sabella, Terry Jan Reedy, and others in bpo-5680 and bpo-37627.)

New in 3.7.5:

Add optional line numbers for IDLE editor windows. Windows open without line numbers unless set otherwise in the General tab of the configuration dialog. Line numbers for an existing window are shown and hidden in the Options menu. (Contributed by Tal Einat and Saimadhav Heblikar in bpo-17535.)

5.23 importlib

`importlib.abc.ResourceReader` ABC가 도입되어 패키지에서 리소스를 로딩하는 것을 지원합니다. *importlib.resources* 를 참조하세요. (Contributed by Barry Warsaw, Brett Cannon in bpo-32248.)

`importlib.reload()` 는 이제 모듈에 스펙이 없는 경우 `ModuleNotFoundError` 를 발생시킵니다. (Contributed by Garvit Khatri in bpo-29851.)

`importlib.find_spec()` 은 이제 지정된 부모 모듈이 패키지가 아닌 경우 (즉 `__path__` 어트리뷰트가 없는 경우) `AttributeError` 대신에 `ModuleNotFoundError` 를 발생시킵니다. (Contributed by Milan Oberkirch in bpo-30436.)

새로운 `importlib.source_hash()` 는 전달 된 소스의 해시를 계산하는 데 사용될 수 있습니다. 해시 기반 *.pyc* 파일 은 함수가 반환하는 값을 포함합니다.

5.24 io

새로운 `TextIOWrapper.reconfigure()` 메서드를 사용하여 텍스트 스트림을 새로운 설정으로 재구성 할 수 있습니다. (Contributed by Antoine Pitrou in bpo-30526 and INADA Naoki in bpo-15216.)

5.25 ipaddress

`ipaddress.IPv6Network` 와 `ipaddress.IPv4Network` 의 새로운 `subnet_of()` 와 `supernet_of()` 메서드는 네트워크 포함 테스트에 사용될 수 있습니다. (Contributed by Michel Albert and Cheryl Sabella in bpo-20825.)

5.26 itertools

`itertools.islice()` 는 start, stop, step 인자로 정수류 객체 를 받아들입니다. (Contributed by Will Roberts in bpo-30537.)

5.27 locale

`locale.format_string()` 의 새로운 *monetary* 인자는 변환에 화폐 천 단위 분리자 및 그룹화 문자열을 사용하도록 만듭니다. (Contributed by Garvit in bpo-10379.)

`locale.getpreferredencoding()` 함수는 이제 안드로이드나 강제 *UTF-8* 모드 일 때 항상 `'UTF-8'` 을 반환합니다.

5.28 logging

`Logger` 인스턴스는 이제 피클(pickle) 할 수 있습니다. (Contributed by Vinay Sajip in bpo-30520.)

새로운 `StreamHandler.setStream()` 메서드는 처리기 생성 후 로거 스트림을 대체하는 데 사용될 수 있습니다. (Contributed by Vinay Sajip in bpo-30522.)

이제 `logging.config.fileConfig()` 에 전달된 구성에서 처리기 생성자에 대한 키워드 인자를 지정할 수 있습니다. (Contributed by Preston Landers in bpo-31080.)

5.29 math

새로운 `math.remainder()` 함수는 IEEE 754 스타일의 나머지 연산을 구현합니다. (Contributed by Mark Dickinson in bpo-29962.)

5.30 mimetypes

.bmp의 MIME 형이 `'image/x-ms-bmp'` 에서 `'image/bmp'` 로 변경되었습니다. (Contributed by Nitish Chandra in bpo-22589.)

5.31 msilib

새로운 `Database.Close()` 메서드를 사용하여 MSI (후입 선출법(last-in, first-out)) 데이터베이스를 닫을 수 있습니다. (Contributed by Berker Peksag in bpo-20486.)

5.32 multiprocessing

새로운 `Process.close()` 메서드는 명시적으로 프로세스 객체를 닫고 그와 연관된 모든 자원을 해제합니다. 하부 프로세스가 여전히 실행 중이면 `ValueError` 를 일으킵니다. (Contributed by Antoine Pitrou in bpo-30596.)

새로운 `Process.kill()` 메서드는 유닉스에서 `SIGKILL` 시그널을 사용하여 프로세스를 종료시키는데 사용될 수 있습니다. (Contributed by Vitor Pereira in bpo-30794.)

`Process` 에 의해 생성된 데몬이 아닌 스레드는 이제 프로세스 종료 시에 조인(join)됩니다. (Contributed by Antoine Pitrou in bpo-18966.)

5.33 os

`os.fwalk()` 는 이제 경로 인자로 `bytes` 도 받아들입니다. (Contributed by Serhiy Storchaka in bpo-28682.)

`os.scandir()` 은 파일 기술자 에 대한 지원을 얻었습니다. (Contributed by Serhiy Storchaka in bpo-25996.)

새로운 `register_at_fork()` 함수는 프로세스 포크 때 실행될 파이썬 콜백을 등록 할 수 있게 합니다. (Contributed by Antoine Pitrou in bpo-16500.)

`os.preadv()` (`os.readv()` 와 `os.pread()` 의 기능을 결합한 것)와 `os.pwritev()` 함수 (`os.writev()` 와 `os.pwrite()` 의 기능을 결합한 것)를 추가했습니다. (Contributed by Pablo Galindo in bpo-31368.)

`os.makedirs()` 의 mode 인자는 더는 새로 생성된 중간 수준 디렉터리의 파일 사용 권한 비트에 영향을 미치지 않습니다. (Contributed by Serhiy Storchaka in bpo-19930.)

`os.dup2()` 는 이제 새로운 파일 기술자를 반환합니다. 이전에는 항상 `None` 을 반환했습니다. (Contributed by Benjamin Peterson in bpo-32441.)

os.stat() 에 의해 반환된 구조체는 이제 솔라리스와 그 파생물에서 st_fstype 어트리뷰트를 포함합니다. (Contributed by Jesús Cea Avión in bpo-32659.)

5.34 pathlib

새로운 Path.is_mount() 메서드는 이제 POSIX 시스템에서 사용할 수 있으며 경로가 마운트 지점인지 아닌지를 결정하는 데 사용할 수 있습니다. (Contributed by Cooper Ry Lees in bpo-30897.)

5.35 pdb

pdb.set_trace() 는 이제 선택적 *header* 키워드 전용 인자를 취합니다. 주어진 경우, 디버깅이 시작되기 바로 전에 콘솔에 출력됩니다. (Contributed by Barry Warsaw in bpo-31389.)

pdb 명령행은 이제 스크립트 파일의 대안으로 -m module_name 을 받아들입니다. (Contributed by Mario Corchero in bpo-32206.)

5.36 py_compile

py_compile.compile() 은 – 그리고 자연히 compileall 도 – 이제 SOURCE_DATE_EPOCH 환경 변수가 설정되면 해시 기반 유효성 검사를 위해 .pyc 파일을 무조건 생성합니다. 이것은 .pyc 파일의 재현 가능한 빌드 를 보장할 수 있도록 한다. (Contributed by Bernhard M. Wiedemann in bpo-29708.)

5.37 pydoc

pydoc 서버는 이제 새로운 -n 명령행 인자로 지정된 임의의 호스트 이름에 바인드 할 수 있습니다. (Contributed by Feanil Patel in bpo-31128.)

5.38 queue

새로운 SimpleQueue 클래스는 무제한 FIFO (후입 선출법 (last-in, first-out)) 큐입니다. (Contributed by Antoine Pitrou in bpo-14976.)

5.39 re

플래그 re.ASCII, re.LOCALE 및 re.UNICODE 를 그룹의 스코프 내에서 설정할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-31690.)

re.split() 는 이제 빈 문자열과 일치하는 r'\b', '^$' 또는 (?=-) 와 같은 패턴으로 나누는 것을 지원합니다. (Contributed by Serhiy Storchaka in bpo-25054.)

re.LOCALE 플래그로 컴파일된 정규식은 더는 컴파일 시간의 로케일에 의존하지 않습니다. 로케일 설정은 컴파일된 정규식이 사용될 때 적용됩니다. (Contributed by Serhiy Storchaka in bpo-30215.)

정규식이 앞으로 의미적인 변경이 있을 문자 집합 구조물(가령 중첩된 집합과 집합 연산)을 포함하면 이제 FutureWarning 을 줍니다. (Contributed by Serhiy Storchaka in bpo-30349.)

copy.copy() 와 copy.deepcopy() 를 사용하여 컴파일된 정규식과 매치 객체를 복사 할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-10076.)

5.40 signal

`signal.set_wakeup_fd()` 함수의 새로운 *warn_on_full_buffer* 인자는 웨이크업 버퍼 오버플로가 발생했을 때 파이썬이 표준 에러로 경고를 출력할지를 지정할 수 있게 합니다. (Contributed by Nathaniel J. Smith in bpo-30050.)

5.41 socket

새로운 `socket.getblocking()` 메서드는 소켓이 블로킹 모드에 있으면 `True` 를 반환하고, 그렇지 않으면 `False` 를 반환합니다. (Contributed by Yury Selivanov in bpo-32373.)

새로운 `socket.close()` 함수는 전달된 소켓 파일 기술자를 닫습니다. 이 함수는 플랫폼 간 호환성을 높이기 위해 `os.close()` 대신 사용되어야 합니다. (Contributed by Christian Heimes in bpo-32454.)

