
What's New in Python

출시 버전 3.9.23

A. M. Kuchling

7월 09, 2025

Contents

1	요약 - 배포 주요 사항	3
2	여러분의 코드에서 DeprecationWarning 을 확인해야 합니다	4
3	새로운 기능	4
3.1	딕셔너리 병합과 업데이트 연산자	4
3.2	접두사와 접미사를 제거하는 새로운 문자열 메서드	5
3.3	표준 컬렉션의 형 힌트 제네릭	5
3.4	새로운 구문 분석기	5
4	기타 언어 변경	5
5	새 모듈	6
5.1	zoneinfo	6
5.2	graphlib	7
6	개선된 모듈	7
6.1	ast	7
6.2	asyncio	7
6.3	compileall	7
6.4	concurrent.futures	8
6.5	curses	8
6.6	datetime	8
6.7	distutils	8
6.8	fcntl	8
6.9	ftplib	8
6.10	gc	8
6.11	hashlib	9
6.12	http	9
6.13	IDLE과 idlelib	9
6.14	imaplib	9
6.15	importlib	10
6.16	inspect	10
6.17	ipaddress	10
6.18	math	10
6.19	multiprocessing	10
6.20	nntplib	11

6.21	os	11
6.22	pathlib	11
6.23	pdb	11
6.24	poplib	11
6.25	pprint	11
6.26	pydoc	12
6.27	random	12
6.28	signal	12
6.29	smtplib	12
6.30	socket	12
6.31	time	12
6.32	sys	12
6.33	tempfile	13
6.34	tracemalloc	13
6.35	typing	13
6.36	unicodedata	13
6.37	venv	13
6.38	xml	13
7	최적화	13
8	폐지	15
9	제거	16
10	파이썬 3.9로 이식하기	18
10.1	파이썬 API의 변경	18
10.2	C API의 변경	19
10.3	CPython 바이트 코드 변경	19
11	빌드 변경	20
12	C API 변경	20
12.1	새로운 기능	20
12.2	파이썬 3.9로 이식하기	21
12.3	제거	22
13	파이썬 3.9.1의 주목할만한 변경 사항	24
13.1	typing	24
13.2	macOS 11.0 (Big Sur) 및 Apple Silicon Mac 지원	24
14	파이썬 3.9.2의 주목할만한 변경 사항	24
14.1	collections.abc	24
14.2	urllib.parse	25
15	파이썬 3.9.3의 주목할만한 변경 사항	25
16	Notable changes in Python 3.9.5	25
16.1	urllib.parse	25
17	Notable security feature in 3.9.14	25
18	Notable Changes in 3.9.17	25
18.1	tarfile	25
19	Notable changes in 3.9.20	26

19.1	ipaddress	26
19.2	email	26
20	Notable changes in 3.9.23	26
20.1	os.path	26
20.2	tarfile	26
	색인	27

배포 3.9.23

날짜 7월 09, 2025

편집자 Łukasz Langa

이 기사에서는 3.8과 비교하여 파이썬 3.9의 새로운 기능에 대해 설명합니다. 파이썬 3.9는 2020년 10월 5일에 출시되었습니다.

자세한 내용은 changelog를 참조하십시오.

더 보기:

PEP 596 - 파이썬 3.9 출시 일정

1 요약 - 배포 주요 사항

새로운 문법 기능:

- **PEP 584**, 병합(union) 연산자가 dict에 추가되었습니다.
- **PEP 585**, 표준 컬렉션의 형 힌트 제네릭;
- **PEP 614**, 데코레이터에 대한 완화된 문법 제한.

새로운 내장 기능:

- **PEP 616**, 접두사와 접미사를 제거하는 문자열 메서드.

표준 라이브러리의 새로운 기능:

- **PEP 593**, 유연한 함수와 변수 어노테이션;
- 경쟁과 시그널 없이 프로세스 관리를 허용하는 `os.pidfd_open()`이 추가되었습니다.

인터프리터 개선:

- **PEP 573**, C 확장형의 메서드에서 모듈 상태에 빠르게 액세스하기;
- **PEP 617**, CPython은 이제 PEG를 기반으로 하는 새로운 구문 분석기를 사용합니다;
- 많은 파이썬 내장(range, tuple, set, frozenset, list, dict)이 이제 **PEP 590** 벡터콜을 사용하여 빨라졌습니다;
- 가비지 수거는 부활한 객체에서 블록되지 않습니다;
- 많은 파이썬 모듈(_abc, audioop, _bz2, _codecs, _contextvars, _crypt, _functools, _json, _locale, math, operator, resource, time, _weakref)은 이제 **PEP 489**에 정의된 대로 다단계 초기화를 사용합니다;
- 많은 표준 라이브러리 모듈(audioop, ast, grp, _hashlib, pwd, _posixsubprocess, random, select, struct, termios, zlib)은 이제 **PEP 384**에서 정의한 안정(stable) ABI를 사용하고 있습니다.

새로운 라이브러리 모듈:

- **PEP 615**, IANA 시간대 데이터베이스는 이제 zoneinfo 모듈을 통해 표준 라이브러리에 있습니다;
- 이제 새로운 graphlib 모듈에서 그래프의 위상 정렬(topological sort) 구현이 제공됩니다.

릴리스 프로세스 변경:

- **PEP 602**, CPython은 연간 릴리스 주기를 채택합니다.

2 여러분의 코드에서 DeprecationWarning을 확인해야 합니다

파이썬 2.7이 여전히 지원될 때, 파이썬 2.7과의 과거 호환성을 위해 파이썬 3의 많은 기능이 유지되었습니다. 파이썬 2.7 지원이 종료되면서, 이러한 이전 버전과의 호환성 계층이 제거되었거나, 곧 제거될 예정입니다. 그들 대부분은 몇 년 동안 DeprecationWarning 경고를 내보냈습니다. 예를 들어, collections.abc.Mapping 대신 collections.Mapping을 사용하면 2012년에 배포된 파이썬 3.3 이후 DeprecationWarning을 내보냅니다.

-W default 명령 줄 옵션으로 응용 프로그램을 테스트하여 DeprecationWarning과 PendingDeprecationWarning을 보십시오. 또는 -W error로 아예 에러로 취급하십시오. 경고 필터를 사용하면 제삼자 코드의 경고를 무시할 수 있습니다.

파이썬 3.9는 파이썬 프로젝트 지원 담당자가 파이썬 2 지원 제거를 구성하고 파이썬 3.9에 대한 지원을 추가할 수 있도록 더 많은 시간을 제공하기 위한, 파이썬 2 과거 호환성 계층을 제공하는 마지막 버전입니다.

collections 모듈의 추상 베이스 클래스에 대한 별칭(collections.abc.Mapping에 대한 collections.Mapping 별칭과 같은)은 이전 버전과의 호환성을 위해 마지막 배포 하나에서 유지됩니다. 파이썬 3.10에서 제거됩니다.

더 일반적으로, 파이썬 개발 모드에서 테스트를 실행하면 다음 파이썬 버전과 호환되도록 코드를 준비하는 데 도움이 됩니다.

참고: 이 버전의 파이썬에서 기존의 많은 폐지가 제거되기도 했습니다. [제거](#) 섹션을 참조하십시오.

3 새로운 기능

3.1 딕셔너리 병합과 업데이트 연산자

내장 dict 클래스에 병합(|)과 업데이트(|=) 연산자가 추가되었습니다. 이들은 딕셔너리 병합의 기존 dict.update와 {**d1, **d2} 메서드를 보완합니다.

예:

```
>>> x = {"key1": "value1 from x", "key2": "value2 from x"}
>>> y = {"key2": "value2 from y", "key3": "value3 from y"}
>>> x | y
{'key1': 'value1 from x', 'key2': 'value2 from y', 'key3': 'value3 from y'}
>>> y |= x
{'key2': 'value2 from x', 'key3': 'value3 from y', 'key1': 'value1 from x'}
```

자세한 설명은 **PEP 584**를 참조하십시오. (Contributed by Brandt Bucher in [bpo-36144](#).)

3.2 접두사와 접미사를 제거하는 새로운 문자열 메서드

문자열에서 불필요한 접두사나 접미사를 쉽게 제거하기 위해 `str.removeprefix(prefix)`와 `str.removesuffix(suffix)`가 추가되었습니다. 해당 `bytes`, `bytearray` 및 `collections.UserString` 메서드도 추가되었습니다. 자세한 설명은 [PEP 616](#)을 참조하십시오. (Contributed by Dennis Sweeney in [bpo-39939](#).)

3.3 표준 컬렉션의 형 힌트 제네릭

형 어노테이션에서 이제 `typing`에서 해당 대문자 형(예를 들어 `List`나 `Dict`)을 импорт 하는 대신 `list`와 `dict`와 같은 내장 컬렉션 형을 제네릭 형으로 사용할 수 있습니다. 표준 라이브러리의 일부 다른 형도 이제 제네릭입니다, 예를 들어 `queue.Queue`.

예:

```
def greet_all(names: list[str]) -> None:
    for name in names:
        print("Hello", name)
```

자세한 내용은 [PEP 585](#)를 참조하십시오. (Contributed by Guido van Rossum, Ethan Smith, and Batuhan Taşkaya in [bpo-39481](#).)

3.4 새로운 구문 분석기

파이썬 3.9는 [LL\(1\)](#) 대신 [PEG](#)를 기반으로 하는 새로운 구문 분석기를 사용합니다. 새로운 구문 분석기의 성능은 기존 구문 분석기의 성능과 거의 비슷하지만, [PEG](#) 형식은 새로운 언어 기능을 설계할 때 [LL\(1\)](#)보다 더 유연합니다. 파이썬 3.10 이상에서 이 유연성을 사용하기 시작할 것입니다.

`ast` 모듈은 새 구문 분석기를 사용하며 이전 구문 분석기와 같은 `AST`를 생성합니다.

파이썬 3.10에서는, 이전 구문 분석기가 삭제되어, 여기에 의존하는 모든 기능도 제거됩니다 (주로 오랜 기간 폐지되었던 `parser` 모듈). 파이썬 3.9에서 *만*, 명령 줄 스위치(`-X oldparser`)나 환경 변수(`PYTHONOLDPARSER=1`)를 사용하여 [LL\(1\)](#) 구문 분석기로 다시 전환 할 수 있습니다.

자세한 내용은 [PEP 617](#)을 참조하십시오. (Contributed by Guido van Rossum, Pablo Galindo and Lysandros Nikolaou in [bpo-40334](#).)

