
What's New in Python

Version 3.9.20

A. M. Kuchling

septembre 08, 2024

Python Software Foundation
Email : docs@python.org

Table des matières

1	Résumé – Points forts de la publication	3
2	Vous devez vérifier la présence de DeprecationWarning dans votre code	3
3	Nouvelles fonctionnalités	4
3.1	Opérateurs de fusion et de mise à jour de dictionnaires	4
3.2	Nouvelles méthodes sur les chaînes pour retirer des préfixes et des suffixes	4
3.3	Indication des types paramétrables dans les collections natives	4
3.4	Nouvel analyseur syntaxique	5
4	Autres changements au langage	5
5	Nouveaux modules	6
5.1	<i>zoneinfo</i>	6
5.2	<i>graphlib</i>	6
6	Modules améliorés	6
6.1	<i>ast</i>	6
6.2	<i>asyncio</i>	7
6.3	<i>compileall</i>	7
6.4	<i>concurrent.futures</i>	7
6.5	<i>curses</i>	8
6.6	<i>datetime</i>	8
6.7	<i>distutils</i>	8
6.8	<i>fcntl</i>	8
6.9	<i>ftplib</i>	8
6.10	<i>gc</i>	8
6.11	<i>hashlib</i>	8
6.12	<i>http</i>	9
6.13	<i>IDLE</i> et <i>idlelib</i>	9
6.14	<i>imaplib</i>	9
6.15	<i>importlib</i>	9
6.16	<i>inspect</i>	10
6.17	<i>ipaddress</i>	10
6.18	<i>math</i>	10
6.19	<i>multiprocessing</i>	10
6.20	<i>nntplib</i>	10
6.21	<i>os</i>	10

6.22	<code>pathlib</code>	11
6.23	<code>pdb</code>	11
6.24	<code>poplib</code>	11
6.25	<code>pprint</code>	11
6.26	<code>pydoc</code>	11
6.27	<code>random</code>	11
6.28	<code>signal</code>	11
6.29	<code>smtplib</code>	11
6.30	<code>socket</code>	12
6.31	<code>time</code>	12
6.32	<code>sys</code>	12
6.33	<code>tempfile</code>	12
6.34	<code>tracemalloc</code>	12
6.35	<code>typing</code>	12
6.36	<code>unicodedata</code>	13
6.37	<code>venv</code>	13
6.38	<code>xml</code>	13
7	Optimisations	13
8	Obsolescence	14
9	Retraits	15
10	Portage vers Python 3.9	17
10.1	Changements dans l'API Python	17
10.2	Changements dans l'API C	18
10.3	Changements au code intermédiaire CPython	18
11	Changements à la compilation	19
12	Changements à l'API C	19
12.1	Nouvelles fonctionnalités	19
12.2	Portage vers Python 3.9	20
12.3	Retraits	21
13	Changements importants dans Python 3.9.1	22
13.1	<code>typing</code>	22
13.2	Prise en charge de <i>macOS</i> 11.0 (Big Sur) et de Mac sur processeur Apple	22
14	Changements importants dans Python 3.9.2	22
14.1	<code>collections.abc</code>	22
14.2	<code>urllib.parse</code>	23
15	Notable changes in Python 3.9.3	23
16	Notable changes in Python 3.9.5	23
16.1	<code>urllib.parse</code>	23
17	Notable security feature in 3.9.14	23
18	Notable Changes in 3.9.17	23
18.1	<code>tarfile</code>	23
19	Notable changes in 3.9.20	24
19.1	<code>ipaddress</code>	24
19.2	<code>email</code>	24
	Index	25

Version 3.9.20

Date septembre 08, 2024

Rédacteur Łukasz Langa

This article explains the new features in Python 3.9, compared to 3.8. Python 3.9 was released on October 5th, 2020.

Pour plus de détails, voir le changelog.

Voir aussi :

PEP 596 – Calendrier de sortie de Python 3.9

1 Résumé – Points forts de la publication

Nouvelles fonctionnalités de syntaxe :

- **PEP 584**, ajout de l'opérateur d'union à `dict` ;
- **PEP 585**, indications des types paramétrables dans les collections natives ;
- **PEP 614**, restrictions grammaticales assouplies pour les décorateurs.

Nouvelles fonctionnalités natives :

- **PEP 616**, méthodes sur les chaînes pour enlever des préfixes et des suffixes.

Nouvelles fonctionnalités dans la bibliothèque standard :

- **PEP 593**, annotations flexibles pour les fonctions et les variables ;
- ajout de `os.pidfd_open()` qui permet la gestion de processus sans concurrence critique et sans signaux.

Améliorations de l'interpréteur :

- **PEP 573**, accès rapide à l'état des modules à partir des méthodes des types dans les extensions C ;
- **PEP 617**, CPython utilise maintenant un analyseur lexical *PEG* (*Parsing Expression Grammar*) ;
- certains types Python natifs (*range*, *tuple*, *set*, *frozenset*, *list*, *dict*) sont maintenant plus rapides grâce aux appels *vectorcall* (**PEP 590**) ;
- le ramasse-miettes ne bloque pas sur les objets ressuscités ;
- certains modules Python (`_abc`, `audioop`, `_bz2`, `_codecs`, `_contextvars`, `_crypt`, `_functools`, `_json`, `_locale`, `math`, `operator`, `resource`, `time`, `_weakref`) utilisent maintenant l'initialisation multi-phase telle que définie dans PEP 489 ;
- certains modules de la bibliothèque standard (`audioop`, `ast`, `grp`, `_hashlib`, `pwd`, `_posixsubprocess`, `random`, `select`, `struct`, `termios`, `zlib`) utilisent maintenant l'*ABI* (*Application Binary Interface*) stable définie dans PEP 384.

Nouveaux modules de la bibliothèque :

- **PEP 615**, la base de données des fuseaux horaires de l'IANA est maintenant incluse dans le module `zoneinfo` de la bibliothèque standard ;
- une implémentation du tri topologique d'un graphe est maintenant disponible dans le nouveau module `graphlib`.

Changements aux procédures de publication :

- **PEP 602**, CPython adopte un cycle annuel de publications.

2 Vous devez vérifier la présence de `DeprecationWarning` dans votre code

Quand Python 2.7 était encore pris en charge, plusieurs fonctionnalités dans Python 3 étaient gardées pour la rétro-compatibilité avec Python 2.7. Avec la fin de la prise en charge de Python 2, ces couches de rétro-compatibilité ont été retirées ou le seront bientôt. La plupart d'entre elles émettaient un avertissement `DeprecationWarning` depuis plusieurs années. Par exemple, utiliser `collections.Mapping` plutôt que `collections.abc.Mapping` émettait un `DeprecationWarning` depuis Python 3.3, publié en 2012.

Testez votre application avec l'option de ligne de commande `-W default` pour voir les `DeprecationWarning` et les `PendingDeprecationWarning`, ou même avec `-W error` pour les traiter comme des erreurs. Le filtre des avertissements peut être utilisé pour ignorer les avertissements provenant de code de tierces parties.

Python 3.9 est la dernière version qui fournit ces couches de rétro compatibilité avec Python 2, afin de donner plus de temps aux mainteneurs de projets Python pour organiser l'arrêt de la prise en charge de Python 2 et pour assurer celle de Python 3.9.

Les alias vers Abstract Base Classes dans le module `collections`, comme l'alias `collections.Mapping` vers `collections.abc.Mapping`, sont conservés pour une dernière version au titre de la rétrocompatibilité. Ils seront supprimés de Python 3.10.

Plus généralement, essayez d'exécuter vos tests dans le Python Development Mode qui aide à préparer votre code pour le rendre compatible avec la prochaine version de Python.

Note : certaines déclarations d'obsolescence pré-existantes ont été retirées de cette version de Python. Voir la section [Retraits](#).

3 Nouvelles fonctionnalités

3.1 Opérateurs de fusion et de mise à jour de dictionnaires

Ajout des opérateurs de fusion (`|`) et de mise-à-jour (`|=`) à la classe native `dict`. Ils viennent compléter les méthodes existantes `dict.update` et `{**d1, **d2}` pour fusionner des dictionnaires.

Exemple :

```
>>> 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'}
```

Voir [PEP 584](#) pour une description complète (contribution de *Brandt Bucher* dans [bpo-36144](#)).

3.2 Nouvelles méthodes sur les chaînes pour retirer des préfixes et des suffixes

`str.removeprefix` (préfixe) et `str.removesuffix` (suffixe) ont été ajoutées pour supprimer facilement un préfixe ou un suffixe inutile d'une chaîne. Les méthodes correspondantes de `bytes`, `bytearray` et `collections.UserString` ont également été ajoutées. Voir [PEP 616](#) pour une description complète (contribution de Dennis Sweeney dans [bpo-39939](#)).