`socket` 모듈은 이제 `socket.TCP_CONGESTION` (리눅스 2.6.13), `socket.TCP_USER_TIMEOUT` (Linux 2.6.37) 그리고 `socket.TCP_NOTSENT_LOWAT` (리눅스 3.12) 상수를 노출합니다. (Contributed by Omar Sandoval in bpo-26273 and Nathaniel J. Smith in bpo-29728.)

가상 기계와 호스트 간의 통신을 허용하기 위해 `socket.AF_VSOCK` 소켓 지원이 추가되었습니다. (Contributed by Cathy Avery in bpo-27584.)

소켓은 이제 기본적으로 파일 기술자에서 패밀리, 형 및 프로토콜을 자동 감지합니다. (Contributed by Christian Heimes in bpo-28134.)

5.42 socketserver

`socketserver.ThreadingMixIn.server_close()` 는 이제 모든 데몬이 아닌 스레드가 완료될 때까지 대기합니다. `socketserver.ForkingMixIn.server_close()` 는 모든 자식 프로세스가 완료될 때까지 대기합니다.

새로운 `socketserver.ForkingMixIn.block_on_close` 클래스 어트리뷰트를 `socketserver.ForkingMixIn` 과 `socketserver.ThreadingMixIn` 클래스에 추가했습니다. 3.7 이전의 동작을 얻으려면 클래스 어트리뷰트를 `False` 로 설정하십시오.

5.43 sqlite3

`sqlite3.Connection` 은 이제 하부 SQLite 라이브러리의 버전이 3.6.11 이상일 때 `backup()` 메서드를 노출합니다. (Contributed by Lele Gaifax in bpo-27645.)

`sqlite3.connect()` 의 *database* 인자는 이제 문자열뿐만 아니라 모든 경로류 객체 를 받아들입니다. (Contributed by Anders Lorentsen in bpo-31843.)

5.44 ssl

`ssl` 모듈은 이제 `match_hostname()` 대신 OpenSSL의 내장 API를 사용하여 호스트 이름이나 IP 주소를 확인합니다. 값은 TLS 핸드 셰이크 중에 유효성이 검사됩니다. 호스트 이름 검사 실패와 같은 인증서 유효성 검사 오류가 발생하면 이제 `SSLCertVerificationError` 가 발생하고 적절한 TLS 경고 메시지와 함께 핸드 셰이크가 중단됩니다. 새 예외에는 추가 정보가 들어 있습니다. 호스트 이름 유효성 검증은 `SSLContext.hostname_checks_common_name` 을 사용하여 사용자 정의 할 수 있습니다. (Contributed by Christian Heimes in bpo-31399.)

참고: 향상된 호스트 이름 검사에는 OpenSSL 1.0.2 또는 1.1과 호환되는 *libssl* 구현이 필요합니다. 따라서, OpenSSL 0.9.8 및 1.0.1은 더는 지원되지 않습니다 (더 자세한 내용은 플랫폼 지원 제거를 보세요). ssl 모듈은 대부분 LibreSSL 2.7.2 이상과 호환됩니다.

ssl 모듈은 더는 SNI TLS 확장에서 IP 주소를 전송하지 않습니다. (Contributed by Christian Heimes in bpo-32185.)

match_hostname() 은 www*.example.org 와 같은 부분적인 와일드카드를 더는 지원하지 않습니다. (Contributed by Mandeep Singh in bpo-23033 and Christian Heimes in bpo-31399.)

ssl 모듈의 기본 사이퍼 군 선택은 이제 하드 코딩된 화이트리스트가 아닌 블랙리스트 접근법을 사용합니다. 파이썬은 더는 OpenSSL 보안 업데이트 때문에 차단된 사이퍼를 다시 활성화하지 않습니다. 기본 사이퍼 군 선택은 컴파일 시점에 구성 할 수 있습니다. (Contributed by Christian Heimes in bpo-31429.)

국제화된 도메인 이름 (IDN) 을 포함하는 서버 인증서의 유효성 검사가 이제 지원됩니다. 이 변경의 일부로, SSLSocket.server_hostname 어트리뷰트는 이제 U-label 형식 ("pythön.org") 보다는 A-label 형식 ("xn--pythn-mua.org") 으로 기대하는 호스트 이름을 저장합니다. (Contributed by Nathaniel J. Smith and Christian Heimes in bpo-28414.)

ssl 모듈은 TLS 1.3과 OpenSSL 1.1.1을 예비적이고 실험적으로 지원합니다. 파이썬 3.7.0 배포 당시, OpenSSL 1.1.1은 아직 개발 중이며 TLS 1.3은 아직 완성되지 않았습니다. TLS 1.3 핸드 셰이크와 프로토콜은 TLS 1.2 및 그 이전 버전과 약간 다르게 동작합니다. ssl-tlsv1_3을 참조하세요. (Contributed by Christian Heimes in bpo-32947, bpo-20995, bpo-29136, bpo-30622 and bpo-33618)

SSLSocket 과 SSLObject 는 더는 공개 생성자를 가지고 있지 않습니다. 직접 인스턴스를 만드는 것은 결코 문서로 만들어지고 지원되는 기능이 아닙니다. 인스턴스는 SSLContext 메서드 wrap_socket() 과 wrap_bio() 로 생성되어야 합니다. (Contributed by Christian Heimes in bpo-32951)

최소 및 최대 TLS 프로토콜 버전을 설정하기 위한 OpenSSL 1.1 API는 SSLContext.minimum_version 과 SSLContext.maximum_version 로 제공됩니다. 지원되는 프로토콜은 여러 가지 새 플래그로 표시됩니다, 가령 HAS_TLSv1_1. (Contributed by Christian Heimes in bpo-32609.)

TLS 1.3 포스트 핸드 셰이크 인증을 활성화하는 SSLContext.post_handshake_auth과 시작시키는 ssl.SSLSocket.verify_client_post_handshake() 를 추가했습니다. (Contributed by Christian Heimes in bpo-34670.)

5.45 string

string.Template 은 이제 중괄호로 둘러싼 치환식과 그렇지 않은 치환식의 정규식을 따로 선택적으로 수정할 수 있도록 합니다. (Contributed by Barry Warsaw in bpo-1198569.)

5.46 subprocess

subprocess.run() 함수는 새로운 *capture_output* 키워드 인자를 받습니다. 참일 때, stdout과 stderr가 캡처됩니다. 이것은 subprocess.PIPE 를 *stdout* 및 *stderr* 인자로 전달하는 것과 같습니다. (Contributed by Bo Bayles in bpo-32102.)

subprocess.run 함수와 subprocess.Popen 생성자는 이제 *text* 키워드 인자를 *universal_newlines* 의 별칭으로 받아들입니다. (Contributed by Andrew Clegg in bpo-31756.)

윈도우에서 *close_fds* 의 기본값은 표준 핸들을 리디렉션 할 때 False 에서 True 로 변경되었습니다. 이제 표준 핸들을 리디렉션 할 때 *close_fds* 를 참으로 설정할 수 있습니다. subprocess.Popen 을 참조하세요. 이것은 지원되는 모든 플랫폼에서 이제 *close_fds* 의 기본값이 True 임을 뜻합니다. (Contributed by Segev Finer in bpo-19764.)

subprocess 모듈은 이제 `subprocess.call()`, `subprocess.run()` 중에, 또는 `Popen` 컨텍스트 관리자에 있는 동안 `KeyboardInterrupt` 를 더 우아하게 처리합니다. 이제 `KeyboardInterrupt` 예외 처리를 계속하기 전에 자식이 종료될 때까지 약간의 시간을 기다립니다. (Contributed by Gregory P. Smith in bpo-25942.)

5.47 sys

새로운 `sys.breakpointhook()` 훅 함수는 내장 `breakpoint()` 에 의해 호출됩니다. (Contributed by Barry Warsaw in bpo-31353.)

안드로이드에서, 새로운 `sys.getandroidapilevel()` 은 빌드 시간 안드로이드 API 버전을 반환합니다. (Contributed by Victor Stinner in bpo-28740.)

새로운 `sys.get_coroutine_origin_tracking_depth()` 함수는 새로운 `sys.set_coroutine_origin_tracking_depth()` 에 의해 설정된 현재 코루틴 원점 추적 깊이를 반환합니다. `asyncio` 가 폐지된 `sys.set_coroutine_wrapper()` 대신 이 새로운 API를 사용하도록 변환되었습니다. (Contributed by Nathaniel J. Smith in bpo-32591.)

5.48 time

PEP 564 는 `time` 모듈에 나노초 해상도의 새로운 함수 6개를 추가합니다:

- `time.clock_gettime_ns()`
- `time.clock_settime_ns()`
- `time.monotonic_ns()`
- `time.perf_counter_ns()`
- `time.process_time_ns()`
- `time.time_ns()`

새로운 시계 식별자가 추가되었습니다:

- `time.CLOCK_BOOTTIME` (리눅스): 시스템이 일시 중지된 시간을 포함한다는 점만 제외하고는 `time.CLOCK_MONOTONIC` 과 같습니다.
- `time.CLOCK_PROF` (FreeBSD, NetBSD, OpenBSD): 고해상도의 프로세스별 CPU 타이머.
- `time.CLOCK_UPTIME` (FreeBSD, OpenBSD): 절댓값이 시스템이 실행 중이었고 중단되지 않은 시간인 시간. 정확한 업타임 측정을 제공합니다.