4 기타 언어 변경

- `__import__()`는 이제 `ValueError` 대신 `ImportError`를 발생시킵니다. 이는 상대적 임포트가 최상위 수준 패키지를 넘어갈 때 발생했습니다. (Contributed by Ngalm Siregar in [bpo-37444](#).)
- 파이썬은 이제 명령 줄에 지정된 스크립트 파일 이름의 절대 경로를 연습니다 (예: `python3 script.py`): `__main__` 모듈의 `__file__` 어트리뷰트는 상대 경로 대신 절대 경로가 됩니다. 이 경로는 현재 디렉터리가 `os.chdir()`에 의해 변경된 후에도 계속 유효합니다. 부작용으로, 이 경우 트래이스백은 `__main__` 모듈 프레임에 대해 절대 경로를 표시합니다. (Contributed by Victor Stinner in [bpo-20443](#).)
- 파이썬 개발 모드와 디버그 빌드에서, 이제 문자열 인코딩과 디코딩 연산을 위해 `encoding`과 `errors` 인자가 검사됩니다. 예: `open()`, `str.encode()` 및 `bytes.decode()`.

기본적으로, 최상의 성능을 위해, `errors` 인자는 첫 번째 인코딩/디코딩 예러에서만 확인되며 빈 문자열에 대해서는 `encoding` 인자가 무시되는 경우가 있습니다. (Contributed by Victor Stinner in [bpo-37388](#).)

- `"".replace("", s, n)` 는 이제 0이 아닌 모든 `n`에 대해 빈 문자열 대신 `s`를 반환합니다. 이제 `"".replace("", s)` 와 일관성 있습니다. `bytes`와 `bytearray` 객체에 대해 유사한 변경 사항이 있습니다. (Contributed by Serhiy Storchaka in [bpo-28029](#).)
- 이제 모든 유효한 표현식을 데코레이터로 사용할 수 있습니다. 이전에는, 문법이 훨씬 제한적이었습니다. 자세한 내용은 [PEP 614](#)를 참조하십시오. (Contributed by Brandt Bucher in [bpo-39702](#).)
- `typing` 모듈에 대한 도움말이 개선되었습니다. 모든 특수 형식과 특수 제네릭 에일리어스(`Union`과 `List` 같은)에 대한 독스트링이 이제 표시됩니다. `List[int]`와 같은 제네릭 에일리어스로 `help()`를 사용하면 해당 구상형(이 경우 `list`)에 대한 도움말이 표시됩니다. (Contributed by Serhiy Storchaka in [bpo-40257](#).)
- `aclose()` / `asend()` / `athrow()`의 병렬 실행은 이제 금지되며, 이제 `ag_running`은 비동기 제너레이터의 실제 실행 상태를 반영합니다. (Contributed by Yuri Selivanov in [bpo-30773](#).)
- `__iter__` 메서드 호출 시 예기치 않은 예러가 더는 `in` 연산자와 `operator` 모듈의 `contains()`, `indexOf()` 및 `countOf()` 함수에서 `TypeError`에 의해 마스킹 되지 않습니다. (Contributed by Serhiy Storchaka in [bpo-40824](#).)
- Unparenthesized lambda expressions can no longer be the expression part in an `if` clause in comprehensions and generator expressions. See [bpo-41848](#) and [bpo-43755](#) for details.

5 새 모듈

5.1 zoneinfo

`zoneinfo` 모듈은 IANA 시간대 데이터베이스 지원을 표준 라이브러리에 도입합니다. 시스템의 시간대 데이터로 뒷받침되는 구상 `datetime.tzinfo` 구현인 `zoneinfo.ZoneInfo`를 추가합니다.

예:

```
>>> from zoneinfo import ZoneInfo
>>> from datetime import datetime, timedelta

>>> # Daylight saving time
>>> dt = datetime(2020, 10, 31, 12, tzinfo=ZoneInfo("America/Los_Angeles"))
>>> print(dt)
2020-10-31 12:00:00-07:00
>>> dt.tzname()
'PDT'

>>> # Standard time
>>> dt += timedelta(days=7)
>>> print(dt)
2020-11-07 12:00:00-08:00
>>> print(dt.tzname())
'PST'
```

IANA 데이터베이스를 제공하지 않는 플랫폼의 대체 데이터 소스로, `tzdata` 모듈이 자사(first-party) 패키지로 출시되었습니다 - PyPI를 통해 배포되고 CPython 핵심 팀에서 유지 관리합니다.

더 보기:

PEP 615 - 표준 라이브러리의 IANA 시간대 데이터베이스에 대한 지원 Paul Ganssle이 작성하고 구현한 PEP

5.2 graphlib

`graphlib.TopologicalSorter` 클래스를 포함하는 `graphlib`를 추가하여 그래프의 위상 정렬을 수행하는 기능을 제공합니다. (Contributed by Pablo Galindo, Tim Peters and Larry Hastings in [bpo-17005](#).)

6 개선된 모듈

6.1 ast

`indent` 옵션을 `dump()`에 추가하여 여러 줄 들여쓰기 된 출력을 생성할 수 있습니다. (Contributed by Serhiy Storchaka in [bpo-37995](#).)

`ast.AST` 객체를 역 구문 분석하고, 구문 분석할 때 동등한 `ast.AST` 객체를 생성하는 코드가 담긴 문자열을 생성하는 데 사용할 수 있는 `ast.unparse()`를 `ast` 모듈의 함수로 추가했습니다. (Contributed by Pablo Galindo and Batuhan Taskaya in [bpo-38870](#).)

AST 노드를 생성하는 데 사용된 ASDL 서명이 포함된 독스트링을 AST 노드에 추가했습니다. (Contributed by Batuhan Taskaya in [bpo-39638](#).)

6.2 asyncio

심각한 보안 문제로 인해, `asyncio.loop.create_datagram_endpoint()`의 `reuse_address` 매개 변수는 더는 지원되지 않습니다. 이것은 UDP에서 소켓 옵션 `SO_REUSEADDR`의 동작 때문입니다. 자세한 내용은 `loop.create_datagram_endpoint()` 설명서를 참조하십시오. (Contributed by Kyle Stanley, Antoine Pitrou, and Yuri Selivanov in [bpo-37228](#).)

`ThreadPoolExecutor`가 닫기를 끝내기를 기다리는 기본 실행기의 종료(`shutdown`)를 예약하는 새 코루틴 `shutdown_default_executor()`가 추가되었습니다. 또한, `asyncio.run()`은 새로운 코루틴을 사용하도록 갱신되었습니다. (Contributed by Kyle Stanley in [bpo-34037](#).)

프로세스 파일 기술자를 폴링 하는 리눅스 특정 자식 감시자 구현인, `asyncio.PidfdChildWatcher`를 추가했습니다. ([bpo-38692](#))

새로운 코루틴 `asyncio.to_thread()`를 추가했습니다. 주로 이벤트 루프 블록을 피하고자 별도의 스레드에서 IO 병목 함수를 실행하는 데 사용되며, 기본적으로 키워드 인자를 직접 취할 수 있는 `run_in_executor()`의 고수준 버전으로 작동합니다. (Contributed by Kyle Stanley and Yuri Selivanov in [bpo-32309](#).)

시간 초과로 인해 태스크를 취소할 때, `asyncio.wait_for()`는 이제 양의 `timeout`과 마찬가지로 `timeout`이 `<= 0`인 경우에도 취소가 완료될 때까지 기다립니다. (Contributed by Elvis Pranskevichus in [bpo-32751](#).)

`asyncio`는 이제 `ssl.SSLSocket` 소켓과 호환되지 않는 메서드를 호출할 때 `TypeError`를 발생시킵니다. (Contributed by Ido Michael in [bpo-37404](#).)

6.3 compileall

복제된 `.pyc` 파일에 대해 하드 링크를 사용하는 새로운 가능성이 추가되었습니다: `hardlink_dupes` 매개 변수와 `-hardlink-dupes` 명령 줄 옵션. (Contributed by Lumír ‘Frenzy’ Balhar in [bpo-40495](#).)

결과 `.pyc` 파일에서 경로 조작을 위한 새로운 옵션을 추가했습니다: `stripdir`, `prependdir`, `limit_sl_dest` 매개 변수와 `-s`, `-p`, `-e` 명령 줄 옵션. 최적화 수준에 대한 옵션을 여러 번 지정할 수 있는 가능성이 추가되었습니다. (Contributed by Lumír ‘Frenzy’ Balhar in [bpo-38112](#).)

6.4 concurrent.futures

실행기를 종료하기 전에 완료되기를 기다리는 대신, 실행을 시작하지 않은 모든 계류 중인 퓨처를 취소하는 새 `cancel_futures` 매개 변수를 `concurrent.futures.Executor.shutdown()` 에 추가했습니다. (Contributed by Kyle Stanley in [bpo-39349](#).)

`ThreadPoolExecutor`와 `ProcessPoolExecutor` 에서 데몬 스레드를 제거했습니다. 이를 통해 종료 절차에서 서브 인터프리터와의 호환성과 예측성이 향상됩니다. (Contributed by Kyle Stanley in [bpo-39812](#).)

재사용 가능한 유틸 작업자가 없을 때만, `ProcessPoolExecutor` 의 작업자가 요청 시 스폰 됩니다. 이는 시작 오버헤드를 최적화하고 유틸 작업자에게 손실되는 CPU 시간을 줄입니다. (Contributed by Kyle Stanley in [bpo-39207](#).)

6.5 curses

`curses.get_escdelay()`, `curses.set_escdelay()`, `curses.get_tabsize()` 및 `curses.set_tabsize()` 함수를 추가했습니다. (Contributed by Anthony Sottile in [bpo-38312](#).)

6.6 datetime

`datetime.date`의 `isocalendar()` 와 `datetime.datetime`의 `isocalendar()` 메서드는 이제 tuple 대신 `namedtuple()` 을 반환합니다. (Contributed by Dong-hee Na in [bpo-24416](#).)

6.7 distutils

upload 명령은 이제 SHA2-256 및 Blake2b-256 해시 다이제스트를 만듭니다. MD5 다이제스트를 차단하는 플랫폼에서는 MD5를 건너뛸니다. (Contributed by Christian Heimes in [bpo-40698](#).)

6.8fcntl

상수 `F_OFD_GETLK`, `F_OFD_SETLK` 및 `F_OFD_SETLKW`를 추가했습니다. (Contributed by Dong-hee Na in [bpo-38602](#).)

6.9 ftplib

비 블로킹 소켓의 생성을 막기 위해 생성자에 대해 주어진 시간제한이 0이면 `FTP` 와 `FTP_TLS` 는 이제 `ValueError`를 발생시킵니다. (Contributed by Dong-hee Na in [bpo-39259](#).)