3.3 Indication des types paramétrables dans les collections natives

Dans les annotations de type, vous pouvez maintenant utiliser les types de collection natifs tels que `list` et `dict` comme types génériques au lieu d'importer les types en majuscules correspondants (par exemple `List` ou `Dict`) à partir de `typing`. D'autres types de la bibliothèque standard sont maintenant génériques, par exemple `queue.Queue`.

Exemple :

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

Voir [PEP 585](#) pour plus de détails (contribution de Guido van Rossum, Ethan Smith et Batuhan Taşkaya dans [bpo-39481](#)).

3.4 Nouvel analyseur syntaxique

Python 3.9 utilise un nouvel analyseur syntaxique, basé sur PEG au lieu de LL(1). Les performances du nouvel analyseur sont à peu près comparables à celles de l'ancien, mais le formalisme PEG est plus souple que LL(1) lorsqu'il s'agit de concevoir de nouvelles fonctionnalités du langage. Nous commencerons à utiliser cette flexibilité en Python 3.10 et supérieur.

Le module `ast` utilise le nouvel analyseur syntaxique et produit le même AST que l'ancien analyseur.

En Python 3.10, l'ancien analyseur sera supprimé ainsi que toutes les fonctionnalités qui en dépendent (principalement le module `parser`, qui est obsolète depuis longtemps). En Python 3.9 *seulement*, vous pouvez revenir à l'analyseur LL(1) en utilisant une option en ligne de commande (`-X oldparser`) ou une variable d'environnement (`PYTHONOLDPARSER=1`).

Voir [PEP 617](#) pour plus de détails (contribution de Guido van Rossum, Pablo Galindo et Lysandros Nikolaou dans [bpo-40334](#)).

4 Autres changements au langage

- `__import__()` lève maintenant `ImportError` plutôt que `ValueError`, ce qui se produisait lorsqu'un import relatif remontait plus loin que le répertoire racine du paquet (contribution de *Ngalim Siregar* dans [bpo-37444](#)).
- Python récupère maintenant le chemin absolu du nom de fichier du script spécifié sur la ligne de commande (ex. : `python3 script.py`) : l'attribut `__file__` du module `__main__` est devenu un chemin absolu, plutôt qu'un chemin relatif. Ces nouveaux chemins restent valides même après que le répertoire courant est changé par `os.chdir()`. En conséquence, les traces d'appels affichent le chemin absolu pour les cadres du module `__main__` dans ce cas (contribution de *Victor Stinner* dans [bpo-20443](#)).
- In the Python Development Mode and in debug build, the *encoding* and *errors* arguments are now checked for string encoding and decoding operations. Examples : `open()`, `str.encode()` and `bytes.decode()`.
Par défaut, afin d'améliorer les performances, l'argument *errors* n'est consulté qu'au moment de la première erreur d'encodage-décodage et l'argument *encoding* est parfois ignoré pour les chaînes vides (contribution de *Victor Stinner* dans [bpo-37388](#)).
- `"".replace("", s, n)` renvoie maintenant `s` plutôt qu'une chaîne vide pour toute valeur non-nulles de `n`. Ce comportement est maintenant cohérent avec `"".replace("", s)`. Les instances de `bytes` et `bytearray` ont le même comportement (contribution de *Serhiy Storchaka* dans [bpo-28029](#)).
- Toute expression valide peut maintenant être utilisée comme décorateur (voir [decorator](#)). Préalablement, la grammaire était beaucoup plus contraignante. Voir [PEP 614](#) pour les détails (contributions de *Brandt Bucher* dans [bpo-39702](#)).
- Aide améliorée pour le module `typing`. La documentation est maintenant visible pour toutes les formes spéciales et les alias générique spéciaux (comme `Union` et `List`). Utiliser `help()` sur un alias générique comme `List[int]` montre l'aide correspondant au type réel (`list` dans ce cas). (Contribution de *Serhiy Storchaka* dans [bpo-40257](#)).
- L'exécution parallèle de `aclose()` / `asend()` / `athrow()` est maintenant interdite, et `ag_running` reflète maintenant le statut réel du générateur asynchrone. (Contribué par *Yury Selivanov* dans [bpo-30773](#).)
- Les erreurs inattendues lors d'un appel de la méthode `__iter__` ne sont plus masquées par `TypeError` dans l'opérateur `in` et les fonctions `contains()`, `indexOf()` et `countOf()` du module `operator`. (Contribué par *Serhiy Storchaka* dans [bpo-40824](#).)
- Les expressions `lambda` sans parenthèses ne peuvent plus être la partie expression dans une clause `if` dans les compréhensions de liste et les expressions de générateur. Voir [bpo-41848](#) et [bpo-43755](#) pour plus de détails.

5 Nouveaux modules

5.1 *zoneinfo*

Le module `zoneinfo` ajoute la prise en charge de la base de donnée des fuseaux horaires de l'IANA dans la bibliothèque standard. Il ajoute `zoneinfo.ZoneInfo`, une implémentation concrète de `datetime.tzinfo` à partir des données de fuseaux horaires du système.

Exemple :

```
>>> 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
```

Comme option de rabat pour les plateformes qui n'incluent pas la base de donnée de l'IANA, le module quasi-natif `tzdata` a été publié -- il est distribué sur PyPI et maintenu par l'équipe principale de CPython.

Voir aussi :

PEP 615 — Prise en charge de la base de donnée des fuseaux horaires de l'IANA dans la bibliothèque standard
PEP écrit et implémenté par *Paul Ganssle*

5.2 *graphlib*

Un nouveau module, `graphlib`, a été ajouté. Il contient la classe `graphlib.TopologicalSorter` qui offre des fonctionnalités pour faire le tri topologique de graphes (contribution de *Pablo Galindo*, *Tim Peters* et *Larry Hastings* dans [bpo-17005](#)).

6 Modules améliorés

6.1 *ast*

Ajout de l'option `indent` à `dump()` qui permet de produire une sortie multi-ligne indentée (contribution de *Serhiy Storchaka* dans [bpo-37995](#)).

Ajout de la fonction `ast.unparse()` au module `ast` pour renverser l'analyse syntaxique d'un objet `ast.AST` et produire une chaîne de code qui produirait un `ast.AST` équivalent lorsqu'il est analysé (contribution de *Pablo Galindo* et *Batuhan Taskaya* dans [bpo-38870](#)).

Ajout de chaînes de documentation aux nœuds AST qui contiennent les signatures *ASDL* utilisées pour construire ce nœud (contribution de *Batuhan Taskaya* dans [bpo-39638](#)).

6.2 asyncio

En raison de problèmes de sécurité importants, le paramètre `reuse_address` de `asyncio.loop.create_datagram_endpoint()` n'est plus supporté. C'est à cause du comportement de l'option socket `SO_REUSEADDR` en UDP. Pour plus de détails, voir la documentation de `loop.create_datagram_endpoint()`. (Contribué par Kyle Stanley, Antoine Pitrou, et Yury Selivanov dans [bpo-37228](#).)

Ajout d'une nouvelle coroutine `shutdown_default_executor()` qui programme une extinction de l'exécuteur par défaut qui attend que `ThreadPoolExecutor` se termine. Aussi, `asyncio.run()` à été mis à jour pour utiliser la nouvelle coroutine. (Contribué par Kyle Stanley dans [bpo-34037](#).)

Ajout de `asyncio.PidfdChildWatcher`, un surveillant de processus enfants spécifique à Linux qui interroge le descripteur de fichier du processus. ([bpo-38692](#))

Added a new coroutine `asyncio.to_thread()`. It is mainly used for running IO-bound functions in a separate thread to avoid blocking the event loop, and essentially works as a high-level version of `run_in_executor()` that can directly take keyword arguments. (Contributed by Kyle Stanley and Yury Selivanov in [bpo-32309](#).)

When cancelling the task due to a timeout, `asyncio.wait_for()` will now wait until the cancellation is complete also in the case when `timeout` is ≤ 0 , like it does with positive timeouts. (Contributed by Elvis Pranskevichus in [bpo-32751](#).)

`asyncio` lève maintenant une `TypeError` lors d'un appel à des méthodes incompatibles avec un socket `ssl.SSLSocket`. (Contribué par Ido Michael dans [bpo-37404](#).)

6.3 compileall

Ajout de la possibilité d'utiliser des liens physiques pour les fichiers `.pyc` dupliqués : paramètre `hardlink_dupes` et option de ligne de commande `--hardlink-dupes` (contribution de Lumír 'Frenzy' Balhar dans [bpo-40495](#)).