새로운 `time.thread_time()` 과 `time.thread_time_ns()` 함수는 스레드 당 CPU 시간을 측정하는 데 사용될 수 있습니다. (Contributed by Antoine Pitrou in bpo-32025.)

새로운 `time.pthread_getcpuclockid()` 함수는 스레드 별 CPU 시간 시계의 시계 ID를 반환합니다.

5.49 tkinter

이제 새로운 `tkinter.ttk.Spinbox` 클래스를 사용할 수 있습니다. (Contributed by Alan Moore in bpo-32585.)

5.50 tracemalloc

`tracemalloc.Traceback` 은 더 일반 트레이스백과 같이 동작하여, 프레임을 가장 오래된 것부터 가장 최근의 것으로 정렬합니다. `Traceback.format()` 은 이제 음의 *limit* 을 받아들이고, 결과를 `abs(limit)` 개의 가장 오래된 프레임으로 잘라냅니다. 이전 동작을 얻으려면, `Traceback.format()` 에 새로운 *most_recent_first* 인자를 사용하십시오. (Contributed by Jesse Bakker in bpo-32121.)

5.51 types

이제 새로운 `WrapperDescriptorType`, `MethodWrapperType`, `MethodDescriptorType`, `ClassMethodDescriptorType` 클래스를 사용할 수 있습니다. (Contributed by Manuel Krebber and Guido van Rossum in bpo-29377, and Serhiy Storchaka in bpo-32265.)

새로운 `types.resolve_bases()` 함수는 **PEP 560** 에 지정된 대로 MRO 항목을 동적으로 결정합니다. (Contributed by Ivan Levkivskyi in bpo-32717.)

5.52 unicodedata

내부 `unicodedata` 데이터베이스가 유니코드 11 을 사용하도록 업그레이드되었습니다. (Contributed by Benjamin Peterson.)

5.53 unittest

새로운 `-k` 명령행 옵션은 이름 부분 문자열이나 유닉스 셸과 같은 패턴으로 테스트를 필터링 할 수 있습니다. 예를 들어 `python -m unittest -k foo` 는 `foo_tests.SomeTest.test_something`, `bar_tests.SomeTest.test_foo` 를 실행하지만, `bar_tests.FooTest.test_something` 는 실행하지 않습니다. (Contributed by Jonas Haag in bpo-32071.)

5.54 unittest.mock

`sentinel` 어트리뷰트는 이제 복사되거나 피클될 때 그들의 아이덴티티를 보존합니다. (Contributed by Serhiy Storchaka in bpo-20804.)

새로운 `seal()` 함수는 `Mock` 인스턴스를 봉인하도록 허용합니다. 추가적인 어트리뷰트 모의 객체를 만들 수 없도록 합니다. 봉인은 모의 객체인 모든 어트리뷰트에 재귀적으로 적용됩니다. (Contributed by Mario Corchero in bpo-30541.)

5.55 urllib.parse

`urllib.parse.quote()` 가 **RFC 2396**에서 **RFC 3986**으로 갱신되어, 기본적으로 이스케이프 되지 않는 문자 집합에 ~ 가 추가되었습니다. (Contributed by Christian Theune and Ratnadeep Debnath in bpo-16285.)

5.56 uu

`uu.encode()` 함수는 이제 선택적 *backtick* 키워드 인자를 받아들입니다. 참일 때, 0은 스페이스 대신에 `'`'` 로 표현됩니다. (Contributed by Xiang Zhang in bpo-30103.)

5.57 uuid

새로운 `UUID.is_safe` 어트리뷰트는 생성된 UUID가 다중 프로세스에 안전한 방법으로 생성되었는지에 대해 플랫폼이 주는 정보를 전달합니다. (Contributed by Barry Warsaw in bpo-22807.)

`uuid.getnode()` 는 이제 지역적으로 관리되는 MAC 주소보다 보편적으로 관리되는 MAC 주소를 선호합니다. 이것은 `uuid.uuid1()` 에서 반환된 UUID의 글로벌 유일성을 더 잘 보장합니다. 지역적으로 관리되는 MAC 주소 만 사용할 수 있는 경우, 처음 발견된 MAC 주소가 반환됩니다. (Contributed by Barry Warsaw in bpo-32107.)

5.58 warnings

기본 경고 필터의 초기화가 다음과 같이 변경되었습니다:

- 명령행 옵션(`-b` 및 새로운 CPython 특정 `-X dev` 옵션을 포함합니다)을 통해 활성화된 경고는 항상 `sys.warnoptions` 어트리뷰트를 통해 경고 절차로 전달됩니다.
- 명령행 또는 환경을 통해 활성화된 경고 필터의 우선순위는 다음과 같습니다:
  - `-b` (또는 `-bb`)에 의한 `BytesWarning` 필터
  - `-W` 옵션으로 지정된 모든 필터
  - `PYTHONWARNINGS` 환경 변수로 지정된 모든 필터
  - 다른 모든 CPython 특정 필터 (예, 새로운 `-X dev` 모드를 위해 추가된 `default` 필터)
  - 경고 절차에 의해 직접 정의된 모든 묵시적 필터
- CPython 디버그 빌드에서, 이제 모든 경고가 기본적으로 표시됩니다 (묵시적 필터 목록이 비어 있습니다)

(Contributed by Nick Coghlan and Victor Stinner in bpo-20361, bpo-32043, and bpo-32230.)

폐지 경고는 단일 파일 스크립트 및 대화식 프롬프트에서 다시 한번 기본적으로 표시됩니다. 자세한 내용은 *PEP 565: __main__ 의 DeprecationWarning 표시* 를 참조하십시오. (Contributed by Nick Coghlan in bpo-31975.)

5.59 xml

DTD 및 외부 엔티티 조회에 대한 완화로서, `xml.dom.minidom` 및 `xml.sax` 모듈은 기본적으로 더는 외부 엔티티를 처리하지 않습니다. (Contributed by Christian Heimes in bpo-17239.)

5.60 xml.etree

`find()` 의 ElementPath 서술자는 이제 `[. = "text"]` 로 자식의 텍스트뿐만 아니라 현재 노드의 텍스트를 비교할 수 있습니다. 서술자는 가독성을 높이기 위해 스페이스를 추가할 수도 있습니다. (Contributed by Stefan Behnel in bpo-31648.)

5.61 xmlrpc.server

`SimpleXMLRPCDispatcher.register_function` 는 이제 데코레이터로 사용할 수 있습니다. (Contributed by Xiang Zhang in bpo-7769.)

5.62 zipapp

함수 `create_archive()` 는 이제 사용자가 저장소에 포함되어야 하는 파일을 선택할 수 있도록 선택적 *filter* 인자를 받아들입니다. (Contributed by Irmen de Jong in bpo-31072.)

함수 `create_archive()` 는 이제 압축된 저장소를 생성하기 위해 선택적 *compressed* 인자를 받아들입니다. 명령행 옵션 `--compress` 도 압축을 지원하기 위해 추가되었습니다. (Contributed by Zhiming Wang in bpo-31638.)

5.63 zipfile

`ZipFile` 은 이제 압축 수준을 제어하기 위해 새로운 *compresslevel* 매개변수를 받아들입니다. (Contributed by Bo Bayles in bpo-21417.)

`ZipFile` 에 의해 생성된 저장소에 있는 서브 디렉터리는 이제 알파벳순으로 저장됩니다. (Contributed by Bernhard M. Wiedemann in bpo-30693.)

6 C API 변경

스레드-로컬 저장소를 위한 새로운 API가 구현되었습니다. 개요는 *PEP 539: 스레드-로컬 저장소를 위한 새로운 C API* 를, 완전한 레퍼런스는 thread-specific-storage-api를 보십시오. (Contributed by Masayuki Yamamoto in bpo-25658.)

새로운 컨텍스트 변수 기능은 다수의 새로운 C API를 노출합니다.

새로운 `PyImport_GetModule()` 함수는 주어진 이름으로 이전에 임포트 한 모듈을 반환합니다. (Contributed by Eric Snow in bpo-28411.)

새로운 `Py_RETURN_RICHCOMPARE` 매크로는 풍부한 비교 함수를 작성하기 쉽게 합니다. (Contributed by Petr Victorin in bpo-23699.)

새로운 `Py_UNREACHABLE` 매크로는 도달할 수 없는 코드 경로를 표시하는 데 사용될 수 있습니다. (Contributed by Barry Warsaw in bpo-31338.)

`tracemalloc` 은 이제 새로운 `PyTraceMalloc_Track()` 과 `PyTraceMalloc_Untrack()` 함수를 통해 C API를 노출합니다. (Contributed by Victor Stinner in bpo-30054.)

새로운 `import__find__load__start()` 와 `import__find__load__done()` 정적 마커를 사용하여 모듈 임포트를 추적 할 수 있습니다. (Contributed by Christian Heimes in bpo-31574.)

구조체 `PyMemberDef`, `PyGetSetDef`, `PyStructSequence_Field`, `PyStructSequence_Desc` 그리고 `wrapperbase` 의 필드 `name`과 `doc` 은 이제 `char *` 이 아니라 `const char *` 형입니다. (Contributed by Serhiy Storchaka in bpo-28761.)