6.10 gc

가비지 수거기가 일부 객체를 되살리는 컬렉션을 만들 때 (파이널라이저가 실행된 후 격리된 순환 외부에서 도달할 수 있습니다), 여전히 도달할 수 없는 모든 객체의 컬렉션을 차단하지 않습니다. (Contributed by Pablo Galindo and Tim Peters in [bpo-38379](#).)

가비지 수거기에서 객체가 파이널라이즈되었는지 확인하는 새 함수 `gc.is_finalized()` 를 추가했습니다. (Contributed by Pablo Galindo in [bpo-39322](#).)

6.11 hashlib

hashlib 모듈은 이제 사용할 수 있으면 OpenSSL의 SHA3 해시와 SHAKE XOF를 사용할 수 있습니다. (Contributed by Christian Heimes in [bpo-37630](#).)

내장 해시 모듈은 이제 `./configure --without-builtin-hashlib-hashes`로 비활성화하거나 `./configure --with-builtin-hashlib-hashes=sha3,blake2`로 선택적으로 활성화하여 OpenSSL 기반 구현의 사용을 강제할 수 있습니다. (Contributed by Christian Heimes in [bpo-40479](#))

6.12 http

HTTP 상태 코드 103 `EARLY_HINTS`, 418 `IM_A_TEAPOT` 및 425 `TOO_EARLY`가 `http.HTTPStatus`에 추가되었습니다. (Contributed by Dong-hee Na in [bpo-39509](#) and Ross Rhodes in [bpo-39507](#).)

6.13 IDLE과 idlib

커서 깜박임을 토글하는 옵션이 추가되었습니다. (Contributed by Zackery Spytz in [bpo-4603](#).)

이스케이프 키는 이제 IDLE 완료 창을 닫습니다. (Contributed by Johnny Najera in [bpo-38944](#).)

모듈 이름 완성 리스트에 키워드를 추가했습니다. (Contributed by Terry J. Reedy in [bpo-37765](#).)

New in 3.9 maintenance releases

Make IDLE invoke `sys.excepthook()` (when started without `-n`). User hooks were previously ignored. (Contributed by Ken Hilton in [bpo-43008](#).)

위의 변경 사항은 3.8 유지 보수 릴리스로 역 이식되었습니다.

Rearrange the settings dialog. Split the General tab into Windows and Shell/Ed tabs. Move help sources, which extend the Help menu, to the Extensions tab. Make space for new options and shorten the dialog. The latter makes the dialog better fit small screens. (Contributed by Terry Jan Reedy in [bpo-40468](#).) Move the indent space setting from the Font tab to the new Windows tab. (Contributed by Mark Roseman and Terry Jan Reedy in [bpo-33962](#).)

Apply syntax highlighting to `.pyi` files. (Contributed by Alex Waygood and Terry Jan Reedy in [bpo-45447](#).)

6.14 imaplib

IMAP4와 IMAP4_SSL에는 이제 생성자에 선택적 `timeout` 매개 변수가 있습니다. 또한 `open()` 메서드에는 이제 이 변경으로 인해 선택적 `timeout` 매개 변수가 있습니다. 재정의된 IMAP4_SSL과 IMAP4_stream 메서드에서 이 변경이 적용되었습니다. (Contributed by Dong-hee Na in [bpo-38615](#).)

`imaplib.IMAP4.unselect()`가 추가되었습니다. `imaplib.IMAP4.unselect()`는 선택한 사서함과 관련된 서버 자원을 해제하고 서버를 인증된 상태로 되돌립니다. 이 명령은 현재 선택된 사서함에서 메시지가 영구적으로 제거되지 않는다는 점을 제외하고 `imaplib.IMAP4.close()`와 같은 작업을 수행합니다. (Contributed by Dong-hee Na in [bpo-40375](#).)

6.15 importlib

`import` 문과의 일관성을 개선하기 위해, `importlib.util.resolve_name()` 은 이제 잘못된 상대 임포트 시도에 대해 `ValueError` 대신 `ImportError`를 발생시킵니다. (Contributed by Ngalim Siregar in [bpo-37444](#).)

불변 모듈 객체를 게시하는 임포트 로더는 이제 개별 모듈 외에도 불변 패키지를 게시할 수 있습니다. (Contributed by Dino Viehland in [bpo-39336](#).)

`importlib_resources` 버전 1.5의 역 이식과 일치하는 패키지 데이터의 하위 디렉터리를 지원하는 `importlib.resources.files()` 함수가 추가되었습니다. (Contributed by Jason R. Coombs in [bpo-39791](#).)

`importlib_metadata` 버전 1.6.1로부터 `importlib.metadata`를 새로 고쳤습니다.

6.16 inspect

`inspect.BoundArguments.arguments`가 `OrderedDict`에서 일반 `dict`로 변경되었습니다. (Contributed by Inada Naoki in [bpo-36350](#) and [bpo-39775](#).)

6.17 ipaddress

`ipaddress`는 이제 IPv6 스코프 된 주소(접미사 `%<scope_id>` 가 있는 IPv6 주소)를 지원합니다.

스코프 된 IPv6 주소는 `ipaddress.IPv6Address`를 사용하여 구문 분석할 수 있습니다. 존재하면, `scope_id` 어트리뷰트를 통해 스코프 존 ID를 사용할 수 있습니다. (Contributed by Oleksandr Pavliuk in [bpo-34788](#).)

Starting with Python 3.9.5 the `ipaddress` module no longer accepts any leading zeros in IPv4 address strings. (Contributed by Christian Heimes in [bpo-36384](#).)

6.18 math

여러 인자를 처리하도록 `math.gcd()` 함수를 확장했습니다. 이전에는 두 개의 인자만 지원했습니다. (Contributed by Serhiy Storchaka in [bpo-39648](#).)

`math.lcm()` 을 추가했습니다: 지정된 인자의 최소 공배수를 반환합니다. (Contributed by Mark Dickinson, Ananthakrishnan and Serhiy Storchaka in [bpo-39479](#) and [bpo-39648](#).)

`math.nextafter()` 를 추가했습니다: y 를 향해 x 뒤의 다음 부동 소수점 값을 반환합니다. (Contributed by Victor Stinner in [bpo-39288](#).)

`math.ulp()` 를 추가했습니다: 부동 소수점의 최하위 비트 값을 반환합니다. (Contributed by Victor Stinner in [bpo-39310](#).)

6.19 multiprocessing

`multiprocessing.SimpleQueue` 클래스에는 큐를 명시적으로 닫는 새로운 `close()` 메서드가 있습니다. (Contributed by Victor Stinner in [bpo-30966](#).)

6.20 nntplib

생성자에 대해 주어진 시간제한이 0이면 비 블로킹 소켓을 만드는 것을 막기 위해 NNTP와 NNTP_SSL은 이제 `ValueError`를 발생시킵니다. (Contributed by Dong-hee Na in [bpo-39259](#).)

6.21 os

`si_code`에 `CLD_KILLED`와 `CLD_STOPPED`를 추가했습니다. (Contributed by Dong-hee Na in [bpo-38493](#).)

파일 기술자를 사용한 프로세스 관리를 위해 리눅스 특정 `os.pidfd_open()` ([bpo-38692](#)) 과 `os.P_PIDFD` ([bpo-38713](#))를 노출했습니다.

`os.unsetenv()` 함수는 이제 윈도우에서도 사용할 수 있습니다. (Contributed by Victor Stinner in [bpo-39413](#).)

`os.putenv()` 와 `os.unsetenv()` 함수를 이제 항상 사용할 수 있습니다. (Contributed by Victor Stinner in [bpo-39395](#).)

`os.waitstatus_to_exitcode()` 함수를 추가했습니다: 대기 상태를 종료 코드로 변환합니다. (Contributed by Victor Stinner in [bpo-40094](#).)

As of 3.9.20, `os.mkdir()` and `os.makedirs()` on Windows now support passing a *mode* value of `0o700` to apply access control to the new directory. This implicitly affects `tempfile.mkdtemp()` and is a mitigation for CVE-2024-4030. Other values for *mode* continue to be ignored. (Contributed by Steve Dower in [gh-118486](#).)

6.22 pathlib

`os.readlink()` 와 유사하게 작동하는 `pathlib.Path.readlink()` 를 추가했습니다. (Contributed by Girts Folkmanis in [bpo-30618](#))

6.23 pdb

이제 윈도우에서 `Pdb`는 `~/.pdbrc`를 지원합니다. (Contributed by Tim Hopper and Dan Lidral-Porter in [bpo-20523](#).)

6.24 poplib

생성자에 대해 주어진 시간제한이 0이면 비 블로킹 소켓을 만드는 것을 막기 위해 POP3과 POP3_SSL은 이제 `ValueError`를 발생시킵니다. (Contributed by Dong-hee Na in [bpo-39259](#).)

6.25 pprint

`pprint`는 이제 `types.SimpleNamespace`를 예쁘게 인쇄 할 수 있습니다. (Contributed by Carl Bordum Hansen in [bpo-37376](#).)

6.26 pydoc

독스트링은 이제 클래스, 함수, 메서드 등뿐만 아니라 자체 `__doc__` 어트리뷰트를 가진 모든 객체에 대해 표시됩니다. (Contributed by Serhiy Storchaka in [bpo-40257](#).)

6.27 random

새로운 `random.Random.randbytes` 메서드를 추가했습니다: 무작위 바이트열을 생성합니다. (Contributed by Victor Stinner in [bpo-40286](#).)

6.28 signal

`pid` 대신 파일 기술자를 사용하여 프로세스에 시그널을 보내는 리눅스 특정 `signal.pidfd_send_signal()` 을 노출했습니다. ([bpo-38712](#))

6.29 smtplib

생성자에 대해 주어진 시간제한이 0이면 비 블로킹 소켓을 만드는 것을 막기 위해 `SMTP`와 `SMTP_SSL`은 이제 `ValueError`를 발생시킵니다. (Contributed by Dong-hee Na in [bpo-39259](#).)

`LMTP` 생성자는 이제 선택적 `timeout` 매개 변수를 갖습니다. (Contributed by Dong-hee Na in [bpo-39329](#).)

6.30 socket

`socket` 모듈은 이제 리눅스 4.1 이상에서 `CAN_RAW_JOIN_FILTERS` 상수를 내보냅니다. (Contributed by Stefan Tatschner and Zackery Spytz in [bpo-25780](#).)

`socket` 모듈은 이제 `CAN_J1939` 프로토콜을 지원하는 플랫폼에서 이를 지원합니다. (Contributed by Karl Ding in [bpo-40291](#).)