Ajout d'options pour manipuler les chemins des fichiers `.pyc` résultants : paramètres `stripdir`, `prependdir`, `limit_sl_dest` et options de la ligne de commande `-s`, `-p`, `-e`. Ajout de la possibilité de spécifier l'option pour un niveau d'optimisation plus d'une fois (contribution de Lumír 'Frenzy' Balhar dans [bpo-38112](#)).

6.4 concurrent.futures

Ajout du nouveau paramètre `cancel_futures` à `concurrent.futures.Executor.shutdown()` qui annule tous les futurs en attente qui n'ont pas commencé leur exécution, plutôt que d'attendre qu'ils aient complété avant d'arrêter l'exécuteur (contribution Kyle Stanley dans [bpo-39349](#)).

Removed daemon threads from `ThreadPoolExecutor` and `ProcessPoolExecutor`. This improves compatibility with subinterpreters and predictability in their shutdown processes. (Contributed by Kyle Stanley in [bpo-39812](#).)

Workers in `ProcessPoolExecutor` are now spawned on demand, only when there are no available idle workers to reuse. This optimizes startup overhead and reduces the amount of lost CPU time to idle workers. (Contributed by Kyle Stanley in [bpo-39207](#).)

6.5 curses

Ajout des fonctions `curses.get_escdelay()`, `curses.set_escdelay()`, `curses.get_tabsize()`, et `curses.set_tabsize()` (contribution d'*Anthony Sottile* dans [bpo-38312](#)).

6.6 datetime

La méthode `isocalendar()` de `datetime.date` et la méthode `isocalendar()` de `datetime.datetime` renvoient maintenant un `namedtuple()` plutôt qu'un `tuple` (contribution de *Dong-hee Na* dans [bpo-24416](#)).

6.7 distutils

La commande **upload** crée maintenant des condensats SHA2-256 et Blake2b-256. Elle ne produit pas MD5 sur les plateformes qui bloquent les condensats MD5 (contribution de *Christian Heimes* dans [bpo-40698](#)).

6.8 fcntl

Ajout des constantes `F_OFD_GETLK`, `F_OFD_SETLK` et `F_OFD_SETLKW` (contribution de *Dong-hee Na* dans [bpo-38602](#)).

6.9 ftplib

Afin d'empêcher la création de *socket* non-bloquant, les classes `FTP` et `FTP_TLS` lèvent maintenant `ValueError` quand zéro est passé à *timeout* dans leur constructeur (contribution de *Dong-hee Na* dans [bpo-39259](#)).

6.10 gc

Quand le ramasse-miettes fait un ramassage pour lequel certains objets sont ressuscités (ils sont atteignables à partir de l'extérieur des cycles isolés après que les *finaliseurs* soient exécutés), le ramassage des objets qui sont toujours inatteignables n'est pas suspendu (contribution de *Pablo Galindo* et *Tim Peters* dans [bpo-38379](#)).

Ajout de la fonction `gc.is_finalized()` pour vérifier si un objet a été finalisé par le ramasse-miettes (contribution de *Pablo Galindo* dans [bpo-39322](#)).

6.11 hashlib

Le module `hashlib` peut maintenant utiliser les algorithmes de hachage SHA3 et SHAKE XOF de *OpenSSL* quand ils sont disponibles (contribution de *Christian Heimes* dans [bpo-37630](#)).

Les modules de hachage natifs peuvent être désactivés avec `./configure --without-builtin-hashlib-hashes` ou activés sélectivement avec par exemple `./configure --with-builtin-hashlib-hashes=sha3,blake2` pour forcer l'utilisation de l'implémentation *OpenSSL* (contribution de *Christian Heimes* dans [bpo-40479](#)).

6.12 http

Les codes d'état HTTP 103 `EARLY_HINTS`, 418 `IM_A_TEAPOT` et 425 `TOO_EARLY` ont été ajoutés à `http.HTTPStatus` (contribution de *Dong-hee Na* dans [bpo-39509](#) et Ross Rhodes dans [bpo-39507](#)).

6.13 IDLE et *idlelib*

Ajout d'une option pour désactiver le clignotement du curseur (contribution de *Zackery Spytz* dans [bpo-4603](#)).

La fenêtre de complétion de IDLE est maintenant fermée par la touche d'échappement (contribution de *Johnny Najera* dans [bpo-38944](#)).

Ajout de mots clés à la liste de complétion de noms de modules (contribution de *Terry J. Reedy* dans [bpo-37765](#)).

New in 3.9 maintenance releases

IDLE invoque maintenant `sys.excepthook()` (lorsque lancé sans `-n`). Auparavant, les fonctions de rappel définies par l'utilisateur étaient ignorées (contribution par *Ken Hilton* dans [bpo-43008](#)).

Les changements ci-haut ont été rétro-portés dans la version de maintenance de 3.8.

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*

IMAP4 et IMAP4_SSL ont maintenant un paramètre optionnel *timeout* dans leur constructeur. De plus, la méthode `open()` a maintenant un paramètre optionnel *timeout* avec ce changement. Les méthodes surchargées de IMAP4_SSL et IMAP4_stream appliquent ce changement (contribution de *Dong-hee Na* dans [bpo-38615](#)).

Ajout de `imaplib.IMAP4.unselect()`. `imaplib.IMAP4.unselect()` libère les ressources du serveur associées à la boîte de courriel sélectionnée et remet le serveur à l'état *authenticated*. Cette commande exécute les mêmes actions que `imaplib.IMAP4.close()`, mais aucun message n'est retiré de façon permanente de la boîte courriel sélectionnée (contribution de *Dong-hee Na* dans [bpo-40375](#)).

6.15 *importlib*

Pour plus de cohérence avec les instructions d'import, `importlib.util.resolve_name()` lève maintenant `ImportError` plutôt que `ValueError` pour les tentatives d'import relatifs invalides (contribution de *Ngalim Siregar* dans [bpo-37444](#)).

Les chargeurs d'importation qui publient des objets modules immuables peuvent maintenant aussi publier des paquets immuables en plus des modules individuels (contribution de *Dino Viehland* dans [bpo-39336](#)).

Ajout de la fonction `importlib.resources.files()` qui prend en charge les sous-répertoires dans les données du paquet. Aussi rétro-porté pour les version précédentes de Python dans `importlib_resources` version 1.5 (contribution de *Jason R. Coombs* dans [bpo-39791](#)).

Mise à jour de `importlib.metadata` à partir de `importlib_metadata` version 1.6.1.

6.16 inspect

`inspect.Arguments.arguments`, qui était un `OrderedDict`, devient un dict natif régulier (contribution de *Inada Naoki* dans [bpo-36350](#) et [bpo-39775](#)).

6.17 ipaddress

`ipaddress` prend désormais en charge les IPv6 avec une portée spécifiée (adresses IPv6 avec un suffixe `%<scope_id>`).

Les adresses IPv6 avec une portée spécifiée peuvent être analysées avec `ipaddress.IPv6Address`. S'il est présent, l'indice de la zone de la portée est disponible dans l'attribut `scope_id` (contribution de *Oleksandr Pavliuk* dans [bpo-34788](#)).

À partir de Python 3.9.5, le module `ipaddress` n'accepte plus aucun zéro en tête des chaînes de caractères IPv4. (Contribué par Christian Heimes dans [bpo-36384](#)).

6.18 math

Extension de la fonction `math.gcd()` pour gérer les arguments multiples. Préalablement, elle ne gérait que deux arguments (contribution de *Serhiy Storchaka* dans [bpo-39648](#)).

Ajout de `math.lcm()` : renvoie le plus petit commun multiple des arguments spécifiés (contribution de *Mark Dickinson*, *Ananthakrishnan* et *Serhiy Storchaka* dans [bpo-39479](#) et [bpo-39648](#)).

Ajout de `math.nextafter()` : renvoie la prochaine valeur à point flottant après `x` en direction de `y` (contribution de *Victor Stinner* dans [bpo-39288](#)).

Ajout de `math.ulp()` : renvoie la valeur du bit le moins significatif d'un `float` (contribution de *Victor Stinner* dans [bpo-39310](#)).

6.19 multiprocessing

La classe `multiprocessing.SimpleQueue` a une nouvelle méthode `close()` pour fermer la queue de façon explicite (contribution de *Victor Stinner* dans [bpo-30966](#)).

6.20 nntplib

Afin d'empêcher la création d'interfaces de connexion (*sockets*) non-bloquantes, les classes `NNTP` et `NNTP_SSL` lèvent maintenant `ValueError` si la valeur passée à `timeout` dans leur constructeur est zéro (contribution de *Dong-hee Na* dans [bpo-39259](#)).