`PyUnicode_AsUTF8AndSize()` 와 `PyUnicode_AsUTF8()` 의 결과는 이제 `char *` 이 아니라 `const char *` 입니다. (Contributed by Serhiy Storchaka in bpo-28769.)

`PyMapping_Keys()`, `PyMapping_Values()` 및 `PyMapping_Items()` 의 결과는 이제 리스트 또는 튜플이 아니라 항상 리스트입니다. (Contributed by Oren Milman in bpo-28280.)

함수 `PySlice_Unpack()`과 `PySlice_AdjustIndices()` 를 추가했습니다. (Contributed by Serhiy Storchaka in bpo-27867.)

`PyOS_AfterFork()` 는 폐지되고, 새 함수 `PyOS_BeforeFork()`, `PyOS_AfterFork_Parent()` 그리고 `PyOS_AfterFork_Child()`로 대체되었습니다. (Contributed by Antoine Pitrou in bpo-16500.)

공개 API의 일부였던 `PyExc_RecursionErrorInst` 싱글톤은 제거되었습니다. 청소되지 않은 멤버가 인터프리터의 파이널리제이션 과정에서 세그멘테이션 오류를 일으킬 수 있기 때문입니다. (Contributed by Xavier de Gaye in bpo-22898 and bpo-30697.)

시간대 지원을 위한 timezone 생성자 `PyTimeZone_FromOffset()` 과 `PyTimeZone_FromOffsetAndName()` 및 UTC 싱글톤에 액세스하는 `PyDateTime_TimeZone_UTC` C API 지원을 추가했습니다. (Contributed by Paul Ganssle in bpo-10381.)

`PyThread_start_new_thread()` 및 `PyThread_get_thread_ident()` 의 결과와 `PyThreadState_SetAsyncExc()` 의 *id* 매개 변수의 형이 `long`에서 `unsigned long`로 변경되었습니다. (Contributed by Serhiy Storchaka in bpo-6532.)

`PyUnicode_AsWideCharString()` now raises a `ValueError` if the second argument is `NULL` and the `wchar_t*` string contains null characters. (Contributed by Serhiy Storchaka in bpo-30708.)

시동 절차와 동적 메모리 할당자의 변경으로 인해 대부분의 C API 함수를 호출하기 전에 `Py_Initialize()` 를 호출해야 한다는 오래전부터 문서화 된 요구 사항이 이제 더 중요해졌고, 이를 지키지 않으면 내장형 응용 프로그램에서 세그멘테이션 오류로 이어질 수 있습니다. 자세한 내용은 이 문서의 *파이썬 3.7로 이식하기* 절과 C API 설명서의 pre-init-safe 절을 참조하십시오.

새로운 `PyInterpreterState_GetID()` 는 주어진 인터프리터의 유일한 ID를 반환합니다. (Contributed by Eric Snow in bpo-29102.)

`Py_DecodeLocale()`, `Py_EncodeLocale()` 은 이제 *UTF-8 모드* 가 활성화될 때 UTF-8 인코딩을 사용합니다. (Contributed by Victor Stinner in bpo-29240.)

`PyUnicode_DecodeLocaleAndSize()` 와 `PyUnicode_EncodeLocale()`은 이제 `surrogateescape` 에러 처리기에 현재 로케일 인코딩을 사용합니다. (Contributed by Victor Stinner in bpo-29240.)

`PyUnicode_FindChar()` 의 *start* 와 *end* 매개 변수가 문자열 슬라이스처럼 동작하도록 조정되었습니다. (Contributed by Xiang Zhang in bpo-28822.)

7 빌드 변경

`--without-threads` 빌드 지원이 제거되었습니다. `threading` 모듈은 이제 항상 사용할 수 있습니다. (Contributed by Antoine Pitrou in bpo-31370.)

OSX 이외의 유닉스 플랫폼에서 `_ctypes` 모듈을 빌드할 때 사용되는 libffi의 전체 복사본이 더는 포함되지 않습니다. 이러한 플랫폼에서 `_ctypes` 를 빌드 할 때 libffi의 설치된 사본이 필요합니다. (Contributed by Zachary Ware in bpo-27979.)

윈도우 빌드 프로세스는 외부 소스를 가져오는 데 더는 서브버전에 의존하지 않습니다. 대신 파이썬 스크립트를 사용해서 GitHub에서 zip 파일을 내려받습니다. 시스템에서 파이썬 3.6이 발견되지 않으면 (`py -3.6` 를 통해), 이 목적을 위해 NuGet으로 32-비트 파이썬의 사본을 내려받습니다. (Contributed by Zachary Ware in bpo-30450.)

`ssl` 모듈은 OpenSSL 1.0.2 또는 1.1 호환 libssl을 요구합니다. OpenSSL 1.0.1의 유효 기간은 2016-12-31 에 만료되었고 더는 지원되지 않습니다. LibreSSL도 일시적으로 지원되지 않습니다. 버전 2.6.4까지의 LibreSSL 배포에는 필수 OpenSSL 1.0.2 API가 없습니다.

8 최적화

C로 구현된 다양한 표준 라이브러리 클래스의 많은 메서드를 호출하는 오버헤드가, `METH_FASTCALL` 규칙을 사용하도록 더 많은 코드를 이식함으로써 상당히 감소하였습니다. (Contributed by Victor Stinner in bpo-29300, bpo-29507, bpo-29452, and bpo-29286.)

다양한 최적화를 통해 파이썬의 시작 시간을 리눅스에서 10%, macOS에서 최대 30%까지 줄였습니다. (Contributed by Victor Stinner, INADA Naoki in bpo-29585, and Ivan Levkivskyi in bpo-31333.)

메서드 호출은 이제 연결된 메서드 인스턴스 생성을 피하는 바이트 코드 변경으로 인해 최대 20% 빨라졌습니다. (Contributed by Yury Selivanov and INADA Naoki in bpo-26110.)

`asyncio` 모듈은 자주 사용되는 함수들에 대해 여러 가지 주목할만한 최적화가 이루어졌습니다:

- `asyncio.get_event_loop()` 함수가 C로 다시 구현되어 최대 15배 빨라졌습니다. (Contributed by Yury Selivanov in bpo-32296.)
- `asyncio.Future` 콜백 관리가 최적화되었습니다. (Contributed by Yury Selivanov in bpo-32348.)
- `asyncio.gather()` 는 이제 최대 15% 빨라졌습니다. (Contributed by Yury Selivanov in bpo-32355.)
- `asyncio.sleep()`은 이제 *delay* 인자가 0이거나 음수일 때 최대 2배 빠릅니다. (Contributed by Andrew Svetlov in bpo-32351.)
- asyncio 디버그 모드의 성능 오버헤드가 감소하였습니다. (Contributed by Antoine Pitrou in bpo-31970.)

PEP 560 작업 의 결과로, `typing` 의 임포트 시간이 7배 단축되었으며, 많은 typing 연산이 이제 더 빨라졌습니다. (Contributed by Ivan Levkivskyi in bpo-32226.)

`sorted()`와 `list.sort()` 는 일반적인 사용에 대해 최대 40-75% 더 빠르게 최적화되었습니다. (Contributed by Elliot Gorokhovsky in bpo-28685.)

`dict.copy()` 는 이제 5.5 배 빠릅니다. (Contributed by Yury Selivanov in bpo-31179.)

`hasattr()`과 `getattr()`은 이제 *name* 이 발견되지 않고 *obj* 가 `object.__getattr__()` 또는 `object.__getattribute__()` 를 재정의하지 않을 때 약 4배 빨라졌습니다. (Contributed by INADA Naoki in bpo-32544.)

문자열에서 특정 유니코드 문자 (가령 우크라이나어 "Є")를 검색하는 것은 다른 문자를 검색하는 것보다 최대 25배까지 느렸습니다. 이제는 최악의 상황에도 겨우 3배 느립니다. (Contributed by Serhiy Storchaka in bpo-24821.)

`collections.namedtuple()` 팩토리가 재구현되어 네임드 튜플을 4 에서 6배 빠르게 생성합니다. (Contributed by Jelle Zijlstra with further improvements by INADA Naoki, Serhiy Storchaka, and Raymond Hettinger in bpo-28638.)

`date.fromordinal()` 과 `date.fromtimestamp()` 는 이제 일반적일 때 최대 30% 더 빠릅니다. (Contributed by Paul Ganssle in bpo-32403.)

`os.fwalk()` 함수는 `os.scandir()` 을 사용함으로써 이제 최대 2배 빨라졌습니다. (Contributed by Serhiy Storchaka in bpo-25996.)

`os.scandir()` 함수를 사용하여 `shutil.rmtree()` 함수의 속도가 20–40% 향상되었습니다. (Contributed by Serhiy Storchaka in bpo-28564.)

최적화된 정규식 의 대소 문자를 구별하지 않는 매칭과 검색. 일부 패턴 검색은 이제 최대 20배까지 빨라질 수 있습니다. (Contributed by Serhiy Storchaka in bpo-30285.)

`re.compile()` 은 이제 `flags` 매개 변수가 `RegexFlag` 인 경우 int 객체로 변환합니다. 이제 파이썬 3.5만큼 빠르고, 패턴에 따라 파이썬 3.6보다 약 10% 빠릅니다. (Contributed by INADA Naoki in bpo-31671.)