The `socket` module now has the `socket.send_fds()` and `socket.recv_fds()` functions. (Contributed by Joannah Nanjey, Shinya Okano and Victor Stinner in [bpo-28724](#).)

6.31 time

AIX에서, `thread_time()` 은 이제 10ms 해상도의 `clock_gettime(CLOCK_THREAD_CPUTIME_ID)` 가 아니라 나노초 해상도의 `thread_cputime()` 으로 구현됩니다. (Contributed by Batuhan Taskaya in [bpo-40192](#))

6.32 sys

새로운 `sys.platlibdir` 어트리뷰트를 추가했습니다: 플랫폼별 라이브러리 디렉터리의 이름. 표준 라이브러리의 경로와 설치된 확장 모듈의 경로들을 빌드하는 데 사용됩니다. 대부분 플랫폼에서 `"lib"`와 같습니다. Fedora와 SuSE에서는, 64비트 플랫폼에서 `"lib64"`와 같습니다. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in [bpo-129459](#).)

이전에는, 비 대화형일 때 `sys.stderr`가 블록 버퍼링 되었습니다. 이제 `stderr`은 기본적으로 항상 줄 버퍼링 됩니다. (Contributed by Jendrik Seipp in [bpo-13601](#).)

6.33 tempfile

As of 3.9.20 on Windows, the default mode `0o700` used by `tempfile.mkdtemp()` now limits access to the new directory due to changes to `os.mkdir()`. This is a mitigation for CVE-2024-4030. (Contributed by Steve Dower in [gh-118486](#).)

6.34 tracemalloc

특정 코드 조각의 최대 사용량을 측정하기 위해 추적한 메모리 블록의 최대 크기를 현재 크기로 설정하는 `tracemalloc.reset_peak()` 가 추가되었습니다. (Contributed by Huon Wilson in [bpo-40630](#).)

6.35 typing

PEP 593은 문맥 별 메타 데이터로 기존 형을 데코레이트 하는 `typing.Annotated` 형과 실행 시간에 메타 데이터에 액세스하기 위해 `typing.get_type_hints()` 에 새로운 `include_extras` 매개 변수를 도입했습니다. (Contributed by Till Varoquaux and Konstantin Kashin.)

6.36 unicodedata

유니코드 데이터베이스가 버전 13.0.0으로 갱신되었습니다. ([bpo-39926](#)).

6.37 venv

`venv`가 제공하는 활성화 스크립트는 이제 모두 항상 `__VENV_PROMPT__`로 지정된 값을 사용하여 프롬프트 사용자 정의를 일관되게 지정합니다. 이전에는 일부 스크립트는 `__VENV_PROMPT__`를 무조건 사용했으며, 다른 스크립트는 설정되었을 때만 (이것이 기본입니다) `__VENV_PROMPT__`를 사용했으며 어떤 것은 대신 `__VENV_NAME__`을 사용했습니다. (Contributed by Brett Cannon in [bpo-37663](#).)

6.38 xml

`xml.etree.ElementTree`를 XML 파일로 직렬화할 때 어트리뷰트 내의 공백 문자가 이제 유지됩니다. EOLN은 더는 “n”으로 정규화되지 않습니다. 이것은 XML 명세의 2.11 섹션을 해석하는 방법에 대한 논의 결과입니다. (Contributed by Mefistotelis in [bpo-39011](#).)

7 최적화

- 컴프리헨션에서 임시 변수를 대입하는 관용구를 최적화했습니다. 이제 컴프리헨션에서 `for y in [expr]`는 단순 대입 `y = expr`만큼 빠릅니다. 예를 들면:

```
sums = [s for s in [0] for x in data for s in [s + x]]
```

`:=` 연산자와 달리 이 관용구는 변수를 외부 스코프로 누출하지 않습니다.

(Contributed by Serhiy Storchaka in [bpo-32856](#).)

- 다중 스레드 응용 프로그램에서 시그널 처리를 최적화했습니다. 메인 스레드와 다른 스레드가 신호를 받으면, 처리할 수 없는 계류 중인 시그널을 확인하기 위해 바이트 코드 평가 루프가 더는 각 바이트 코드 명령어에서 중단되지 않습니다. 메인 인터프리터의 메인 스레드만 시그널을 처리할 수 있습니다.

이전에는, 메인 스레드가 시그널을 처리할 때까지 각 명령에서 바이트 코드 평가 루프가 중단되었습니다. (Contributed by Victor Stinner in [bpo-40010](#).)

- `closefrom()` 을 사용하여 FreeBSD에서 subprocess 모듈을 최적화했습니다. (Contributed by Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak and Victor Stinner in [bpo-38061](#).)
- `PyLong_FromDouble()` 은 이제 long에 맞는 값에 대해 최대 1.87배 더 빠릅니다. (Contributed by Sergey Fedoseev in [bpo-37986](#).)
- 많은 파이썬 내장(`range`, `tuple`, `set`, `frozenset`, `list`, `dict`)은 이제 **PEP 590** 벡터콜 프로토콜을 사용하여 가속됩니다. (Contributed by Dong-hee Na, Mark Shannon, Jeroen Demeyer and Petr Viktorin in [bpo-37207](#).)
- 다른 집합이 기본 집합보다 훨씬 큰 경우에 최적화된 `difference_update()`. (Suggested by Evgeny Kapun with code contributed by Michele Orrù in [bpo-8425](#).)
- 파이썬의 작은 객체 할당자(`obmalloc.c`)는 이제 하나의 빈 아레나를 OS로 반환하지 않고 즉시 재사용할 수 있도록 허용합니다. 이렇게 하면 각 이터레이션마다 아레나가 새로 만들어지고 파괴될 수 있도록 하는 간단한 루프에서의 스레싱(`thrashing`)이 방지됩니다. (Contributed by Tim Peters in [bpo-37257](#).)
- 이제 부동 소수점 연산의 정수 나눗셈의 성능이 개선되었습니다. 또한 이 연산에 대한 `ZeroDivisionError`의 메시지가 갱신됩니다. (Contributed by Dong-hee Na in [bpo-39434](#).)
- UTF-8과 ascii 코덱으로 짧은 ASCII 문자열을 디코딩하는 것이 이제 약 15% 더 빨라졌습니다. (Contributed by Inada Naoki in [bpo-37348](#).)

다음은 파이썬 3.4에서 파이썬 3.9까지의 성능 향상에 대한 요약입니다:

Python version	3.4	3.5	3.6	3.7	3.8	3.9
-----	---	---	---	---	---	---
Variable and attribute read access:						
<code>read_local</code>	7.1	7.1	5.4	5.1	3.9	3.9
<code>read_nonlocal</code>	7.1	8.1	5.8	5.4	4.4	4.5
<code>read_global</code>	15.5	19.0	14.3	13.6	7.6	7.8
<code>read_builtin</code>	21.1	21.6	18.5	19.0	7.5	7.8
<code>read_classvar_from_class</code>	25.6	26.5	20.7	19.5	18.4	17.9
<code>read_classvar_from_instance</code>	22.8	23.5	18.8	17.1	16.4	16.9
<code>read_instancevar</code>	32.4	33.1	28.0	26.3	25.4	25.3
<code>read_instancevar_slots</code>	27.8	31.3	20.8	20.8	20.2	20.5
<code>read_namedtuple</code>	73.8	57.5	45.0	46.8	18.4	18.7
<code>read_boundmethod</code>	37.6	37.9	29.6	26.9	27.7	41.1
Variable and attribute write access:						
<code>write_local</code>	8.7	9.3	5.5	5.3	4.3	4.3
<code>write_nonlocal</code>	10.5	11.1	5.6	5.5	4.7	4.8
<code>write_global</code>	19.7	21.2	18.0	18.0	15.8	16.7
<code>write_classvar</code>	92.9	96.0	104.6	102.1	39.2	39.8
<code>write_instancevar</code>	44.6	45.8	40.0	38.9	35.5	37.4
<code>write_instancevar_slots</code>	35.6	36.1	27.3	26.6	25.7	25.8
Data structure read access:						
<code>read_list</code>	24.2	24.5	20.8	20.8	19.0	19.5
<code>read_deque</code>	24.7	25.5	20.2	20.6	19.8	20.2
<code>read_dict</code>	24.3	25.7	22.3	23.0	21.0	22.4
<code>read_strdict</code>	22.6	24.3	19.5	21.2	18.9	21.5
Data structure write access:						
<code>write_list</code>	27.1	28.5	22.5	21.6	20.0	20.0
<code>write_deque</code>	28.7	30.1	22.7	21.8	23.5	21.7

(다음 페이지에 계속)

(이전 페이지에서 계속)

write_dict	31.4	33.3	29.3	29.2	24.7	25.4
write_strdict	28.4	29.9	27.5	25.2	23.1	24.5
Stack (or queue) operations:						
list_append_pop	93.4	112.7	75.4	74.2	50.8	50.6
deque_append_pop	43.5	57.0	49.4	49.2	42.5	44.2
deque_append_popleft	43.7	57.3	49.7	49.7	42.8	46.4
Timing loop:						
loop_overhead	0.5	0.6	0.4	0.3	0.3	0.3

이 결과는 Tools/scripts/var_access_benchmark.py에 있는 변수 액세스 벤치마크 스크립트에서 생성되었습니다. 벤치마크 스크립트는 타이밍을 나노초로 표시합니다. 벤치마크는 python.org에서 찾을 수 있는 macOS 64비트 빌드를 실행하는 Intel® Core™ i7-4960HQ processor에서 측정되었습니다.