6.21 os

Ajout de `CLD_KILLED` et `CLD_STOPPED` pour `si_code` (contribution de *Dong-hee Na* dans [bpo-38493](#)).

Sur Linux, exposition de `os.pidfd_open()` ([bpo-38692](#)) et de `os.P_PIDFD` ([bpo-38713](#)) pour la gestion de processus avec des descripteurs de fichiers.

La fonction `os.unsetenv()` est maintenant aussi disponible sur Windows (contribution de *Victor Stinner* dans [bpo-39413](#)).

Les fonctions `os.putenv()` et `os.unsetenv()` sont maintenant toujours disponibles (contribution de *Victor Stinner* dans [bpo-39395](#)).

Ajout de la fonction `os.waitstatus_to_exitcode()` : convertit la valeur de retour de `wait()` en code de sortie (contribution de *Victor Stinner* dans [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

Ajout de `pathlib.Path.readlink()` qui a un comportement similaire à `os.readlink()` (contribution de *Girts Folkmanis* dans [bpo-30618](#)).

6.23 pdb

Sur Windows, `Pdb` gère maintenant `~/ .pdbrc` (contribution de *Tim Hopper* et *Dan Lidral-Porter* dans [bpo-20523](#)).

6.24 poplib

Afin de prévenir la création d'une interface de connexion (*socket*) non-bloquante, `POP3` et `POP3_SSL` lèvent maintenant une `ValueError` si la valeur passée à *timeout* dans leur constructeur est zéro (contribution de *Dong-hee Na* dans [bpo-39259](#)).

6.25 pprint

`pprint` peut maintenant faire l'affichage formaté de `types.SimpleNamespace` (contribution de *Carl Bordum Hansen* dans [bpo-37376](#)).

6.26 pydoc

La chaîne de documentation est maintenant affichée non seulement pour les classes, fonctions, méthodes, etc., mais aussi pour tout objet qui a son propre attribut `__doc__` (contribution de *Serhiy Storchaka* dans [bpo-40257](#)).

6.27 random

Ajout de la méthode `random.Random.randbytes` : génère des octets aléatoires (contribution de *Victor Stinner* dans [bpo-40286](#)).

6.28 signal

Sur Linux, exposition de `signal.pidfd_send_signal()` pour envoyer des signaux à un processus en utilisant un descripteur de fichier plutôt qu'un identifiant de processus (*pid*). ([bpo-38712](#))

6.29 smtplib

Afin d'éviter la création d'une interface de connexion (*socket*) non-bloquante, `SMTP` et `SMTP_SSL` lèvent maintenant une `ValueError` si la valeur de *timeout* passée à leur constructeur est zéro (contribution de *Dong-hee Na* dans [bpo-39259](#)).

Le constructeur de `LMTP` a maintenant un paramètre *timeout* optionnel (contribution de *Dong-hee Na* dans [bpo-39329](#)).

6.30 *socket*

Le module `socket` exporte maintenant la constante `CAN_RAW_JOIN_FILTERS` à partir de Linux 4.1 (contribution de *Stefan Tatschner* et *Zackery Spytz* dans [bpo-25780](#)).

Le module `socket` prend maintenant en charge le protocole `CAN_J1939` sur les plateformes où il est disponible (contribution de *Karl Ding* dans [bpo-40291](#)).

Le module `socket` possède maintenant les fonctions `socket.send_fds()` et `socket.recv_fds()` (contribution de *Joannah Nanjeye*, *Shinya Okano* et *Victor Stinner* dans [bpo-28724](#)).

6.31 *time*

On AIX, `thread_time()` is now implemented with `thread_cputime()` which has nanosecond resolution, rather than `clock_gettime(CLOCK_THREAD_CPUTIME_ID)` which has a resolution of 10 ms. (Contributed by *Batuhan Taskaya* in [bpo-40192](#))

6.32 *sys*

Ajout du nouvel attribut `sys.platlibdir` : répertoire des bibliothèques spécifiques à la plateforme. Il est utilisé pour construire le chemin de la bibliothèque standard et les chemins des modules d'extension installés. Il est égal à `"lib"` sur la plupart des plateformes. Sur *Fedora* et *SuSE*, il est égal à `"lib64"` sur les plateformes 64-bit (contribution de *Jan Matějek*, *Matěj Cepl*, *Charalampos Stratakis* et *Victor Stinner* dans [bpo-1294959](#)).

Préalablement, `sys.stderr` avait un tampon par blocs quand il n'était pas interactif. Maintenant `stderr` a toujours par défaut un tampon par lignes (contribution de *Jendrik Seipp* dans [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*

Ajout de `tracemalloc.reset_peak()` qui affecte la taille d'apogée (la plus grande taille atteinte) du bloc mémoire tracé à la taille courante, pour mesurer la taille d'apogée de parties de code spécifiques (contribution de *Huon Wilson* dans [bpo-40630](#)).

6.35 *typing*

PEP 593 ajoute le type `typing.Annotated` pour décorer les types existants avec des métadonnées spécifiques au contexte et le nouveau paramètre `include_extras` à `typing.get_type_hints()` pour récupérer les métadonnées à l'exécution (contribution de *Till Varoquaux* et *Konstantin Kashin*).

6.36 unicodedata

La base de donnée Unicode a été mise à jour à la version 13.0.0. ([bpo-39926](#)).

6.37 venv

Tous les scripts d'activations fournis par `venv` spécifient maintenant la personnalisation de l'invite de façon cohérente en utilisant toujours la valeur spécifiée dans `__VENV_PROMPT__`. Précédemment certains scripts utilisaient toujours `__VENV_PROMPT__`, d'autres seulement quand la variable était affectée, et un utilisait plutôt `__VENV_NAME__` (contribution de *Brett Cannon* dans [bpo-37663](#)).

6.38 xml

Les caractères d'espace à l'intérieur des attributs sont maintenant préservés lors de la sérialisation d'`xml.etree.ElementTree` en fichier XML. Les fin de lignes ne sont plus normalisées par "n". Ceci est le résultat d'une discussion sur comment interpréter la section 2.11 de la spécification XML (contribution de *Mefistotelis* dans [bpo-39011](#)).

7 Optimisations

- L'affectation d'une variable temporaire dans les compréhensions est optimisée. Maintenant `for y in [expr]` dans les compréhensions est aussi rapide qu'une affectation simple `y = expr`. Par exemple :

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

Contrairement à l'opérateur `:=`, cet idiome ne laisse pas déborder une variable dans la portée externe.

(contribution de *Serhiy Storchaka* dans [bpo-32856](#)).

- Optimisation de la gestion des signaux dans les applications avec plus d'un fil d'exécution (*multithreaded applications*). Si un fil d'exécution autre que le fil principal reçoit un signal, la boucle d'évaluation du code intermédiaire (*bytecode*) n'est plus interrompue à chaque instruction intermédiaire pour vérifier s'il y a des signaux en attente qui ne peuvent être gérés. Seul le fil principal de l'interpréteur principal peut gérer les signaux.

Précédemment, la boucle d'évaluation du code intermédiaire était interrompue à chaque instruction jusqu'à ce que le fil principal gère les signaux (contribution de *Victor Stinner* dans [bpo-40010](#)).

- Optimisation du module `subprocess` sur FreeBSD en utilisant `closefrom()` (contribution de *Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak* et *Victor Stinner* dans [bpo-38061](#)).
- `PyLong_FromDouble()` is now up to 1.87x faster for values that fit into `long`. (Contributed by *Sergey Fedoseev* in [bpo-37986](#).)
- Certaines fonctions et classes natives (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) sont maintenant plus rapides grâce à l'utilisation du protocole de vectorisation de **PEP 590** (contribution de *Dong-hee Na, Mark Shannon, Jeroen Demeyer* et *Petr Viktorin* dans [bpo-37207](#)).
- Optimisation de `difference_update()` pour le cas où l'autre *set* est beaucoup plus grand que celui de base. (Suggestion par *Evgeny Kapun* et codage contribué par *Michele Orrù* dans [bpo-8425](#).)
- L'allocateur de petits objets de Python (`obmalloc.c`) alloue maintenant (au plus) un aréna vide qui reste disponible pour réutilisation immédiate, sans le renvoyer au système d'exploitation. Ceci prévient l'emballement (*thrashing*) dans les boucles simples où un aréna pourrait être créé puis détruit à nouveau à chaque itération (contribution de *Tim Peters* dans [bpo-37257](#)).
- L'opération de division entière d'un *float* est plus performante. De plus, le message de `ZeroDivisionError` pour cette opération a été mis à jour (contribution de *Dong-hee Na* dans [bpo-39434](#)).
- Le décodage de courtes chaînes ASCII avec les codecs *UTF-8* et *ascii* est maintenant 15 % plus rapide (contribution de *Inada Naoki* dans [bpo-37348](#)).