클래스 selectors.EpollSelector, selectors.PollSelector 와 selectors.DevpollSelector 의 modify() 메서드는 높은 부하가 걸릴 때 10% 정도 더 빨라질 수 있습니다. (Contributed by Giampaolo Rodola' in bpo-30014)

상수 폴딩은 핍홀 최적화기에서 새로운 AST 최적화기로 이동되어, 더욱 일관된 최적화를 수행 할 수 있습니다. (Contributed by Eugene Toder and INADA Naoki in bpo-29469 and bpo-11549.)

abc 에 있는 대부분 함수와 메서드는 C로 재작성되었습니다. 이로 인해 추상 베이스 클래스 생성과, 이에 대한 isinstance() 와 issubclass() 호출이 1.5 배 더 빠릅니다. 또한, 파이썬 시작 시간을 최대 10% 단축합니다. (Contributed by Ivan Levkivskyi and INADA Naoki in bpo-31333)

datetime.date 와 datetime.datetime 의 대체 생성자의 경우, 서브 클래스를 만들지 않을 때 파이썬 생성자를 우회해서 속도가 현저히 향상되었습니다. (Contributed by Paul Ganssle in bpo-32403)

array.array 인스턴스의 비교 속도가 어떤 경우에는 상당히 향상되었습니다. 같은 정수 형의 값을 보유한 배열을 비교할 때, 이제는 10배에서 70배 빠릅니다. (Contributed by Adrian Wielgosik in bpo-24700.)

math.erf() 와 math.erfc() 함수는 이제 대부분 플랫폼에서 (더 빠른) C 라이브러리 구현을 사용합니다. (Contributed by Serhiy Storchaka in bpo-26121.)

9 기타 CPython 구현 변경

- 트레이스 훅은 이제 추적할 프레임의 새로운 f_trace_lines 와 f_trace_opcodes 어트리뷰트를 설정하여, 인터프리터로부터 line 대신 opcode 이벤트를 수신하도록 선택할 수 있습니다. (Contributed by Nick Coghlan in bpo-31344.)
- 이름 공간 패키지 모듈 어트리뷰트에 대한 일관성 문제를 수정했습니다. 이름 공간 모듈 객체는 이제 __file__ 을 None (이전에는 설정되지 않았습니다)으로 설정하고, __spec__.origin 도 None (이전에는 문자열 "namespace")으로 설정됩니다. bpo-32305를 참조하세요. 또한, 이름 공간 모듈 객체의 __spec__.loader 는 __loader__ 와 같은 값으로 설정됩니다 (이전에는 None 으로 설정되었습니다). bpo-32303을 참조하세요.
- locals() 딕셔너리는 이제 변수가 정의된 어휘 순서로 표시됩니다. 이전에는 순서가 정의되지 않았습니다. (Contributed by Raymond Hettinger in bpo-32690.)
- distutils upload 명령은 더는 CR 줄 마침 문자를 CRLF로 바꾸려고 시도하지 않습니다. 이것은 CR 에 해당하는 바이트로 끝나는 sdists의 손상 문제를 수정합니다. (Contributed by Bo Bayles in bpo-32304.)

10 폐지된 파이썬 동작

일드 표현식(yield 와 yield from 절 모두)은 이제 컴프리헨션과 제너레이터 표현식에서 폐지되었습니다 (가장 왼쪽의 for 절의 이터러블 표현식은 제외합니다). 이것은 컴프리헨션이 언제나 적절한 유형의 컨테이너를 즉시 반환하도록 하고(잠재적으로 제너레이터 이터레이터 객체를 반환하는 것이 아니라), 제너레이터 표현식이 명시적 일드 표현식의 출력 중간에 묵시적 출력을 끼워 넣으려고 시도하지 못하게 하기 위함입니다. 파이썬 3.7에서 이러한 표현식은 컴파일될 때 DeprecationWarning 을 내보내고, 파이썬 3.8에서는 SyntaxError 가 됩니다. (Contributed by Serhiy Storchaka in bpo-10544.)

complex 의 서브 클래스를 object.__complex__() 에서 반환하는 것은 폐지되었고, 향후 파이썬 버전에서 에러가 발생할 것입니다. 이것은 __complex__() 를 object.__int__() 및 object.__float__() 와 일관성 있도록 만듭니다. (Contributed by Serhiy Storchaka in bpo-28894.)

11 폐지된 파이썬 모듈, 함수 및 메서드

11.1 aifc

`aifc.openfp()` 는 폐지되었고 파이썬 3.9에서 제거될 것입니다. 대신에 `aifc.open()` 을 사용하십시오. (Contributed by Brian Curtin in bpo-31985.)

11.2 asyncio

`asyncio.Lock` 및 다른 asyncio 동기화 프리미티브의 인스턴스를 직접 `await` 하는 지원은 폐지되었습니다. 동기화 자원을 획득하고 해제하기 위해서는 비동기 컨텍스트 관리자를 사용해야 합니다. (Contributed by Andrew Svetlov in bpo-32253.)

`asyncio.Task.current_task()` 와 `asyncio.Task.all_tasks()` 메서드는 폐지되었습니다. (Contributed by Andrew Svetlov in bpo-32250.)

11.3 collections

파이썬 3.8에서, `collections.abc` 의 추상 베이스 클래스는 더는 `collections` 모듈에 노출되지 않습니다. 이것은 구상 클래스와 추상 베이스 클래스 사이의 명확한 구분을 만드는 데 도움이 됩니다. (Contributed by Serhiy Storchaka in bpo-25988.)

11.4 dbm

`dbm.dumb` 은 이제 읽기 전용 파일 읽기를 지원하며 변경되지 않았을 때 더는 색인 파일을 쓰지 않습니다. `'r'` 및 `'w'` 모드에서 색인 파일이 빠지고 새로 만들어지면, 이제 폐지 경고가 표시됩니다 (향후의 파이썬 배포에서는 에러가 발생합니다). (Contributed by Serhiy Storchaka in bpo-28847.)

11.5 enum

파이썬 3.8에서, `Enum` 클래스에서 열거형 이외의 객체가 포함되었는지 검사하려고 하면 `TypeError` 가 발생합니다 (예 `1 in Color`); 마찬가지로 플래그가 아닌 객체를 `Flag` 멤버에 포함되었는지 검사하려고 하면 `TypeError` 가 발생합니다 (예 `1 in Perm.RW`); 현재 두 연산 모두 대신 `False`를 반환합니다. (Contributed by Ethan Furman in bpo-33217.)

11.6 gettext

`gettext` 에서 복수형을 선택하기 위해 정수가 아닌 값을 사용하는 것이 폐지되었습니다. 올바르게 작동된 적이 없습니다. (Contributed by Serhiy Storchaka in bpo-28692.)

11.7 importlib

메서드 `MetaPathFinder.find_module()` (`MetaPathFinder.find_spec()` 로 대체되었습니다)와 `PathEntryFinder.find_loader()` (`PathEntryFinder.find_spec()` 로 대체되었습니다) 는 둘 다 파이썬 3.4에서 폐지되었고, 이제는 `DeprecationWarning` 을 일으킵니다. (Contributed by Matthias Bussonnier in bpo-29576)

`importlib.abc.ResourceLoader` ABC는 폐지되었고, 대신 `importlib.abc.ResourceReader` 를 사용합니다.

11.8 locale

`locale.format()` 은 폐지되었습니다. 대신 `locale.format_string()` 을 사용하십시오. (Contributed by Garvit in bpo-10379.)

11.9 macpath

`macpath` 는 이제 폐지되었고 파이썬 3.8에서 제거될 것입니다. (Contributed by Chi Hsuan Yen in bpo-9850.)

11.10 threading

`dummy_threading`과 `_dummy_thread` 는 폐지되었습니다. 스레딩을 비활성화하여 파이썬을 빌드하는 것이 더는 가능하지 않습니다. 대신 `threading`을 사용하십시오. (Contributed by Antoine Pitrou in bpo-31370.)

11.11 socket

`socket.htons()` 와 `socket.ntohs()` 에서 자동 인자 값 자름은 폐지되었습니다. 이후 버전의 파이썬에서는 전달된 인자가 16비트보다 큰 경우 예외가 발생합니다. (Contributed by Oren Milman in bpo-28332.)

11.12 ssl

`ssl.wrap_socket()` 은 폐지되었습니다. 대신 `ssl.SSLContext.wrap_socket()` 을 사용하십시오. (Contributed by Christian Heimes in bpo-28124.)

11.13 sunau

`sunau.openfp()` 는 폐지되었고 파이썬 3.9에서 제거될 것입니다. 대신에 `sunau.open()` 을 사용하십시오. (Contributed by Brian Curtin in bpo-31985.)

11.14 sys

`sys.set_coroutine_wrapper()`와 `sys.get_coroutine_wrapper()`를 폐지했습니다.

문서로 만들어지지 않은 `sys.callstats()` 함수는 폐지되었고 향후 파이썬 버전에서 제거될 것입니다. (Contributed by Victor Stinner in bpo-28799.)

11.15 wave

`wave.openfp()` 는 폐지되었고 파이썬 3.9에서 제거될 것입니다. 대신 `wave.open()` 을 사용하십시오. (Contributed by Brian Curtin in bpo-31985.)