8 폐지

- `distutils.bdist_msi` 명령은 이제 폐지되었습니다, 대신 `bdist_wheel`(휠 패키지)을 사용하십시오. (Contributed by Hugo van Kemenade in [bpo-39586](#).)
- 현재 `math.factorial()`은 음이 아닌 정숫값을 갖는 `float` 인스턴스를 허용합니다 (가령 5.0). 정숫값이 아니거나 음수인 `float`에 대해서는 `ValueError`를 발생시킵니다. 이제 폐지되었습니다. 향후 파이썬 버전에서는 모든 `float`에 대해 `TypeError`를 발생시킬 것입니다. (Contributed by Serhiy Storchaka in [bpo-37315](#).)
- `parser`와 `symbol` 모듈은 폐지되었고 이후 버전의 파이썬에서 제거될 예정입니다. 대부분의 사용 사례에서 사용자는 `ast` 모듈을 사용하여 추상 구문 트리(AST) 생성과 컴파일 단계를 활용할 수 있습니다.
- 공개 C API 함수 `PyParser_SimpleParseStringFlags()`, `PyParser_SimpleParseStringFlagsFilename()`, `PyParser_SimpleParseFileFlags()` 및 `PyNode_Compile()`는 폐지되었고 파이썬 3.10에서 이전 구문 분석기와 함께 제거됩니다.
- 불리언 문맥에서 `NotImplemented`를 사용하는 것은 거의 확실히 잘못된 풍부한 비교 구현의 결과이므로 폐지되었습니다. 향후 버전의 파이썬에서 `TypeError`가 될 것입니다. (Contributed by Josh Rosenberg in [bpo-35712](#).)
- `random` 모듈은 현재 모든 해시 가능한 형을 시드 값으로 허용합니다. 불행히도, 이러한 형 중 일부는 결정론적 해시값을 보장하지 않습니다. 파이썬 3.9 이후, 이 모듈은 시드를 `None`, `int`, `float`, `str`, `bytes` 및 `bytearray`로 제한합니다.
- `mode` 인자를 지정하지 않고 쓰기 위해 `GzipFile` 파일을 여는 것은 폐지되었습니다. 향후 파이썬 버전에서는 이때 기본적으로 항상 읽기 위해 열립니다. 쓰기 위해 열면서 경고가 나오지 않도록 하기 위해서는 `mode` 인자를 지정하십시오. (Contributed by Serhiy Storchaka in [bpo-28286](#).)
- `_tkinter.TkappType`의 `split()` 메서드를 폐지하고, 더 일관되고 예측 가능한 동작을 갖는 `splitlist()` 메서드로 대체합니다. (Contributed by Serhiy Storchaka in [bpo-38371](#).)
- 코루틴 객체를 `asyncio.wait()`에 명시적으로 전달하는 것은 폐지되었고 버전 3.11에서 제거됩니다. (Contributed by Yury Selivanov and Kyle Stanley in [bpo-34790](#).)
- `binhex4`와 `hexbin4` 표준은 이제 폐지되었습니다. `binhex` 모듈과 다음 `binascii` 함수는 이제 폐지되었습니다:

- `b2a_hqx()`, `a2b_hqx()`
- `rlecode_hqx()`, `rledecode_hqx()`

(Contributed by Victor Stinner in [bpo-39353](#).)

- `ast` 클래스 `slice`, `Index` 및 `ExtSlice`는 폐지된 것으로 간주하고 향후 파이썬 버전에서 제거될 예정입니다. `Index(value)` 대신 `value` 자체를 사용해야 합니다. `ExtSlice(slices)` 대신 `Tuple(slices, Load())`를 사용해야 합니다. (Contributed by Serhiy Storchaka in [bpo-34822](#).)
- `ast` 클래스 `Suite`, `Param`, `AugLoad` 및 `AugStore`는 폐지된 것으로 간주하고 향후 파이썬 버전에서 제거될 예정입니다. 이것들은 구문 분석기에 의해 생성되지 않았으며 파이썬 3의 코드 생성기에 의해 받아들여지지 않았습니다. (Contributed by Batuhan Taskaya in [bpo-39639](#) and [bpo-39969](#) and Serhiy Storchaka in [bpo-39988](#).)
- `PyEval_InitThreads()`와 `PyEval_ThreadsInitialized()` 함수는 이제 폐지되었고 파이썬 3.11에서 제거됩니다. `PyEval_InitThreads()`를 호출해도 이제 아무런 효과가 없습니다. GIL은 파이썬 3.7부터 `Py_Initialize()`로 초기화됩니다. (Contributed by Victor Stinner in [bpo-39877](#).)
- `shlex.split()` 함수에 첫 번째 인자로 `None`을 전달하는 것은 폐지되었습니다. (Contributed by Zackery Spytz in [bpo-33262](#).)
- `smtplib.MailmanProxy()`는 이제 외부 모듈 `mailman` 없이는 사용할 수 없어서 폐지되었습니다. (Contributed by Samuel Colvin in [bpo-35800](#).)
- `lib2to3` 모듈은 이제 `PendingDeprecationWarning`을 방출합니다. 파이썬 3.9는 PEG 구문 분석기로 전환되었으며 ([PEP 617](#)을 참조하십시오), 파이썬 3.10에는 `lib2to3`의 LL(1) 구문 분석기로 구문 분석할 수 없는 새로운 언어 문법이 포함될 수 있습니다. `lib2to3` 모듈은 향후 파이썬 버전의 표준 라이브러리에서 제거될 수 있습니다. `LibCST`나 `parso`와 같은 제삼자 대안을 고려하십시오. (Contributed by Carl Meyer in [bpo-40360](#).)
- `random.shuffle()`의 `random` 매개 변수는 폐지되었습니다. (Contributed by Raymond Hettinger in [bpo-40465](#).)

9 제거

- `unittest.mock.__version__`의 잘못된 버전이 제거되었습니다.
- `nntplib.NNTP`: `xpath()`와 `xgtitle()` 메서드가 제거되었습니다. 이 메서드는 파이썬 3.3부터 폐지되었습니다. 일반적으로, 이러한 확장은 지원되지 않거나 NNTP 서버 관리자가 활성화하지 않습니다. `xgtitle()`의 경우, `nntplib.NNTP.descriptions()`나 `nntplib.NNTP.description()`을 대신 사용하십시오. (Contributed by Dong-hee Na in [bpo-39366](#).)
- `array.array`: `tostring()`과 `fromstring()` 메서드가 제거되었습니다. 이것들은 `tobytes()`와 `frombytes()`의 별칭으로, 파이썬 3.2부터 폐지되었습니다. (Contributed by Victor Stinner in [bpo-38916](#).)
- 설명되지 않은 `sys.callstats()` 함수가 제거되었습니다. 파이썬 3.7부터는, 폐지되었고 항상 `None`을 반환했습니다. 파이썬 3.7에서 이미 제거된 특수 빌드 옵션 `CALL_PROFILE`이 필요했습니다. (Contributed by Victor Stinner in [bpo-37414](#).)
- `sys.getcheckinterval()`과 `sys.setcheckinterval()` 함수가 제거되었습니다. 파이썬 3.2부터 폐지되었습니다. 대신 `sys.getswitchinterval()`과 `sys.setswitchinterval()`을 사용하십시오. (Contributed by Victor Stinner in [bpo-37392](#).)
- C 함수 `PyImport_Cleanup()`이 제거되었습니다. 이렇게 설명되어 있었습니다: “모듈 테이블을 비웁니다. 내부 전용.” (Contributed by Victor Stinner in [bpo-36710](#).)
- `_dummy_thread`와 `dummy_threading` 모듈이 제거되었습니다. 이 모듈은 스레드 지원이 필수인 파이썬 3.7부터 폐지되었습니다. (Contributed by Victor Stinner in [bpo-37312](#).)
- `aifc.open()`에 대한 `aifc.openfp()` 별칭, `sunau.open()`에 대한 `sunau.openfp()` 별칭, `wave.open()`에 대한 `wave.openfp()` 별칭이 제거되었습니다. 파이썬 3.7부터 폐지되었습니다. (Contributed by Victor Stinner in [bpo-37320](#).)

- `threading.Thread`의 `isAlive()` 메서드가 제거되었습니다. 파이썬 3.8부터 폐지되었습니다. 대신 `is_alive()` 를 사용하십시오. (Contributed by Dong-hee Na in [bpo-37804](#).)
- `ElementTree` 모듈에서 클래스 `ElementTree`와 `Element`의 메서드 `getchildren()`과 `getiterator()`가 제거되었습니다. 파이썬 3.2에서 폐지되었습니다. `x.getchildren()` 대신 `iter(x)`나 `list(x)`를, `x.getiterator()` 대신 `x.iter()`나 `list(x.iter())`를 사용하십시오. (Contributed by Serhiy Storchaka in [bpo-36543](#).)
- 낡은 `plistlib` API는 제거되었습니다, 파이썬 3.4부터 폐지되었습니다. `load()`, `loads()`, `dump()` 및 `dumps()` 함수를 사용하십시오. 또한, `use_built_in_types` 매개 변수가 제거되었으며, 표준 `bytes` 객체가 항상 대신 사용됩니다. (Contributed by Jon Janzen in [bpo-36409](#).)
- C 함수 `PyGen_NeedsFinalizing`이 제거되었습니다. **PEP 442**를 구현한 후, 설명되지도, 테스트되지도, CPython 내 어디에서건 사용되지도 않았습니다. Joannah Nanjeyye의 패치. (Contributed by Joannah Nanjeyye in [bpo-15088](#))
- 파이썬 3.1부터 폐지된 별칭 `base64.encodestring()`과 `base64.decodestring()`은 제거되었습니다: 대신 `base64.encodebytes()`와 `base64.decodebytes()`를 사용하십시오. (Contributed by Victor Stinner in [bpo-39351](#).)
- `fractions.gcd()` 함수가 제거되었습니다, 파이썬 3.5부터 폐지되었습니다 ([bpo-22486](#)): 대신 `math.gcd()`를 사용하십시오. (Contributed by Victor Stinner in [bpo-39350](#).)
- `bz2.BZ2File`의 `buffering` 매개 변수가 제거되었습니다. 파이썬 3.0부터 무시되었고 사용하면 `DeprecationWarning`을 방출했습니다. 파일을 여는 방법을 제어하려면 열린 파일 객체를 전달하십시오. (Contributed by Victor Stinner in [bpo-39357](#).)
- `json.loads()`의 `encoding` 매개 변수가 제거되었습니다. 파이썬 3.1부터 폐지되었고 무시되었습니다; 파이썬 3.8부터 사용하면 `DeprecationWarning`을 방출했습니다. (Contributed by Inada Naoki in [bpo-39377](#))
- `with (await asyncio.lock):`와 `with (yield from asyncio.lock):` 문은 더는 지원되지 않습니다, 대신 `async with lock`을 사용하십시오. `asyncio.Condition`과 `asyncio.Semaphore`도 마찬가지입니다. (Contributed by Andrew Svetlov in [bpo-34793](#).)
- `sys.getcounts()` 함수, `-X showalloccount` 명령 줄 옵션 및 C 구조체 `PyConfig`의 `show_alloc_count` 필드가 제거되었습니다. 이들은 `COUNT_ALLOCS` 매크로를 정의하는 특별한 파이썬 빌드가 필요했습니다. (Contributed by Victor Stinner in [bpo-39489](#).)
- `typing.NamedTuple` 클래스의 `_field_types` 어트리뷰트가 제거되었습니다. 파이썬 3.8부터 폐지되었습니다. 대신 `__annotations__` 어트리뷰트를 사용하십시오. (Contributed by Serhiy Storchaka in [bpo-40182](#).)
- `symtable.SymbolTable.has_exec()` 메서드가 제거되었습니다. 2006년부터 폐지되었으며, 호출되면 `False`를 반환하기만 합니다. (Contributed by Batuhan Taskaya in [bpo-40208](#))
- `asyncio.Task.current_task()`와 `asyncio.Task.all_tasks()`가 제거되었습니다. 파이썬 3.7부터 폐지되었고 대신 `asyncio.current_task()`와 `asyncio.all_tasks()`를 사용할 수 있습니다. (Contributed by Rémi Lapeyre in [bpo-40967](#))
- `html.parser.HTMLParser` 클래스의 `unescape()` 메서드가 제거되었습니다 (파이썬 3.4부터 폐지되었습니다). 문자 참조를 해당 유니코드 문자로 변환하는 데 `html.unescape()`를 사용해야 합니다.