Voici un résumé des améliorations de performance de Python 3.4 à Python 3.9 :

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

Ces résultats ont été obtenus avec le script de test de performance des accès aux variables : `Tools/scripts/var_access_benchmark.py`. Le script de test de performances affiche le chronométrage en nanosecondes. Les tests ont été effectués sur un processeur Intel® Core™ i7-4960HQ roulant la compilation macOS 64-bit disponible sur python.org.

8 Obsolescence

- La commande de *distutils* `bdist_msi` est maintenant obsolète, utilisez plutôt `bdist_wheel` (paquet *wheel*) (contribution de *Hugo van Kemenade* dans [bpo-39586](#)).
- Présentement, `math.factorial()` accepte des instances de `float` qui sont des entiers non-négatifs (comme `5.0`). Elle lève une `ValueError` pour les *floats* non-entiers et négatifs. Ceci est maintenant obsolète. Les versions futures de Python vont lever une `TypeError` pour tous les *floats* (contribution de *Serhiy Storchaka* dans [bpo-37315](#)).
- Les modules `parser` et `symbol` sont obsolètes et seront retirés des versions futures de Python. Pour la majorité des cas d'utilisation, les utilisateurs peuvent tirer profit des stades de génération et de compilation d'un arbre de syntaxe abstraite, en utilisant le module `ast`.

- Les fonctions `PyParser_SimpleParseStringFlags()`, `PyParser_SimpleParseStringFlagsFilename()`, `PyParser_SimpleParseFileFlags()` et `PyNode_Compile()` de l'API C Publique sont obsolètes et seront retirées de Python 3.10 en même temps que l'ancien analyseur.
- L'utilisation de `NotImplemented` dans un contexte booléen est devenu obsolète, puisque c'est presque toujours le résultat d'une implémentation incorrecte des opérateurs de comparaison avancés. Ceci va devenir une `TypeError` dans les versions futures de Python (contribution de *Josh Rosenberg* dans [bpo-35712](#)).
- Le module `random` accepte présentement n'importe quel hachable comme valeur d'ensemencement. Malheureusement, certains de ces types ne garantissent pas une valeur de hachage déterministe. Après Python 3.9, le module va restreindre l'ensemencement à `None`, `int`, `float`, `str`, `bytes` et `bytearray`.
- Ouvrir un fichier `GzipFile` en écriture sans spécifier l'argument `mode` est obsolète. Dans les futures versions de Python, il sera toujours ouvert en lecture par défaut. Spécifiez l'argument `mode` pour l'ouvrir en écriture et supprimer l'avertissement (contribution de *Serhiy Storchaka* dans [bpo-28286](#)).
- La méthode `split()` de `_tkinter.TkappType` est obsolète. La méthode `splitlist()` est plus favorable, car elle a un comportement plus cohérent et prévisible (contribution de *Serhiy Storchaka* dans [bpo-38371](#)).
- Le passage explicite d'objets coroutine à `asyncio.wait()` est devenu obsolète et sera retiré dans la version 3.11. (Contribution de *Yury Selivanov* et *Kyle Stanley* dans [bpo-34790](#).)
- Les standards `binhex4` et `hexbin4` sont maintenant obsolètes. Le module `binhex` et les fonctions suivantes du module `binascii` sont maintenant obsolètes :
 - `b2a_hqx()`, `a2b_hqx()`
 - `rlecode_hqx()`, `rledecode_hqx()`
 (contribution de *Victor Stinner* dans [bpo-39353](#)).
- Les classes `slice`, `Index` et `ExtSlice` du module `ast` sont considérées obsolètes et seront retirées des versions futures de Python. `value` doit être utilisé à la place de `Index(value)`. `Tuple(slices, Load())` doit être utilisé plutôt que `ExtSlice(slices)` (contribution de *Serhiy Storchaka* dans [bpo-34822](#)).
- Les classes `Suite`, `Param`, `AugLoad` et `AugStore` du module `ast` sont considérées obsolètes et seront retirées des versions futures de Python. Elles n'étaient pas produites par l'analyseur et n'étaient pas acceptées par le générateur de code dans Python 3 (contribution de *Batuhan Taskaya* dans [bpo-39639](#) et [bpo-39969](#) et *Serhiy Storchaka* dans [bpo-39988](#)).
- Les fonctions `PyEval_InitThreads()` et `PyEval_ThreadsInitialized()` sont maintenant obsolètes et seront retirées de Python 3.11. Maintenant, l'appel de `PyEval_InitThreads()` ne fait rien du tout. Le GIL est initialisé par `Py_Initialize()` depuis Python 3.7 (contribution de *Victor Stinner* dans [bpo-39877](#)).
- Passer `None` comme premier argument à la fonction `shlex.split()` a été rendu obsolète (contribution de *Zackery Spytz* dans [bpo-33262](#)).
- `smtpd.MailmanProxy()` est maintenant obsolète puisqu'elle n'est pas utilisable sans le module externe `mailman` (contribution de *Samuel Colvin* dans [bpo-35800](#)).
- Le module `lib2to3` émet maintenant un `PendingDeprecationWarning`. Python 3.9 est passé à un analyseur *PEG* (voir [PEP 617](#)), et Python 3.10 peut inclure des changements à la syntaxe qui ne sont pas analysables par l'analyseur *LL(1)* de `lib2to3`. Le module `lib2to3` peut être retiré de la bibliothèque standard dans une version future de Python. Considérez une alternative tierce telle que `LibCST` ou `parso` (contribution de *Carl Meyer* dans [bpo-40360](#)).
- Le paramètre `random` de `random.shuffle()` est maintenant obsolète (contribution de *Raymond Hettinger* dans [bpo-40465](#)).

9 Retraits

- La version erronée dans `unittest.mock.__version__` a été retirée.
- Les méthodes de `nntplib.NNTP` : `xpath()` et `xgtitle()` ont été retirées. Ces méthodes sont obsolètes depuis Python 3.3. En général, ces extensions ne sont pas prises en charge ou ne sont pas activées par les administrateurs de serveurs *NNTP*. Pour `xgtitle()` veuillez plutôt utiliser `nntplib.NNTP.descriptions()` ou `nntplib.NNTP.description()` (contribution de *Dong-hee Na* dans [bpo-39366](#)).
- Les méthodes de `array.array` : `tostring()` et `fromstring()` ont été retirées. Elles étaient des alias de `tobytes()` et `frombytes()`, obsolètes depuis Python 3.2 (contribution de *Victor Stinner* dans [bpo-39366](#)).