12 폐지된 C API의 함수 및 형

함수 `PySlice_GetIndicesEx()` 는 폐지되었고, `Py_LIMITED_API` 가 설정되어 있지 않거나, `0x03050400` 과 `0x03060000` (포함되지 않음) 사이의 값이나 `0x03060100` 이상의 값으로 설정되어 있으면 매크로로 대체됩니다. (Contributed by Serhiy Storchaka in bpo-27867.)

`PyOS_AfterFork()` 는 폐지되었습니다. `PyOS_BeforeFork()`, `PyOS_AfterFork_Parent()` 또는 `PyOS_AfterFork_Child()` 를 대신 사용하십시오. (Contributed by Antoine Pitrou in bpo-16500.)

13 플랫폼 지원 제거

- FreeBSD 9와 그 이전 버전은 더는 공식적으로 지원되지 않습니다.
- 확장 모듈 내를 포함하여, 완전한 유니코드 지원을 위해 유닉스 플랫폼은 이제 레거시 `ASCII` 기반 `C` 로케일의 대안으로, `C.UTF-8` (전체 로케일), `C.utf8` (전체 로케일) 또는 `UTF-8` (`LC_CTYPE` 전용 로케일) 중 적어도 하나를 제공할 것으로 기대됩니다.
- OpenSSL 0.9.8 및 1.0.1은 더는 지원되지 않습니다. 이는 여전히 이 버전을 사용하는 이전 플랫폼에서 SSL/TLS를 지원하는 CPython 3.7을 빌드하려면 최신 버전의 OpenSSL에 링크되는 사용자 정의 빌드 옵션이 필요함을 뜻합니다.

  특히, 이 문제는 Debian 8 (일명 "jessie") 및 Ubuntu 14.04 (일명 "Trusty") LTS 리눅스 배포판에 영향을 미칩니다. 아직 OpenSSL 1.0.1을 기본적으로 사용하기 때문입니다.

  Debian 9 ("stretch") 및 Ubuntu 16.04 ("xenial") 는, 다른 LTS 리눅스 배포판(가령, RHEL/CentOS 7.5, SLES 12-SP3)의 최신 릴리스 역시, OpenSSL 1.0.2 이상을 사용하고, 기본 빌드 구성에서 여전히 지원됩니다.

  CPython의 자체 CI 구성 파일은 CPython의 테스트 스위트에서 SSL 호환성 테스트 기반시설을 사용하여 구형 시스템이 제공하는 OpenSSL 대신 OpenSSL 1.1.0을 빌드하고 링크하는 예를 제공합니다.

14 API 및 기능 제거

다음 기능과 API는 파이썬 3.7에서 제거되었습니다:

- `os.stat_float_times()` 함수가 제거되었습니다. 이것은 파이썬 2.2와의 하위 호환성을 위해 파이썬 2.3에서 소개되었으며 파이썬 3.1부터 폐지되었습니다.
- `re.sub()` 의 대체 템플릿에 있는 `'\'` 와 ASCII 글자로 이루어진 알 수 없는 이스케이프는 파이썬 3.5에서 폐지되었고, 이제는 에러를 일으킵니다.

- `tarfile.TarFile.add()` 에서 *exclude* 인자의 지원이 제거되었습니다. 파이썬 2.7 및 3.2에서 폐지되었습니다. 대신 *filter* 인자를 사용하십시오.
- `ntpath` 모듈의 `splitunc()` 함수는 파이썬 3.1에서 폐지되었고, 이제 제거되었습니다. 대신에 `splitdrive()` 함수를 사용하십시오.
- `collections.namedtuple()`은 네임드 튜플 클래스에 대해 생성된 소스 코드를 보여주는 *verbose* 매개 변수 나 `_source` 어트리뷰트를 더는 지원하지 않습니다. 이것은 클래스 생성 속도를 높이기 위해 고안된 최적화 일부입니다. (Contributed by Jelle Zijlstra with further improvements by INADA Naoki, Serhiy Storchaka, and Raymond Hettinger in bpo-28638.)
- 함수 `bool()`, `float()`, `list()` 그리고 `tuple()` 은 더는 키워드 인자를 취하지 않습니다. `int()` 의 첫 번째 인자는 이제 위치 인자로만 전달될 수 있습니다.
- `plistlib` 모듈에서, 이전에 파이썬 2.4에서 폐지된 클래스 `Plist`, `Dict`, `_InternalDict` 가 제거되었습니다. `readPlist()` 함수와 `readPlistFromBytes()` 함수의 결과에서 Dict 값은 이제 정상적인 딕셔너리입니다. 더는 어트리뷰트 액세스를 사용하여 이러한 딕셔너리의 항목에 액세스할 수 없습니다.
- `asyncio.windows_utils.socketpair()` 함수가 삭제되었습니다. 대신에 `socket.socketpair()` 함수를 사용하십시오. 파이썬 3.5부터 모든 플랫폼에서 사용할 수 있습니다. `asyncio.windows_utils.socketpair` 는 파이썬 3.5 이상에서 `socket.socketpair` 의 별칭이었습니다.
- `asyncio` 는 더는 `selectors`와 `_overlapped` 모듈을 `asyncio.selectors` 와 `asyncio._overlapped` 로 노출하지 않습니다. `from asyncio import selectors` 를 `import selectors` 로 대체하십시오.
- `ssl.SSLSocket` 과 `ssl.SSLObject` 객체의 직접적인 인스턴스 생성이 금지되었습니다. 생성자는 공개 생성자로서 문서화, 테스트 또는 설계되지 않았습니다. 사용자는 `ssl.wrap_socket()` 또는 `ssl.SSLContext` 를 사용해야 합니다. (Contributed by Christian Heimes in bpo-32951.)
- 사용되지 않는 `distutils install_misc` 명령이 제거되었습니다. (Contributed by Eric N. Vander Weele in bpo-29218.)

15 모듈 제거

`fpectl` 모듈이 삭제되었습니다. 기본적으로 활성화된 적이 없고, x86-64에서 제대로 작동한 적도 없고, 예기치 않게 C 확장을 깨뜨리는 방식으로 파이썬 ABI를 변경했습니다. (Contributed by Nathaniel J. Smith in bpo-29137.)

16 윈도우 전용 변경

파이썬 런처(py.exe)는 32비트와 64비트 지정자를 받아들일 수 있는데, 마이너 버전을 지정하지 **않아도** 됩니다. 그래서 `py -3.7-32` 뿐만 아니라 `py -3-32` 와 `py -3-64` 도 유효합니다. 또한 *-m*-64와 *-m.n*-64 형식도 이제 받아들이는데, 지정하지 않을 때 32비트가 사용되는 경우도 64비트 파이썬을 강제합니다. 지정된 버전을 사용할 수 없는 경우 py.exe가 에러를 일으키며 종료됩니다. (Contributed by Steve Barnes in bpo-30291.)

런처는 `py -0` 처럼 실행시킬 수 있는데, 설치되어있는 파이썬의 목록을 만들고, 기본값을 애스터리스크로 표시합니다. `py -0p` 를 실행하면 경로가 포함됩니다. py가 일치할 수 없는 버전 지정자로 실행되면 사용 가능한 지정자의 짧은 양식 목록도 인쇄됩니다. (Contributed by Steve Barnes in bpo-30362.)

17 파이썬 3.7로 이식하기

이 섹션에서는 여러분의 코드 수정을 요구하는 앞서 설명한 변경 사항과 버그 수정을 나열합니다.

17.1 파이썬 동작의 변경

- `async`와 `await` 이름은 이제 예약 키워드입니다. 이 이름을 식별자로 사용하는 코드는 이제 `SyntaxError`를 발생시킵니다. (Contributed by Jelle Zijlstra in bpo-30406.)
- **PEP 479** 는 파이썬 3.7의 모든 코드에서 활성화되었습니다. 즉 코루틴과 제너레이터에서 직접 또는 간접적으로 발생하는 `StopIteration` 예외는 `RuntimeError` 예외로 변환됩니다. (Contributed by Yury Selivanov in bpo-32670.)
- `object.__aiter__()` 메서드는 더는 비동기로 선언될 수 없습니다. (Contributed by Yury Selivanov in bpo-31709.)
- 못 보고 넘기는 바람에, 이전 파이썬 버전에서 다음 문법이 받아들여 지는 잘못이 있었습니다:

```
f(1 for x in [1],)

class C(1 for x in [1]):
    pass
```

  파이썬 3.7은 이제 올바르게 `SyntaxError`를 일으킵니다. 제너레이터 표현식은 항상 괄호 안에 바로 들어가 있어야 하며 양쪽에 쉼표를 넣을 수 없고 중복된 괄호는 오직 호출 시에만 생략 할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-32012 and bpo-32023.)

- `-m` 스위치를 사용할 때, 이제 빈 문자열 대신에 초기 작업 디렉터리가 `sys.path`에 추가됩니다 (빈 문자열은 각 임포트 시점의 작업 디렉터리가 동적으로 적용되게 만듭니다). 빈 문자열을 검사하는 프로그램이나 이전 동작에 의존하는 프로그램은 그에 따라 업데이트해야 합니다 (예를 들어, 왜 코드가 빈 문자열을 검사했는지에 따라 `os.getcwd()` 또는 `os.path.dirname(__main__.__file__)` 도 검사해서).