10 파이썬 3.9로 이식하기

이 절에서는 여러분의 코드 수정을 요구할 수도 있는 이전에 설명한 변경 사항과 다른 버그 수정 사항을 나열합니다.

10.1 파이썬 API의 변경

- `__import__()`와 `importlib.util.resolve_name()`은 이전에 `ValueError`를 발생시키던 곳에서 이제 `ImportError`를 발생시킵니다. 특정 예외 형을 포착하고 파이썬 3.9와 이전 버전을 모두 지원하는 호출자는 `except (ImportError, ValueError):`를 사용하여 둘 다 포착해야 합니다.
- `__VENV_PROMPT__`가 ""로 설정될 때 `venv` 활성화 스크립트는 더는 특별한 경우가 아닙니다.
- `select.epoll.unregister()` 메서드는 더는 `EBADF` 에러를 무시하지 않습니다. (Contributed by Victor Stinner in [bpo-39239](#).)
- `buffering` 매개 변수가 제거되었기 때문에, `bz2.BZ2File`의 `compresslevel` 매개 변수는 키워드 전용이 되었습니다. (Contributed by Victor Stinner in [bpo-39357](#).)
- 서브스크립션이 단순화된 AST. 단순 인덱스는 값으로 표시되고, 확장 슬라이스는 튜플로 표시됩니다. `Index(value)`는 `value` 자체를 반환하고, `ExtSlice(slices)`는 `Tuple(slices, Load())`를 반환합니다. (Contributed by Serhiy Storchaka in [bpo-34822](#).)
- `-E`나 `-I` 명령 줄 옵션을 사용할 때 `importlib` 모듈은 이제 `PYTHONCASEOK` 환경 변수를 무시합니다.
- `encoding` 매개 변수가 클래스 `ftplib.FTP`와 `ftplib.FTP_TLS`에 키워드 전용 매개 변수로 추가되었으며, 기본 인코딩이 [RFC 2640](#)을 따르도록 `Latin-1`에서 `UTF-8`로 변경되었습니다.
- `asyncio.loop.shutdown_default_executor()`가 `AbstractEventLoop`에 추가되어서, 이것을 상속하는 대체 이벤트 루프에 이 메서드가 정의되어 있어야 합니다. (Contributed by Kyle Stanley in [bpo-34037](#).)
- 컴파일러 플래그와의 충돌을 방지하기 위해 `__future__` 모듈에서 퓨처 플래그의 상숫값이 갱신되었습니다. 이전에는 `PyCF_ALLOW_TOP_LEVEL_AWAIT`가 `CO_FUTURE_DIVISION`과 충돌했습니다. (Contributed by Batuhan Taskaya in [bpo-39562](#).)
- `array('u')`는 이제 `Py_UNICODE` 대신 `wchar_t`를 C형으로 사용합니다. `Py_UNICODE`는 파이썬 3.3부터 `wchar_t`의 별칭이라서 이 변경은 동작에 영향을 미치지 않습니다. (Contributed by Inada Naoki in [bpo-34538](#).)
- `logging.getLogger()` API는 이제 이름 'root'를 전달할 때 루트 로거를 반환합니다. 이전에는 'root'라는 이름의 비 루트 로거를 반환했습니다. 이것은 사용자 코드가 'root'라는 이름의 루트가 아닌 로거를 명시적으로 원하거나, 'root.py'라는 최상위 모듈에서 `logging.getLogger(__name__)`을 사용하여 로거를 인스턴스화하는 경우에 영향을 미칠 수 있습니다. (Contributed by Vinay Sajip in [bpo-37742](#).)
- `PurePath`의 나눗셈 처리는 이제 `str`이나 `PurePath`의 인스턴스가 아닌 다른 것을 전달하면 `TypeError`를 발생시키는 대신 `NotImplemented`를 반환합니다. 이는 언급한 형을 상속하지 않는 호환 가능한 클래스를 만들 수 있도록 합니다. (Contributed by Roger Aiudi in [bpo-34775](#).)
- Starting with Python 3.9.5 the `ipaddress` module no longer accepts any leading zeros in IPv4 address strings. Leading zeros are ambiguous and interpreted as octal notation by some libraries. For example the legacy function `socket.inet_aton()` treats leading zeros as octal notation. glibc implementation of modern `inet_pton()` does not accept any leading zeros. (Contributed by Christian Heimes in [bpo-36384](#).)
- `codecs.lookup()` now normalizes the encoding name the same way as `encodings.normalize_encoding()`, except that `codecs.lookup()` also converts the name to lower case. For example, "latex+latin1" encoding name is now normalized to "latex_latin1". (Contributed by Jordon Xu in [bpo-37751](#).)

10.2 C API의 변경

- 힙 할당 형(가령 `PyType_FromSpec()` 및 유사한 API로 만들어진 것)의 인스턴스는 파이썬 3.8부터 형 객체에 대한 참조를 보유합니다. 파이썬 3.8의 “C API의 변경”에 표시된 대로, 대부분의 경우, 부작용이 없어야 하지만 사용자 정의 `tp_traverse` 함수가 있는 형의 경우 힙 할당형의 모든 사용자 정의 `tp_traverse` 함수가 객체의 형을 방문하도록 합니다.

예:

```
int
foo_traverse(foo_struct *self, visitproc visit, void *arg) {
    // Rest of the traverse function
    #if PY_VERSION_HEX >= 0x03090000
        // This was not needed before Python 3.9 (Python issue 35810 and
        ↪ 40217)
        Py_VISIT(Py_TYPE(self));
    #endif
}
```

순회 함수가 베이스 클래스(또는 다른 형)의 `tp_traverse`에 위임하면, `Py_TYPE(self)`를 한 번만 방문해야 합니다. `tp_traverse`에서 힙 형만 형을 방문하도록 기대된다는 것에 유의하십시오.

예를 들어, `tp_traverse` 함수가 다음을 포함하면:

```
base->tp_traverse(self, visit, arg)
```

다음을 추가하십시오:

```
#if PY_VERSION_HEX >= 0x03090000
    // This was not needed before Python 3.9 (Python issue 35810 and
    ↪ 40217)
    if (base->tp_flags & Py_TPFLAGS_HEAPTYPE) {
        // a heap type's tp_traverse already visited Py_TYPE(self)
    } else {
        Py_VISIT(Py_TYPE(self));
    }
#else
```

(자세한 내용은 [bpo-35810](#)과 [bpo-40217](#)을 참조하십시오.)

- `PyEval_CallObject`, `PyEval_CallFunction`, `PyEval_CallMethod` 및 `PyEval_CallObjectWithKeywords` 함수는 폐지되었습니다. 대신 `PyObject_Call()`과 그 변형을 사용하십시오. (자세한 내용은 [bpo-29548](#)을 참조하십시오.)

10.3 CPython 바이트 코드 변경

- `assert` 문을 처리하기 위해 `LOAD_ASSERTION_ERROR` 오프코드가 추가되었습니다. 이전에는, `AssertionError` 예외를 가리면 (shadow) `assert` 문이 올바르게 작동하지 않았습니다. (Contributed by Zackery Spytz in [bpo-34880](#).)
- `COMPARE_OP` 오프코드는 4개의 개별 명령어로 분할되었습니다:
 - 풍부한 비교를 위한 `COMPARE_OP`
 - ‘is’와 ‘is not’ 테스트를 위한 `IS_OP`
 - ‘in’과 ‘not in’ 테스트를 위한 `CONTAINS_OP`
 - ‘try-except’ 문에서 예외를 확인하기 위한 `JUMP_IF_NOT_EXC_MATCH`.

(Contributed by Mark Shannon in [bpo-39156](#).)

11 빌드 변경

- `configure` 스크립트에 `--with-platlibdir` 옵션을 추가했습니다: 새 `sys.platlibdir` 어트리뷰트에 저장된, 플랫폼별 라이브러리 디렉터리의 이름. 자세한 정보는 `sys.platlibdir` 어트리뷰트를 참조하십시오. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in [bpo-1294959](#).)
- `COUNT_ALLOCS` 특수 빌드 매크로가 제거되었습니다. (Contributed by Victor Stinner in [bpo-39489](#).)
- 윈도우 이외의 플랫폼에서, `setenv()` 와 `unsetenv()` 함수는 이제 파이썬을 빌드하는 데 필요합니다. (Contributed by Victor Stinner in [bpo-39395](#).)
- 윈도우가 아닌 플랫폼에서, `bdist_wininst` 설치 프로그램 생성은 이제 공식적으로 지원되지 않습니다. (자세한 내용은 [bpo-10945](#)를 참조하십시오.)
- macOS에서 소스로부터 파이썬을 빌드할 때, `_tkinter`는, 이전 macOS 릴리스의 경우와 마찬가지로, `/Library/Frameworks`에 설치되었으면 비 시스템 Tcl과 Tk 프레임 워크와 링크됩니다. macOS SDK가 명시적으로 구성되면 (`--enable-universalsdk=나 -isysroot`를 사용하여), SDK 자체만 검색됩니다. 기본 동작은 여전히 `--with-tcltk-includes`와 `--with-tcltk-libs`로 재정의할 수 있습니다. (Contributed by Ned Deily in [bpo-34956](#).)
- 이제 윈도우 10 ARM64 용으로 파이썬을 빌드할 수 있습니다. (Contributed by Steve Dower in [bpo-33125](#).)
- `--pgo`를 사용할 때 이제 일부 개별 테스트를 건너뜁니다. 문제의 테스트는 PGO 작업 시간을 매우 증가시켰으며 최종 실행 파일의 최적화를 개선하는 데 도움이 되지 않았을 가능성이 높습니다. 이렇게 하면 작업 속도가 약 15배 빨라집니다. 전체 단위 테스트 스위트를 실행하는 것은 느립니다. 이 변경으로 인해 많은 코드 분기가 실행되지 않아서 빌드가 약간 덜 최적화 될 수 있습니다. 훨씬 느린 빌드를 기다릴 의사가 있다면, `./configure [...] PROFILE_TASK="-m test --pgo-extended"`를 사용하여 이전 동작을 복원할 수 있습니다. 우리는 어떤 PGO 작업 집합이 더 빠른 빌드를 생성하는지에 대해 보장하지 않습니다. 결과가 환경, 워크로드 및 컴파일러 도구 체인에 따라 달라질 수 있어서, 이를 중요하게 생각하는 사용자는 자신의 관련 벤치마크를 실행해야 합니다. (자세한 내용은 [bpo-36044](#)와 [bpo-37707](#)를 참조하십시오.)