- [bpo-38916](#)).
- La fonction non-documentée `sys.callstats()` a été retirée. Depuis Python 3.7, elle était obsolète et retournait toujours `None`. Elle dépendait de l'option de construction `CALL_PROFILE` qui avait déjà été retirée dans Python 3.7 (contribution de *Victor Stinner* dans [bpo-37414](#)).
 - Les fonctions `sys.getcheckinterval()` et `sys.setcheckinterval()` ont été retirées. Elle étaient obsolètes depuis Python 3.2. Utilisez plutôt `sys.getswitchinterval()` et `sys.setswitchinterval()` (contribution de *Victor Stinner* dans [bpo-37392](#)).
 - La fonction `C PyImport_Cleanup()` a été retirée. Elle était documentée comme "Vide la table des modules. Pour usage interne seulement." (contribution de *Victor Stinner* dans [bpo-36710](#)).
 - Les modules `_dummy_thread` et `dummy_threading` ont été retirés. Ces modules étaient obsolètes depuis Python 3.7 qui a besoin de la gestion de fils d'exécution multiples (contribution de *Victor Stinner* dans [bpo-37312](#)).
 - L'alias `aifc.openfp()` de `aifc.open()`, l'alias `sunau.openfp()` de `sunau.open()` et l'alias `wave.openfp()` de `wave.open()` ont été retirés. Ils étaient obsolètes depuis Python 3.7 (contribution de *Victor Stinner* dans [bpo-37320](#)).
 - La méthode `isAlive()` de la classe `threading.Thread` a été retirée. Elle était obsolète depuis Python 3.8. Utilisez plutôt `is_alive()` (contribution de *Dong-hee Na* dans [bpo-37804](#)).
 - Les méthodes `getchildren()` et `getiterator()` des classes `ElementTree` et `Element` dans le module `ElementTree` ont été retirées. Elle avaient été déclarées obsolètes dans Python 3.2. Utilisez `iter(x)` ou `list(x)` plutôt que `x.getchildren()` et `x.iter()` ou `list(x.iter())` plutôt que `x.getiterator()` (contribution de *Serhiy Storchaka* dans [bpo-36543](#)).
 - L'ancienne API `plistlib` a été retirée, elle était obsolète depuis Python 3.4. Utilisez les fonctions `load()`, `loads()`, `dump()` et `dumps()`. De plus, le paramètre `use_builtin_types` a été retiré, les objets `bytes` natifs sont toujours utilisés.
 - La fonction `C PyGen_NeedsFinalizing` a été retirée. Elle n'était pas documentée, testée ou utilisée où que ce soit dans CPython après l'implémentation de [PEP 442](#) (contribution de *Joannah Nanjeyke* dans [bpo-15088](#)).
 - `base64.encodestring()` et `base64.decodestring()`, des alias obsolètes depuis Python 3.1, ont été retirées : utilisez plutôt `base64.encodebytes()` et `base64.decodebytes()` (contribution de *Victor Stinner* dans [bpo-39351](#)).
 - La fonction `fractions.gcd()` a été retirée, elle était obsolète depuis Python 3.5 ([bpo-22486](#)) : utilisez plutôt `math.gcd()` (contribution de *Victor Stinner* dans [bpo-39350](#)).
 - Le paramètre `buffering` de la classe `bz2.BZ2File` a été retiré. Depuis Python 3.0, il était ignoré et son utilisation levait un `DeprecationWarning`. Passez un objet fichier déjà ouvert pour déterminer comment le fichier sera ouvert (contribution de *Victor Stinner* dans [bpo-39357](#)).
 - Le paramètre `encoding` de `json.loads()` a été retiré. À partir de Python 3.1, il était obsolète et ignoré ; son utilisation levait un `DeprecationWarning` depuis Python 3.8 (contribution de *Inada Naoki* dans [bpo-39377](#)).
 - Les instructions `with (await asyncio.lock):` et `with (yield from asyncio.lock):` ne sont plus prises en charge, utilisez plutôt `async with lock`. De même pour `asyncio.Condition` et `asyncio.Semaphore` (contribution de *Andrew Svetlov* dans [bpo-34793](#)).
 - La fonction `sys.getcounts()`, l'option de ligne de commande `-X showalloccount` et le champ `show_alloc_count` de la structure `C PyConfig` ont été retirés. Ils dépendaient d'une compilation spéciale de Python en définissant la macro `COUNT_ALLOCS` (contribution de *Victor Stinner* dans [bpo-39489](#)).
 - L'attribut `_field_types` de la classe `typing.NamedTuple` a été retiré. Il était obsolète depuis Python 3.8. Utilisez plutôt l'attribut `__annotations__` (contribution de *Serhiy Storchaka* dans [bpo-40182](#)).
 - La méthode `symtable.SymbolTable.has_exec()` a été retirée. Elle était obsolète depuis 2006 et ne renvoyait que `False` lorsqu'elle était appelée (contribution de *Batuhan Taskaya* dans [bpo-40208](#)).
 - Les méthodes `asyncio.Task.current_task()` et `asyncio.Task.all_tasks()` ont été retirées. Elles étaient obsolètes depuis Python 3.7. Utilisez plutôt `asyncio.current_task()` et `asyncio.all_tasks()` (contribution de *Rémi Lapeyre* dans [bpo-40967](#)).
 - La méthode `unescape()` de la classe `html.parser.HTMLParser` a été retirée (elle était obsolète depuis Python 3.4). `html.unescape()` doit être utilisée pour convertir les références de caractère en leurs caractères Unicode correspondants.

10 Portage vers Python 3.9

Cette section liste les changements mentionnés préalablement et d'autres améliorations qui peuvent demander des changements à votre code.

10.1 Changements dans l'API Python

- `__import__()` et `importlib.util.resolve_name()` lèvent maintenant `ImportError` où elles levaient précédemment `ValueError`. Les appelants qui attrapent les types spécifiques d'exceptions et qui prennent à la fois en charge Python 3.9 et des versions précédentes doivent attraper les deux types avec `except (ImportError, ValueError):`.
- Il n'y a plus de traitement spécial dans les scripts d'activation de `venv` quand `__VENV_PROMPT__` est mis à "".
- La méthode `select.epoll.unregister()` n'ignore plus l'erreur `EBADF` (contribution de *Victor Stinner* dans [bpo-39239](#)).
- Le paramètre `compresslevel` de `bz2.BZ2File` est devenu exclusivement nommé (*keyword-only*) puisque le paramètre `buffering` a été retiré (contribution de *Victor Stinner* in [bpo-39357](#)).
- Simplification de l'AST pour l'indigage. Les indices simples sont représentés par leur valeur, les tranches étendues sont représentées par des *n*-uplets. `Index(value)` va renvoyer `value` elle-même, `ExtSlice(slices)` va renvoyer `Tuple(slices, Load())` (contribution de *Serhiy Storchaka* dans [bpo-34822](#)).
- Le module `importlib` ignore maintenant la variable d'environnement `PYTHONCASEOK` quand les options de ligne de commande `-E` ou `-I` sont utilisées.
- Le paramètre `encoding` a été ajouté aux classes `ftplib.FTP` et `ftplib.FTP_TLS` en paramètre exclusivement nommé et l'encodage par défaut est passé de *Latin-1* à *UTF-8* en conformité à [RFC 2640](#).
- `asyncio.loop.shutdown_default_executor()` a été ajouté à `AbstractEventLoop`, donc les boucles d'événements alternatives qui héritent de celle-ci devraient définir cette méthode (contribution de *Kyle Stanley* dans [bpo-34037](#)).
- Les valeurs constantes des drapeaux futurs du module `__future__` sont mises à jour pour éviter les collisions avec les drapeaux du compilateur. Précédemment, `PyCF_ALLOW_TOP_LEVEL_AWAIT` était en collision avec `CO_FUTURE_DIVISION` (contribution de *Batuhan Taskaya* dans [bpo-39562](#)).
- `array('u')` utilise maintenant le type `C wchar_t` plutôt que `Py_UNICODE`. Ceci ne change pas son comportement puisque `Py_UNICODE` est un alias de `wchar_t` depuis Python 3.3 (contribution de *Inada Naoki* dans [bpo-34538](#)).
- L'API `logging.getLogger()` renvoie maintenant le journaliseur racine quand on lui passe le nom `'root'`, alors que préalablement il renvoyait un journaliseur non-racine appelé `'root'`. Ceci peut affecter les cas où le code de l'utilisateur veut explicitement un journaliseur non-racine appelé `'root'`, ou instancie un journaliseur avec `logging.getLogger(__name__)` dans un module de niveau principal appelé `'root.py'` (contribution de *Vinay Sajip* dans [bpo-37742](#)).
- La gestion de la division de `PurePath` renvoie maintenant `NotImplemented` plutôt que de lever une `TypeError` lorsqu'on passe une valeur autre qu'une instance de `str` ou `PurePath`. Ceci permet la création de classes compatibles qui n'héritent pas de ces types (contribution de *Roger Aiudi* dans [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 Changements dans l'API C

- Instances of heap-allocated types (such as those created with `PyType_FromSpec()` and similar APIs) hold a reference to their type object since Python 3.8. As indicated in the "Changes in the C API" of Python 3.8, for the vast majority of cases, there should be no side effect but for types that have a custom `tp_traverse` function, ensure that all custom `tp_traverse` functions of heap-allocated types visit the object's type.

Exemple :

```
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
}
```

If your traverse function delegates to `tp_traverse` of its base class (or another type), ensure that `Py_TYPE(self)` is visited only once. Note that only heap types are expected to visit the type in `tp_traverse`.

Par exemple, si votre fonction `tp_traverse` inclut :

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

alors ajoutez :

```
#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
```

(Voir [bpo-35810](#) et [bpo-40217](#) pour plus d'information.)

- Les fonctions `PyEval_CallObject`, `PyEval_CallFunction`, `PyEval_CallMethod` et `PyEval_CallObjectWithKeywords` sont obsolètes. Utilisez plutôt `PyObject_Call()` et ses variantes. (Pour plus de détails, voir [bpo-29548](#).)