17.2 파이썬 API의 변경

- `socketserver.ThreadingMixIn.server_close()` 는 이제 모든 데몬이 아닌 스레드가 완료될 때까지 대기합니다. 3.7 이전의 동작을 얻으려면 새로운 `socketserver.ThreadingMixIn.block_on_close` 클래스 어트리뷰트를 `False`로 설정하십시오. (Contributed by Victor Stinner in bpo-31233 and bpo-33540.)
- `socketserver.ForkingMixIn.server_close()` 는 이제 모든 자식 프로세스가 완료될 때까지 대기합니다. 3.7 이전의 동작을 얻으려면 새로운 `socketserver.ForkingMixIn.block_on_close` 클래스 어트리뷰트를 `False`로 설정하십시오. (Contributed by Victor Stinner in bpo-31151 and bpo-33540.)
- `locale.localeconv()` 함수는 이제 일시적으로 `LC_CTYPE` 로케일을 `LC_NUMERIC`의 값으로 설정하는 때도 있습니다. (Contributed by Victor Stinner in bpo-31900.)
- `pkgutil.walk_packages()` 는 *path* 가 문자열이면 `ValueError`를 발생시킵니다. 이전에는 빈 리스트가 반환되었습니다. (Contributed by Sanyam Khurana in bpo-24744.)
- `string.Formatter.format()` 의 포맷 문자열 인자는 이제 위치-전용 입니다. 키워드 인자로 전달하는 것은 파이썬 3.5에서 폐지되었습니다. (Contributed by Serhiy Storchaka in bpo-29193.)
- 클래스 `http.cookies.Morsel`의 어트리뷰트 `key`, `value` 및 `coded_value`는 이제 읽기 전용입니다. 그들에게 대입하는 것은 파이썬 3.5에서 폐지되었습니다. 그것들을 설정하려면 `set()` 메서드를 사용하십시오. (Contributed by Serhiy Storchaka in bpo-29192.)

- `os.makedirs()` 의 *mode* 인자는 더는 새로 생성된 중간 수준 디렉터리의 파일 사용 권한 비트에 영향을 미치지 않습니다. 파일 사용 권한 비트를 설정하기 위해서 `makedirs()` 를 호출하기 전에 umask를 설정할 수 있습니다. (Contributed by Serhiy Storchaka in bpo-19930.)

- `struct.Struct.format` 형은 이제 `bytes` 가 아니라 `str` 입니다. (Contributed by Victor Stinner in bpo-21071.)

- `parse_multipart()` 는 이제 *encoding* 및 *errors* 인자를 받아들이고 `FieldStorage` 과 같은 결과를 반환합니다: 파일이 아닌 필드의 경우 키와 연관된 값은 바이트열이 아니라 문자열의 리스트입니다. (Contributed by Pierre Quentel in bpo-29979.)

- `socket` 의 내부 변경으로 인해, 이번 버전의 파이썬에서 `socket.share` 로 만들어진 소켓에 `socket.fromshare()` 를 호출하는 것은 지원되지 않습니다.

- `BaseException` 의 `repr` 은 후행 쉼표를 포함하지 않도록 변경되었습니다. 대부분의 예외는 이 변경의 영향을 받습니다. (Contributed by Serhiy Storchaka in bpo-30399.)

- `datetime.timedelta` 의 `repr` 은 키워드 인자를 출력에 포함하도록 변경되었습니다. (Contributed by Utkarsh Upadhyay in bpo-30302.)

- `shutil.rmtree()` 는 이제 `os.scandir()` 함수를 사용하여 구현되었으므로, 사용자 지정 처리기 *onerror* 는 이제 디렉터리의 목록을 얻는 데 실패하면 첫 번째 인자가 `os.listdir` 대신 `os.scandir` 로 호출됩니다.

- 유니코드 기술 표준 #18 과같이 정규 표현식에서 중첩 집합 및 집합 연산에 대한 지원이 향후 추가될 수 있습니다. 그러면 문법이 변경됩니다. 미래의 변화를 촉진하기 위해 당분간 모호한 경우에 `FutureWarning` 를 일으킬 것입니다. 이 경우는 리터럴 `'['` 로 시작하거나 리터럴 문자 `'--'`, `'&&'`, `'~~'` 및 `'||'` 을 포함하는 집합을 포함합니다. 경고를 피하려면 백 슬래시로 이스케이프 처리하십시오. (Contributed by Serhiy Storchaka in bpo-30349.)

- 빈 문자열과 일치 할 수 있는 정규식 으로 문자열을 나눈 결과가 변경되었습니다. 예를 들어, `r'\s*'` 로 나누면, 이전처럼 공백 문자뿐만 아니라, 공백 문자가 아닌 문자의 앞과 문자열 끝 바로 직전의 빈 문자열로도 나눕니다. 패턴을 `r'\s+'` 로 변경하면 이전의 동작을 복원 할 수 있습니다. 파이썬 3.5부터 그러한 패턴에 대해 `FutureWarning` 을 만들었습니다.

  빈 문자열과 비어 있지 않은 문자열 모두와 일치하는 패턴의 경우, 모든 일치를 검색한 결과가 다른 경우에도 변경될 수 있습니다. 예를 들어, 문자열 `'a\n\n'` 에서, 패턴 `r'(?m)^\s*?$'` 는 위치 2와 3의 빈 문자열을 일치시킬 뿐만 아니라, 위치 2–3의 문자열 `'\n'` 도 일치시킵니다. 빈 줄만 일치시키려면, 패턴을 `r'(?m)^[^\S\n]*$'` 으로 다시 써야 합니다.

  `re.sub()` 는 이제 이전의 비어 있지 않은 일치에 인접한 빈 일치를 치환합니다. 예를 들어 `re.sub('x*', '-', 'abxd')` 는 이제 `'-a-b-d-'` 대신에 `'-a-b--d-'` 를 돌려줍니다 ('b'와 'd' 사이의 첫 번째 마이너스는 'x'를 치환하고 두 번째 마이너스는 'x'와 'd' 사이의 빈 문자열을 치환합니다).

  (Contributed by Serhiy Storchaka in bpo-25054 and bpo-32308.)

- 변경 `re.escape()` 가 ASCII 문자, 숫자 및 `'_'` 이외의 모든 문자를 이스케이프 하는 대신 정규식 특수 문자만 이스케이프 하도록 변경합니다. (Contributed by Serhiy Storchaka in bpo-29995.)

- `tracemalloc.Traceback` 프레임은 이제 가장 오래된 것부터 가장 최근의 것 순으로 정렬되어, `traceback` 과 더 일관성 있게 만듭니다. (Contributed by Jesse Bakker in bpo-32121.)

- `socket.SOCK_NONBLOCK` 또는 `socket.SOCK_CLOEXEC` 비트 플래그를 지원하는 OS에서, `socket.type` 에 이것들을 더는 적용하지 않습니다. 따라서 `if sock.type == socket.SOCK_STREAM` 과 같은 검사는 모든 플랫폼에서 예상대로 작동합니다. (Contributed by Yury Selivanov in bpo-32331.)

- 윈도우에서 표준 핸들을 리디렉션 할 때 `subprocess.Popen` 의 *close_fds* 인자의 기본값이 `False` 에서 `True` 로 변경되었습니다. 이전에 표준 입출력 리디렉션으로 `subprocess.Popen` 을 사용할 때 상속된 핸들에 의존했다면, 이전 동작을 유지하기 위해서 `close_fds=False` 를 넘기거나 `STARTUPINFO.lpAttributeList` 를 사용해야 합니다.

- importlib.machinery.PathFinder.invalidate_caches() – 묵시적으로 importlib.invalidate_caches() 에 영향을 줍니다 – 는 이제 sys.path_importer_cache 에서 None 으로 설정된 엔트리를 삭제합니다. (Contributed by Brett Cannon in bpo-33169.)
- asyncio 에서, loop.sock_recv(), loop.sock_sendall(), loop.sock_accept(), loop.getaddrinfo(), loop.getnameinfo() 는 문서와 일치하도록 적절한 코루틴 메서드로 변경되었습니다. 이전에는, 이 메서드는 asyncio.Future 인스턴스를 반환했습니다. (Contributed by Yury Selivanov in bpo-32327.)
- asyncio.Server.sockets 는 이제 내부 서버 소켓 리스트를 직접 돌려주는 대신 사본을 반환합니다. (Contributed by Yury Selivanov in bpo-32662.)
- Struct.format 는 bytes 인스턴스 대신에 str 인스턴스가 되었습니다. (Contributed by Victor Stinner in bpo-21071.)
- argparse subparsers can now be made mandatory by passing required=True to ArgumentParser.add_subparsers(). (Contributed by Anthony Sottile in bpo-26510.)
- ast.literal_eval() 이 이제 더 엄격해졌습니다. 임의 숫자를 더하거나 빼는 것이 더는 허용되지 않습니다. (Contributed by Serhiy Storchaka in bpo-31778.)
- Calendar.itermonthdates 는 이제 날짜가 0001-01-01 에서 9999-12-31 범위를 벗어나면 일관되게 예외를 일으킵니다. 이러한 예외를 허용할 수 없는 응용 프로그램을 지원하기 위해 새로운 Calendar.itermonthdays3 와 Calendar.itermonthdays4 를 사용할 수 있습니다. 새로운 메서드는 튜플을 반환하고 datetime.date 에 의해 지원되는 범위에 제한받지 않습니다. (Contributed by Alexander Belopolsky in bpo-28292.)
- collections.ChainMap 은 이제 하부 매핑의 순서를 보존합니다. (Contributed by Raymond Hettinger in bpo-32792.)
- concurrent.futures.ThreadPoolExecutor 와 concurrent.futures.ProcessPoolExecutor 의 submit() 메서드는 인터프리터가 종료하는 동안 호출되면 RuntimeError 를 발생시킵니다. (Contributed by Mark Nemec in bpo-33097.)
- configparser.ConfigParser 생성자는 read_dict() 를 사용하여 기본값을 처리해서, 그 동작을 파서의 나머지와 일관되게 합니다. 이제 기본 딕셔너리의 문자열이 아닌 키와 값은 묵시적으로 문자열로 변환됩니다. (Contributed by James Tocknell in bpo-23835.)
- 문서로 만들어지지 않은 여러 가지 내부 임포트가 제거되었습니다. 한 가지 예는 os.errno를 더는 사용할 수 없다는 것입니다; 대신 import errno를 직접 사용하십시오. 이러한 문서로 만들어지지 않은 내부 임포트는, 마이크로 버전 배포에서도, 언제든지 예고 없이 삭제될 수 있음에 유의하십시오.