12 C API 변경

12.1 새로운 기능

- **PEP 573:** 모듈을 클래스와 연결하는 `PyType_FromModuleAndSpec()` 를 추가했습니다; 모듈과 해당 상태를 가져오는 `PyType_GetModule()` 과 `PyType_GetModuleState()` 를 추가합니다; 메서드가 자신이 정의된 클래스에 액세스할 수 있도록 하는 `PyCMethod`와 `METH_METHOD`를 추가합니다. (Contributed by Marcel Plch and Petr Viktorin in [bpo-38787](#).)
- `PyFrame_GetCode()` 함수를 추가했습니다: 프레임 코드를 얻습니다. `PyFrame_GetBack()` 함수를 추가했습니다: 프레임 다음 외부 프레임을 얻습니다. (Contributed by Victor Stinner in [bpo-40421](#).)
- 제한된 C API에 `PyFrame_GetLineNumber()` 를 추가했습니다. (Contributed by Victor Stinner in [bpo-40421](#).)
- 인터프리터를 얻는 `PyThreadState_GetInterpreter()` 와 `PyInterpreterState_Get()` 함수를 추가했습니다. 파이썬 스레드 상태의 현재 프레임을 가져오는 `PyThreadState_GetFrame()` 함수를 추가했습니다. `PyThreadState_GetID()` 함수를 추가했습니다: 파이썬 스레드 상태의 고유 식별자를 가져옵니다. (Contributed by Victor Stinner in [bpo-39947](#).)

- 인자 없이 콜러블 파이썬 객체를 호출하는 새로운 공용 `PyObject_CallNoArgs()` 함수를 C API에 추가했습니다. 인자 없이 콜러블 파이썬 객체를 호출하는 가장 효율적인 방법입니다. (Contributed by Victor Stinner in [bpo-37194](#).)
 - 제한된 C API의 변경 사항 (`Py_LIMITED_API` 매크로가 정의된 경우):
 - 제한된 API에 대한 일반 함수로 `Py_EnterRecursiveCall()` 과 `Py_LeaveRecursiveCall()` 을 제공합니다. 이전에는, 이들이 매크로로 정의되었지만, 이러한 매크로는 `PyThreadState.recursion_depth` 필드에 액세스할 수 없는 제한된 C API로는 컴파일되지 않았습니다 (이 구조체는 제한된 C API에서 불투명합니다).
 - `PyObject_INIT()` 와 `PyObject_INIT_VAR()` 는 구현 세부 정보를 숨기도록 일반 “불투명” 함수가 됩니다.
- (Contributed by Victor Stinner in [bpo-38644](#) and [bpo-39542](#).)
- `PyModule_AddType()` 함수가 추가되어 모듈에 형을 추가하는 것을 돕습니다. (Contributed by Dong-hee Na in [bpo-40024](#).)
 - `PyObject_GC_IsTracked()` 와 `PyObject_GC_IsFinalized()` 함수를 공용 API에 추가하여 파이썬 객체가 현재 추적되고 있거나 가비지 수거기에 의해 이미 파이널라이즈되었는지 조회할 수 있습니다. (Contributed by Pablo Galindo Salgado in [bpo-40241](#).)
 - 함수 류 객체의 사용자 친화적인 문자열 표현을 얻는 `_PyObject_FunctionStr()` 을 추가했습니다. (Patch by Jeroen Demeyer in [bpo-37645](#).)
 - 하나의 위치 인자로 객체를 호출하는 `PyObject_CallOneArg()` 를 추가했습니다 (Patch by Jeroen Demeyer in [bpo-37483](#).)

12.2 파이썬 3.9로 이식하기

- `PyInterpreterState.eval_frame` ([PEP 523](#)) 에는 이제 새로운 필수 `tstate` 매개 변수 (`PyThreadState*`) 가 필요합니다. (Contributed by Victor Stinner in [bpo-38500](#).)
- 확장 모듈: 모듈 상태가 요청되었지만, 아직 할당되지 않았으면 `PyModuleDef` 의 `m_traverse`, `m_clear` 및 `m_free` 함수가 더는 호출되지 않습니다. 이것은 모듈이 만들어진 직후, 모듈이 실행되기 직전의 경우입니다 (`Py_mod_exec` 함수). 더 정확하게는, `m_size` 가 0보다 크고 모듈 상태 (`PyModule_GetState()` 가 반환하는) 가 NULL이면 이 함수가 호출되지 않습니다.
모듈 상태가 없는 확장 모듈 (`m_size <= 0`) 은 영향을 받지 않습니다.
- 서브 인터프리터에서 `Py_AddPendingCall()` 이 호출되면, 함수는 이제 메인 인터프리터에서 호출되지 않고 서브 인터프리터에서 호출되도록 예약됩니다. 각 서브 인터프리터는 이제 자체 예약된 호출 목록을 갖습니다. (Contributed by Victor Stinner in [bpo-39984](#).)
- `-E` 옵션이 사용될 때 (`PyConfig.use_environment` 가 0으로 설정될 때) 윈도우 레지스트리가 더는 `sys.path` 를 초기화하는 데 사용되지 않습니다. 이것은 윈도우에서 파이썬을 내장할 때 중요합니다. (Contributed by Zackery Spytz in [bpo-8901](#).)
- 전역 변수 `PyStructSequence_UnnamedField` 는 이제 상수이며 상수 문자열을 참조합니다. (Contributed by Serhiy Storchaka in [bpo-38650](#).)
- `PyGC_Head` 구조체는 이제 불투명합니다. 내부 C API (`pycore_gc.h`) 에서만 정의됩니다. (Contributed by Victor Stinner in [bpo-40241](#).)
- `Py_UNICODE_COPY`, `Py_UNICODE_FILL`, `PyUnicode_WSTR_LENGTH`, `PyUnicode_FromUnicode()`, `PyUnicode_AsUnicode()`, `_PyUnicode_AsUnicode` 및 `PyUnicode_AsUnicodeAndSize()` 는 C에서 폐지된 것으로 표시됩니다. 파이썬 3.3 이후 [PEP 393](#) 에서 폐지되었습니다. (Contributed by Inada Naoki in [bpo-36346](#).)

- `Py_FatalError()` 함수는, `Py_LIMITED_API` 매크로가 정의되지 않는 한, 현재 함수의 이름을 자동으로 로그 하는 매크로로 대체됩니다. (Contributed by Victor Stinner in [bpo-39882](#).)
- 벡터콜 프로토콜은 이제 호출자가 키워드 이름으로 문자열만 전달하도록 요구합니다. (자세한 내용은 [bpo-37540](#)을 참조하십시오.)
- 이제 여러 매크로와 함수의 구현 세부 사항이 숨겨집니다:
 - `PyObject_IS_GC()` 매크로가 함수로 변환되었습니다.
 - `PyObject_NEW()` 매크로는 `PyObject_New()` 매크로의 별칭이 되고, `PyObject_NEW_VAR()` 매크로는 `PyObject_NewVar()` 매크로의 별칭이 됩니다. 더는 `PyObject.tp_basicsize` 멤버에 직접 액세스하지 않습니다.
 - `PyObject_GET_WEAKREFS_LISTPTR()` 매크로가 함수로 변환되었습니다: 매크로는 `PyObject.tp_weaklistoffset` 멤버에 직접 액세스했습니다.
 - `PyObject_CheckBuffer()` 매크로가 함수로 변환되었습니다: 매크로는 `PyObject.tp_as_buffer` 멤버에 직접 액세스했습니다.
 - `PyIndex_Check()` 는 이제 구현 세부 사항을 숨기기 위해 항상 불투명한 함수로 선언됩니다: `PyIndex_Check()` 매크로가 제거되었습니다. 매크로는 `PyObject.tp_as_number` 멤버에 직접 액세스했습니다.

(자세한 내용은 [bpo-40170](#)을 참조하십시오.)

12.3 제거

- 제한된 C API에서 `pyfpe.h`의 `PyFPE_START_PROTECT()` 와 `PyFPE_END_PROTECT()` 매크로를 제외했습니다. (Contributed by Victor Stinner in [bpo-38835](#).)
- `PyObject`의 `tp_print` 슬롯이 제거되었습니다. 파이썬 2.7과 이전 버전에서 파일로 객체를 인쇄하는 데 사용되었습니다. 파이썬 3.0부터는, 무시되고 사용되지 않았습니다. (Contributed by Jeroen Demeyer in [bpo-36974](#).)
- 제한된 C API의 변경 사항(`Py_LIMITED_API` 매크로가 정의된 경우):
 - 제한된 C API에서 다음 함수를 제외했습니다:
 - * `PyThreadState_DeleteCurrent()` (Contributed by Joannah Nanjey in [bpo-37878](#).)
 - * `_Py_CheckRecursionLimit`
 - * `_Py_NewReference()`
 - * `_Py_ForgetReference()`
 - * `_PyTraceMalloc_NewReference()`
 - * `_Py_GetRefTotal()`
 - * 제한된 C API에서 결코 작동하지 않은 휴지통 메커니즘.
 - * `PyTrash_UNWIND_LEVEL`
 - * `Py_TRASHCAN_BEGIN_CONDITION`
 - * `Py_TRASHCAN_BEGIN`
 - * `Py_TRASHCAN_END`
 - * `Py_TRASHCAN_SAFE_BEGIN`
 - * `Py_TRASHCAN_SAFE_END`

- 다음 함수와 정의를 내부 C API로 옮겼습니다:

```
* _PyDebug_PrintTotalRefs ()
* _Py_PrintReferences ()
* _Py_PrintReferenceAddresses ()
* _Py_tracemalloc_config
* _Py_AddToAllObjects () (Py_TRACE_REFS 빌드에만 해당)
```

(Contributed by Victor Stinner in [bpo-38644](#) and [bpo-39542](#).)