10.3 Changements au code intermédiaire CPython

- Le code d'opération `LOAD_ASSERTION_ERROR` a été ajouté pour gérer l'instruction `assert`. Précédemment, l'instruction `assert` ne fonctionnait pas correctement si l'exception `AssertionError` était masquée (contribution de *Zackery Spytz* dans [bpo-34880](#)).
- Le code d'opération `COMPARE_OP` a été séparé en quatre instructions distinctes :
 - `COMPARE_OP` pour les comparaisons avancées ;
 - `IS_OP` pour les tests *is* et *is not* ;
 - `CONTAINS_OP` pour les tests *in* et *not in* ;
 - `JUMP_IF_NOT_EXC_MATCH` pour vérifier les exceptions dans les instructions `try-except`. (contribution de *Mark Shannon* dans [bpo-39156](#)).

11 Changements à la compilation

- Ajout de l'option `--with-platlibdir` au script `configure` : nom de répertoire des bibliothèques spécifique à la plateforme, stocké dans le nouvel attribut `sys.platlibdir`. Voir l'attribut `sys.platlibdir` pour plus d'informations (contribution de *Jan Matějek*, *Matěj Cepl*, *Charalampos Stratakis* et *Victor Stinner* dans [bpo-1294959](#)).
- La macro spéciale de compilation `COUNT_ALLOCS` a été retirée (contribution de *Victor Stinner* dans [bpo-39489](#)).
- Sur les plateformes non-Windows, les fonctions `setenv()` et `unsetenv()` sont maintenant nécessaires à la compilation de Python (contribution de *Victor Stinner* dans [bpo-39395](#)).
- Sur les plateformes non-Windows, la création d'installateurs `bdist_wininst` n'est maintenant officiellement plus pris en charge. (Voir [bpo-10945](#) pour plus de détails.)
- When building Python on macOS from source, `_tkinter` now links with non-system Tcl and Tk frameworks if they are installed in `/Library/Frameworks`, as had been the case on older releases of macOS. If a macOS SDK is explicitly configured, by using `--enable-universalsdk=` or `-isysroot`, only the SDK itself is searched. The default behavior can still be overridden with `--with-tcltk-includes` and `--with-tcltk-libs`. (Contributed by Ned Deily in [bpo-34956](#).)
- Python peut maintenant être compilé pour *Windows 10 ARM64* (contribution de *Steve Dower* dans [bpo-33125](#)).
- Certains tests individuels sont maintenant ignorés lorsque `--pgo` est utilisé. Les tests en question augmentaient la durée de la tâche *PGO* de manière significative et n'optimisaient probablement pas l'exécutable final. Cela accélère la tâche par un facteur d'environ 15 fois. Le fait de lancer la suite de tests est lent. Ce changement va peut être avoir pour conséquence des exécutables compilés un peu moins optimisés car moins de branches de code seront exécutées. Si vous souhaitez attendre avec une compilation plus lente, l'ancienne manière de fonctionner peut être obtenue avec `./configure [...] PROFILE_TASK="-m test --pgo-extended"`. Nous ne garantissons pas quel set de tâches PGO produira une compilation plus rapide. Les utilisateurs qui s'y intéressent devraient exécuter leurs propres *benchmarks*, car les résultats peuvent dépendre de l'environnement, de la charge de travail et de la chaîne d'outils du compilateur. (Voir [bpo-36044](#) et [bpo-37707](#) pour plus de détails.)

12 Changements à l'API C

12.1 Nouvelles fonctionnalités

- **PEP 573** : Ajout de `PyType_FromModuleAndSpec()` pour associer un module à une classe; `PyType_GetModule()` et `PyType_GetModuleState()` pour récupérer le module et son état; et `PyCMethod` et `METH_METHOD` pour permettre à une méthode de récupérer la classe dans laquelle elle a été définie (contribution de *Marcel Plch* et *Petr Viktorin* dans [bpo-38787](#)).
- Ajout de la fonction `PyFrame_GetCode()` : récupérer le code d'un cadre. Ajout de la fonction `PyFrame_GetBack()` : récupérer le prochain cadre englobant (contribution de *Victor Stinner* dans [bpo-40421](#)).
- Ajout de `PyFrame_GetLineNumber()` à l'API C limité (contribution de *Victor Stinner* dans [bpo-40421](#)).
- Ajout des fonctions `PyThreadState_GetInterpreter()` et `PyInterpreterState_Get()` pour récupérer l'interpréteur. Ajout de la fonction `PyThreadState_GetFrame()` pour récupérer le cadre de l'état d'un fil d'exécution Python. Ajout de la fonction `PyThreadState_GetID()` : récupérer l'identifiant unique de l'état d'un fil d'exécution Python (contribution de *Victor Stinner* dans [bpo-39947](#)).
- Ajout de la nouvelle fonction publique `PyObject_CallNoArgs()` à l'API C, qui appelle un objet Python callable sans aucun argument. C'est la façon la plus efficace d'appeler un objet Python callable sans aucun argument (contribution de *Victor Stinner* dans [bpo-37194](#)).
- Changements dans l'API C limité (si la macro `Py_LIMITED_API` est définie) :
 - Fournit `Py_EnterRecursiveCall()` et `Py_LeaveRecursiveCall()` comme fonctions régulières pour l'API limité. Précédemment, elles étaient définies en tant que macros, mais ces macros n'étaient pas compilées avec l'API C limité qui ne peut pas accéder au champ `PyThreadState.recursion_depth` (la structure est opaque à l'API C limité);

- `PyObject_INIT()` et `PyObject_INIT_VAR()` deviennent des fonctions "opaques" régulières pour cacher les détails d'implémentation (contribution de *Victor Stinner* dans [bpo-38644](#) et [bpo-39542](#)).
- La fonction `PyModule_AddType()` est ajoutée pour faciliter l'ajout d'un type à un module (contribution de *Dong-hee Na* dans [bpo-40024](#)).
- Ajout des fonctions `PyObject_GC_IsTracked()` et `PyObject_GC_IsFinalized()` à l'API publique qui déterminent respectivement si les objets Python sont présentement suivis ou ont été finalisés par le ramasse-miettes (contribution de *Pablo Galindo Salgado* dans [bpo-40241](#)).
- Ajout de `_PyObject_FunctionStr()` pour récupérer une chaîne conviviale représentant un objet fonction-compatible. (Rustine par *Jeroen Demeyer* dans [bpo-37645](#).)
- Ajout de `PyObject_CallOneArg()` pour appeler un objet avec un argument positionnel. (Rustine par *Jeroen Demeyer* dans [bpo-37483](#).)

12.2 Portage vers Python 3.9

- `PyInterpreterState.eval_frame` (**PEP 523**) nécessite maintenant un nouveau paramètre obligatoire `tstate` (`PyThreadState*`) (contribution de *Victor Stinner* dans [bpo-38500](#)).
- Modules d'extension : les fonctions `m_traverse`, `m_clear` et `m_free` de `PyModuleDef` ne sont plus appelées si l'état du module est demandé mais qu'il n'a pas encore été alloué. C'est le cas immédiatement après que le module a été créé et avant que le module soit exécuté (fonction `Py_mod_exec`). Plus précisément, ces fonctions ne sont pas appelées si `m_size` est plus grand que 0 et que l'état du module (tel que renvoyé par `PyModule_GetState()`) est `NULL`.

Les modules d'extensions sans état de module (`m_size <= 0`) ne sont pas affectés.

- Si `Py_AddPendingCall()` est appelée dans un sous-interpréteur, la fonction est planifiée pour appel dans le sous-interpréteur, plutôt que dans l'interpréteur principal. Chaque sous interpréteur a maintenant sa propre liste d'appels planifiés (contribution de *Victor Stinner* dans [bpo-39984](#)).
 - Le registre de Windows n'est plus utilisé pour initialiser `sys.path` quand l'option `-E` est utilisée (si `PyConfig.use_environment` est affecté à 0). Ceci est pertinent quand Python est embarqué sur Windows (contribution de *Zackery Spytz* dans [bpo-8901](#)).
 - La variable globale `PyStructSequence_UnnamedField` est maintenant une constante et fait référence à une chaîne constante (contribution de *Serhiy Storchaka* dans [bpo-38650](#)).
 - La structure `PyGC_Head` est maintenant opaque. Elle est définie uniquement dans l'API C interne (`pycore_gc.h`) (contribution de *Victor Stinner* dans [bpo-40241](#)).
 - `Py_UNICODE_COPY`, `Py_UNICODE_FILL`, `PyUnicode_WSTR_LENGTH`, `PyUnicode_FromUnicode()`, `PyUnicode_AsUnicode()`, `_PyUnicode_AsUnicode`, et `PyUnicode_AsUnicodeAndSize()` sont étiquetées comme obsolètes en C. Elles avaient été déclarées obsolètes par **PEP 393** depuis Python 3.3 (contribution de *Inada Naoki* dans [bpo-36346](#)).
 - La fonction `Py_FatalError()` est remplacée par une macro qui ajoute automatiquement une entrée au journal avec le nom de la fonction courante, sauf si la macro `Py_LIMITED_API` est définie (contribution de *Victor Stinner* dans [bpo-39882](#)).
 - Le protocole *vectorcall* requiert maintenant que l'appelant ne passe que des chaînes pour les noms des arguments nommés. (Voir [bpo-37540](#) pour plus d'informations.)
 - Les détails d'implémentation de certaines macros et fonctions sont maintenant cachés :
 - la macro `PyObject_IS_GC()` a été convertie en fonction ;
 - La macro `PyObject_NEW()` devient un alias pour la macro `PyObject_New()`, et la macro `PyObject_NEW_VAR()` devient un alias pour la macro `PyObject_NewVar()`. Elles n'ont plus accès directement au membre `PyTypeObject.tp_basicsize` ;
 - La macro `PyObject_GET_WEAKREFS_LISTPTR()` a été convertie en fonction : la macro accédait directement au membre `PyTypeObject.tp_weaklistoffset` ;
 - La macro `PyObject_CheckBuffer()` a été convertie en fonction : la macro accédait directement au membre `PyTypeObject.tp_as_buffer` ;
 - `PyIndex_Check()` est maintenant toujours déclarée comme une fonction opaque pour cacher ses détails d'implémentation : retrait de la macro `PyIndex_Check()`. La macro accédait directement au membre `PyTypeObject.tp_as_number`.
- (Voir [bpo-40170](#) pour plus de détails.)