17.3 C API의 변경

함수 PySlice_GetIndicesEx() 는 크기 조절이 가능한 시퀀스에 대해 안전하지 않은 것으로 간주합니다. 슬라이스 인덱스가 int 의 인스턴스가 아니라 __index__() 메서드를 구현하는 객체인 경우, 시퀀스는 길이를 PySlice_GetIndicesEx() 에 전달한 후 크기를 조정할 수 있습니다. 이로 인해 시퀀스의 길이를 벗어나는 인덱스가 반환될 수 있습니다. 가능한 문제를 피하려면 PySlice_Unpack() 과 PySlice_AdjustIndices() 라는 새로운 함수를 사용하십시오. (Contributed by Serhiy Storchaka in bpo-27867.)

17.4 CPython 바이트 코드 변경

두 개의 새로운 옵코드가 있습니다: `LOAD_METHOD`와 `CALL_METHOD`. (Contributed by Yury Selivanov and INADA Naoki in bpo-26110.)

`STORE_ANNOTATION` 옵코드가 삭제되었습니다. (Contributed by Mark Shannon in bpo-32550.)

17.5 윈도우 전용 변경

`sys.path` 를 재정의하는데 사용되는 파일은 이제 `'sys.path'` 대신에 `<python-executable>._pth` 라고 불립니다. 자세한 정보는 finding_modules 를 보십시오. (Contributed by Steve Dower in bpo-28137.)

17.6 기타 CPython 구현 변경

공개된 CPython 실행시간 초기화 API에 가해질 잠재적인 미래의 변경을 준비하기 위해 (초기의, 하지만 약간 낡은 초안은 **PEP 432**를 보십시오), CPython의 내부 시작 및 구성 관리 로직이 상당히 리팩토링 되었습니다. 이러한 업데이트는 내장형 응용 프로그램과 일반 CPython CLI의 사용자 모두에게 완전히 투명하도록 의도했지만, 리팩토링 변경이 인터프리터 시작 시 다양한 작업의 내부 순서를 변경하므로 내장형 응용 프로그램과 CPython 자체에서 잠재 결함을 드러낼 수 있으므로 여기에서 언급합니다. (처음에는 Nick Coghlan과 Eric Snow가 bpo-22257 의 일부로 이바지했고, Nick, Eric, Victor Stinner가 여러 이슈를 통해 추가로 업데이트했습니다). 영향을 받는 일부 알려진 세부 정보는 이렇습니다:

- `PySys_AddWarnOptionUnicode()` 는 *Py_Initialize*를 호출하기 전에 유니코드 객체를 생성해야 하는 요구사항 때문에 현재 내장형 응용 프로그램에서는 사용할 수 없습니다. 대신에 `PySys_AddWarnOption()` 을 사용하십시오.
- `PySys_AddWarnOption()`으로 내장형 응용 프로그램이 추가한 경고 필터는 이제 인터프리터가 설정한 기본 필터보다 더 일관되게 우선해야 합니다.

기본 경고 필터가 구성되는 방식의 변경으로 인해, `Py_BytesWarningFlag`를 1보다 큰 값으로 설정하는 것이 더는 `BytesWarning` 메시지를 내보내면서 동시에 예외로 변환되도록 하기에 충분하지 못합니다. 대신, 플래그를 설정해야 하고 (경고가 처음에 발생하도록), 예외로 변환하기 위해 명시적으로 `error::BytesWarning` 경고 필터를 추가해야 합니다.

컴파일러가 독스트링을 처리하는 방식의 변화 때문에, 독스트링만으로 구성된 함수 바디의 묵시적인 `return None` 은, 이제 함수의 헤더 행이 아니라 독스트링과 같은 줄에 등장하는 것으로 표시됩니다.

현재 예외 상태가 프레임 객체에서 코루틴으로 옮겨졌습니다. 이는 인터프리터를 간소화하고 제너레이터에 들어가거나 빠져나갈 때 예외 상태를 스와프함으로써 발생하는 모호한 두 가지 버그를 수정했습니다. (Contributed by Mark Shannon in bpo-25612.)

18 파이썬 3.7.1의 주목할만한 변경 사항

3.7.1부터, `Py_Initialize()`는 이제 `Py_Main()`과 같은 환경 설정을 일관되게 읽고 존중합니다 (이전 파이썬 버전에서는, 환경 변수 중 잘 정의되지 않은 부분 집합을 존중했지만 파이썬 3.7.0에서는 bpo-34247로 인해 아무것도 읽지 않았습니다). 이 동작을 원하지 않으면, `Py_Initialize()`를 호출하기 전에 `Py_IgnoreEnvironmentFlag`를 1로 설정하십시오.

3.7.1에서, 컨텍스트 변수를 위한 C API는 `PyObject` 포인터를 사용하도록 변경되었습니다. bpo-34762를 참조하세요.

`xml.dom.minidom` 및 `xml.sax` 모듈은 기본적으로 더는 외부 엔티티를 처리하지 않습니다. bpo-17239를 참조하세요.

3.7.1에서, `tokenize` 모듈은 끝에 줄 바꿈이 없는 입력이 제공될 때 묵시적으로 `NEWLINE` 토큰을 산출합니다. 이 동작은 이제 C 토크나이저가 내부적으로 수행하는 것과 일치합니다. (Contributed by Ammar Askar in bpo-33899.)

19 파이썬 3.7.2의 주목할만한 변경 사항

3.7.2에서, 윈도우에서 `venv`는 더는 원본 바이너리를 복사하지 않고, 대신 `python.exe` 와 `pythonw.exe` 라는 리디렉터 스크립트를 만듭니다. 이렇게 하면 각 파이썬 업데이트마다 모든 가상 환경을 업그레이드하거나 다시 만들어야 하는 오랜 문제를 해결할 수 있습니다. 그러나, 이 배포에서는 새 스크립트를 얻기 위해 가상 환경을 다시 만들어야 함에 유의하세요.

20 Notable changes in Python 3.7.6

Due to significant security concerns, the *reuse_address* parameter of `asyncio.loop.create_datagram_endpoint()` is no longer supported. This is because of the behavior of the socket option `SO_REUSEADDR` in UDP. For more details, see the documentation for `loop.create_datagram_endpoint()`. (Contributed by Kyle Stanley, Antoine Pitrou, and Yury Selivanov in bpo-37228.)

21 Notable changes in Python 3.7.10

Earlier Python versions allowed using both `;` and `&` as query parameter separators in `urllib.parse.parse_qs()` and `urllib.parse.parse_qsl()`. Due to security concerns, and to conform with newer W3C recommendations, this has been changed to allow only a single separator key, with `&` as the default. This change also affects `cgi.parse()` and `cgi.parse_multipart()` as they use the affected functions internally. For more details, please see their respective documentation. (Contributed by Adam Goldschmidt, Senthil Kumaran and Ken Jin in bpo-42967.)

22 Notable changes in Python 3.7.11

A security fix alters the `ftplib.FTP` behavior to not trust the IPv4 address sent from the remote server when setting up a passive data channel. We reuse the ftp server IP address instead. For unusual code requiring the old behavior, set a `trust_server_pasv_ipv4_address` attribute on your FTP instance to `True`. (See bpo-43285)

The presence of newline or tab characters in parts of a URL allows for some forms of attacks. Following the WHATWG specification that updates RFC 3986, ASCII newline `\n`, `\r` and tab `\t` characters are stripped from the URL by the parser `urllib.parse()` preventing such attacks. The removal characters are controlled by a new module level variable `urllib.parse._UNSAFE_URL_BYTES_TO_REMOVE`. (See bpo-43882)

23 Notable security feature in 3.7.14

Converting between `int` and `str` in bases other than 2 (binary), 4, 8 (octal), 16 (hexadecimal), or 32 such as base 10 (decimal) now raises a `ValueError` if the number of digits in string form is above a limit to avoid potential denial of service attacks due to the algorithmic complexity. This is a mitigation for CVE-2020-10735. This limit can be configured or disabled by environment variable, command line flag, or `sys` APIs. See the integer string conversion length limitation documentation. The default limit is 4300 digits in string form.

색인

P

R

S

Y