- `_PyRuntime.getframe` 혹은 `제거` 했고 `_PyRuntime.getframe`의 별칭인 `_PyThreadState_GetFrame` 매크로를 제거했습니다. 이들은 내부 C API에 의해서만 노출되었습니다. `PyThreadFrameGetter` 형도 제거했습니다. (Contributed by Victor Stinner in [bpo-39946](#).)
- C API에서 다음 함수를 제거했습니다. 모든 자유 목록을 지우려면 `PyGC_Collect()`를 명시적으로 호출하십시오. (Contributed by Inada Naoki and Victor Stinner in [bpo-37340](#), [bpo-38896](#) and [bpo-40428](#).)
 - `PyAsyncGen_ClearFreeLists()`
 - `PyContext_ClearFreeList()`
 - `PyDict_ClearFreeList()`
 - `PyFloat_ClearFreeList()`
 - `PyFrame_ClearFreeList()`
 - `PyList_ClearFreeList()`
 - `PyMethod_ClearFreeList()`와 `PyCFunction_ClearFreeList()`: 연결된 메서드 객체의 자유 목록이 제거되었습니다.
 - `PySet_ClearFreeList()`: 파이썬 3.4에서 집합 자유 목록이 제거되었습니다.
 - `PyTuple_ClearFreeList()`
 - `PyUnicode_ClearFreeList()`: 파이썬 3.3에서 유니코드 자유 목록이 제거되었습니다.
- `_PyUnicode_ClearStaticStrings()` 함수를 제거했습니다. (Contributed by Victor Stinner in [bpo-39465](#).)
- `Py_UNICODE_MATCH`를 제거했습니다. [PEP 393](#)에서 폐지되었고, 파이썬 3.3부터 망가졌습니다. 대신 `PyUnicode_Tailmatch()` 함수를 사용할 수 있습니다. (Contributed by Inada Naoki in [bpo-36346](#).)
- 정의되었지만 구현이 없는 헤더 파일들을 정리했습니다. 제거되는 공용 API 심볼은: `_PyBytes_InsertThousandsGroupingLocale`, `_PyBytes_InsertThousandsGrouping`, `_Py_InitializeFromArgs`, `_Py_InitializeFromWideArgs`, `_PyFloat_Repr`, `_PyFloat_Digits`, `_PyFloat_DigitsInit`, `PyFrame_ExtendStack`, `_PyAlterWrapper_Type`, `PyNullImporter_Type`, `PyCmpWrapper_Type`, `PySortWrapper_Type`, `PyNoArgsFunction`입니다. (Contributed by Pablo Galindo Salgado in [bpo-39372](#).)

13 파이썬 3.9.1의 주목할만한 변경 사항

13.1 typing

`typing.Literal`의 동작은 **PEP 586**를 따르고 PEP에 지정된 정적 형 검사기의 동작과 일치하도록 변경되었습니다.

1. `Literal`은 이제 매개 변수를 중복 제거합니다.
2. `Literal` 객체 간의 동등 비교는 이제 순서에 독립적입니다.
3. `Literal` 비교는 이제 형을 존중합니다. 예를 들어, `Literal[0] == Literal[False]`는 이전에 `True`로 평가되었습니다. 이제 `False`입니다. 이 변경을 지원하기 위해, 내부적으로 사용되는 형 캐시는 이제 형 구분을 지원합니다.
4. `Literal` objects will now raise a `TypeError` exception during equality comparisons if any of their parameters are not hashable. Note that declaring `Literal` with mutable parameters will not throw an error:

```
>>> from typing import Literal
>>> Literal[{0}]
>>> Literal[{0}] == Literal[{False}]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

(Contributed by Yuri Karabas in [bpo-42345](#).)

13.2 macOS 11.0 (Big Sur) 및 Apple Silicon Mac 지원

3.9.1부터, 파이썬은 이제 macOS 11.0 (Big Sur)과 Apple Silicon Mac(ARM64 아키텍처 기반)에서 빌드와 실행을 완전히 지원합니다. 이제 새로운 유니버설 빌드 변형인 `universal2`를 사용하여 하나의 실행 파일 집합에서 ARM64와 Intel 64를 모두 네이티브 하게 지원할 수 있습니다. 바이너리는 이제 현재 버전의 macOS에서 빌드되어 다양한 이전 macOS 버전(10.9까지 테스트되었습니다)에 배포되는 동시에 실행시간에 사용 중인 운영 체제 버전에 따라 조건부로 사용 가능한 일부 새로운 OS 기능과 옵션을 만들 수 있습니다 (“약한 링크(weaklinking)”).

(Contributed by Ronald Oussoren and Lawrence D’Anna in [bpo-41100](#).)

14 파이썬 3.9.2의 주목할만한 변경 사항

14.1 collections.abc

`collections.abc.Callable` 제네릭은 이제 `typing.Callable`이 현재 수행하는 것과 유사하게 형 매개 변수를 평활화합니다. 이것은 `collections.abc.Callable[[int, str], str]`의 `__args__`가 `(int, str, str)`임을 의미합니다; 이전에는 `([int, str], str)`였습니다. 이러한 변경을 허용하기 위해, 이제 `types.GenericAlias`를 서브 클래스링할 수 있으며, `collections.abc.Callable` 형을 서브 스크립팅할 때 서브 클래스가 반환됩니다. `typing.get_args()` 나 `__args__`를 통해 인자에 액세스하는 코드는 이 변경 사항을 고려해야 합니다. 파이썬 3.9.1에서 조용히 전달되었을 수 있는 `collections.abc.Callable`의 유효하지 않은 매개 변수화 형식에 대해 `DeprecationWarning`을 방출할 수 있습니다. 이 `DeprecationWarning`은 파이썬 3.10에서 `TypeError`가 됩니다. (Contributed by Ken Jin in [bpo-42195](#).)

14.2 urllib.parse

이전 파이썬 버전에서는 `urllib.parse.parse_qs()` 와 `urllib.parse.parse_qs1()` 에서 쿼리 매개 변수 구분자로 `;` 과 `&` 를 모두 사용할 수 있었습니다. 보안 문제와 최신 W3C 권장 사항을 준수하기 위해, `&` 가 기본값인 단일 구분자 키만 허용하도록 변경되었습니다. 이 변경 사항은 영향을 받는 함수를 내부적으로 사용하므로 `cgi.parse()` 와 `cgi.parse_multipart()` 에도 영향을 줍니다. 자세한 내용은 해당 설명서를 참조하십시오. (Contributed by Adam Goldschmidt, Senthil Kumaran and Ken Jin in [bpo-42967](#).)

15 파이썬 3.9.3의 주목할만한 변경 사항

보안 수정은 수동 데이터 채널을 설정할 때 원격 서버에서 보낸 IPv4 주소를 신뢰하지 않도록 `ftplib.FTP` 동작을 변경합니다. 대신 ftp 서버 IP 주소를 재사용합니다. 이전 동작이 필요한 특별한 코드의 경우, FTP 인스턴스의 `trust_server_pasv_ipv4_address` 어트리뷰트를 `True`로 설정하십시오. ([bpo-43285](#)를 참조하십시오)

16 Notable changes in Python 3.9.5

16.1 urllib.parse

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 in `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](#))

17 Notable security feature in 3.9.14

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.

18 Notable Changes in 3.9.17

18.1 tarfile

- The extraction methods in `tarfile`, and `shutil.unpack_archive()`, have a new *filter* argument that allows limiting tar features that may be surprising or dangerous, such as creating files outside the destination directory. See `tarfile-extraction-filter` for details. In Python 3.12, use without the *filter* argument will show a `DeprecationWarning`. In Python 3.14, the default will switch to `'data'`. (Contributed by Petr Viktorin in [PEP 706](#).)

19 Notable changes in 3.9.20

19.1 ipaddress

- Fixed `is_global` and `is_private` behavior in `IPv4Address`, `IPv6Address`, `IPv4Network` and `IPv6Network`.

19.2 email

- Headers with embedded newlines are now quoted on output.

The `generator` will now refuse to serialize (write) headers that are improperly folded or delimited, such that they would be parsed as multiple headers or joined with adjacent data. If you need to turn this safety feature off, set `verify_generated_headers`. (Contributed by Bas Bloemsaat and Petr Viktorin in [gh-121650](#).)

- `email.utils.getaddresses()` and `email.utils.parseaddr()` now return `(' ', '')` 2-tuples in more situations where invalid email addresses are encountered, instead of potentially inaccurate values. An optional `strict` parameter was added to these two functions: use `strict=False` to get the old behavior, accepting malformed inputs. `getattr(email.utils, 'supports_strict_parsing', False)` can be used to check if the `strict` parameter is available. (Contributed by Thomas Dwyer and Victor Stinner for [gh-102988](#) to improve the CVE-2023-27043 fix.)

20 Notable changes in 3.9.23

20.1 os.path

- The `strict` parameter was backported to `os.path.realpath()` to allow for `tarfile` to use it for security vulnerability mitigation. In particular, when `strict` is set to `os.path.ALLOW_MISSING`, errors other than `FileNotFoundError` will be re-raised; the resulting path can be missing but it will be free of symlinks. (Contributed by Petr Viktorin for CVE 2025-4517.)

20.2 tarfile

- `data_filter()` now normalizes symbolic link targets in order to avoid path traversal attacks. (Contributed by Petr Viktorin in [gh-127987](#) and CVE 2025-4138.)
- `extractall()` now skips fixing up directory attributes when a directory was removed or replaced by another kind of file. (Contributed by Petr Viktorin in [gh-127987](#) and CVE 2024-12718.)
- `extract()` and `extractall()` now (re-)apply the extraction filter when substituting a link (hard or symbolic) with a copy of another archive member, and when fixing up directory attributes. The former raises a new exception, `LinkFallbackError`. (Contributed by Petr Viktorin for CVE 2025-4330 and CVE 2024-12718.)
- `extract()` and `extractall()` no longer extract rejected members when `errorlevel()` is zero. (Contributed by Matt Prodan and Petr Viktorin in [gh-112887](#) and CVE 2025-4435.)

색인

P

PYTHONCASEOK, 18

R

RFC

RFC 2640, 18

RFC 3986, 25

Y

파이썬 향상 제안

PEP 393, 21, 23

PEP 442, 17

PEP 523, 21

PEP 573, 3, 20

PEP 584, 3, 4

PEP 585, 3, 5

PEP 586, 24

PEP 590, 3, 14

PEP 593, 3, 13

PEP 596, 3

PEP 602, 4

PEP 614, 3, 6

PEP 615, 4, 6

PEP 616, 3, 5

PEP 617, 3, 5, 16

PEP 706, 25

환경 변수

PYTHONCASEOK, 18