12.3 Retraits

- Exclusion des macros `PyFPE_START_PROTECT()` et `PyFPE_END_PROTECT()` de l'API C limité (contribution de *Victor Stinner* dans [bpo-38835](#)).
- Le champ `tp_print` de `PyObject` a été retiré. Il était utilisé pour écrire des objets dans des fichiers dans Python 2.7 et dans les versions précédentes. Depuis Python 3.0, il était ignoré et inutilisé (contribution de *Jeroen Demeyer* dans [bpo-36974](#)).
- Changements dans l'API C limité (si la macro `Py_LIMITED_API` est définie) :
 - Exclusion des fonctions suivantes de l'API C limité :
 - `PyThreadState_DeleteCurrent()` (contribution de *Joannah Nanjeyke* dans [bpo-37878](#));
 - `_Py_CheckRecursionLimit`;
 - `_Py_NewReference()`;
 - `_Py_ForgetReference()`;
 - `_PyTraceMalloc_NewReference()`;
 - `_Py_GetRefTotal()`;
 - Le mécanisme *trashcan* qui n'a jamais fonctionné dans l'API C limité;
 - `PyTrash_UNWIND_LEVEL`;
 - `Py_TRASHCAN_BEGIN_CONDITION`;
 - `Py_TRASHCAN_BEGIN`;
 - `Py_TRASHCAN_END`;
 - `Py_TRASHCAN_SAFE_BEGIN`;
 - `Py_TRASHCAN_SAFE_END`.
 - Migration des fonctions et définitions suivantes vers l'API C interne :
 - `_PyDebug_PrintTotalRefs()`;
 - `_Py_PrintReferences()`;
 - `_Py_PrintReferenceAddresses()`;
 - `_Py_tracemalloc_config`;
 - `_Py_AddToAllObjects()` (spécifique aux compilations `Py_TRACE_REFS`).(contribution de *Victor Stinner* dans [bpo-38644](#) et [bpo-39542](#)).
- Retrait de la fonction de rappel `_PyRuntime.getframe` et de la macro `_PyThreadState_GetFrame` qui était un alias pour `_PyRuntime.getframe`. Elles étaient exposées uniquement dans l'API C interne. Retrait aussi du type `PyThreadFrameGetter` (contribution de *Victor Stinner* dans [bpo-39946](#)).
- Retrait des fonctions suivantes de l'API C. Appelez `PyGC_Collect()` explicitement pour vider toutes les *free lists* (contribution de *Inada Naoki* et *Victor Stinner* dans [bpo-37340](#), [bpo-38896](#) et [bpo-40428](#)) :
 - `PyAsyncGen_ClearFreeLists()`;
 - `PyContext_ClearFreeList()`;
 - `PyDict_ClearFreeList()`;
 - `PyFloat_ClearFreeList()`;
 - `PyFrame_ClearFreeList()`;
 - `PyList_ClearFreeList()`;
 - `PyMethod_ClearFreeList()` et `PyCFunction_ClearFreeList()` : les *free lists* des objets méthode liées ont été retirées;
 - `PySet_ClearFreeList()` : la *free list* des objets *sets* a été retirée dans Python 3.4;
 - `PyTuple_ClearFreeList()`;
 - `PyUnicode_ClearFreeList()` : la *free list* des objets *Unicode* a été retirée dans Python 3.3.
- Retrait de la fonction `_PyUnicode_ClearStaticStrings()` (contribution de *Victor Stinner* dans [bpo-39465](#)).
- Retrait de `Py_UNICODE_MATCH`. Elle était devenue obsolète par **PEP 393**, non-fonctionnelle depuis Python 3.3. La fonction `PyUnicode_Tailmatch()` peut être utilisée à sa place (contribution de *Inada Naoki* dans [bpo-36346](#)).
- Nettoyage dans les fichiers d'en-têtes d'interfaces déclarées pour lesquelles il n'y avait pas d'implémentation. Les symboles retirés de l'API publique sont : `_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` (contribution de *Pablo Galindo Salgado* dans [bpo-39372](#)).

13 Changements importants dans Python 3.9.1

13.1 *typing*

Le comportement de `typing.Literal` a été changé pour être conforme avec [PEP 586](#) et pour avoir le comportement des vérificateurs de types statique spécifiés dans le PEP :

1. `Literal` de-duplique maintenant les paramètres ;
2. Les comparaisons d'égalité entre les objets `Literal` sont maintenant indépendantes de l'ordre ;
3. Les comparaisons des `Literal` respectent maintenant les types. Par exemple, `Literal[0] == Literal[False]` évaluait précédemment à `True`. C'est maintenant `False`. Pour gérer ce changement, le cache interne des types utilisés peut maintenant différencier les types ;
4. Les objets `Literal` lèvent maintenant une exception `TypeError` lors des comparaisons d'égalités si n'importe quel de leurs paramètres n'est pas immuable. Prenez-note que déclarer un `Literal` avec un paramètre muable ne lève pas une erreur :

```
>>> 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'
```

(Contribution de *Yurii Karabas* dans [bpo-42345](#).)

13.2 Prise en charge de *macOS 11.0 (Big Sur)* et de *Mac sur processeur Apple*

Depuis la version 3.9.1, Python est maintenant totalement compatible (compilation et lancement) sur *macOS 11.0 (Big Sur)* et sur les *Macs Apple Silicon* (basés sur l'architecture ARM64). Un nouveau variant de *build*, *universal2*, est maintenant disponible et supporte nativement les plateformes ARM64 et Intel 64 en un seul jeu d'exécutables. Les binaires peuvent aussi être compilés sur les versions actuelles de *macOS* pour ensuite être déployés sur des versions antérieures (testé jusqu'à 10.9), tout en rendant de nouvelles fonctionnalités du système d'exploitation disponibles en fonction de la version du système utilisée à l'exécution ("*weaklinking*").

(contribution de *Ronald Oussoren* et *Lawrence D'Anna* dans [bpo-41100](#)).

14 Changements importants dans Python 3.9.2

14.1 *collections.abc*

`collections.abc.Callable` generic now flattens type parameters, similar to what `typing.Callable` currently does. This means that `collections.abc.Callable[[int, str], str]` will have `__args__` of `(int, str, str)`; previously this was `([int, str], str)`. To allow this change, `types.GenericAlias` can now be subclassed, and a subclass will be returned when subscripting the `collections.abc.Callable` type. Code which accesses the arguments via `typing.get_args()` or `__args__` need to account for this change. A `DeprecationWarning` may be emitted for invalid forms of parameterizing `collections.abc.Callable` which may have passed silently in Python 3.9.1. This `DeprecationWarning` will become a `TypeError` in Python 3.10. (Contributed by Ken Jin in [bpo-42195](#).)

14.2 urllib.parse

Les versions précédentes de Python acceptaient à la fois `;` et `&` comme séparateurs des paramètres de requête dans `urllib.parse.parse_qs()` et `urllib.parse.parse_qsl()`. Pour des raisons de sécurité, et pour être conforme aux nouvelles recommandations du W3C, ceci a été changé pour ne permettre qu'une seule clé de séparation, avec `&` par défaut. Ce changement affecte aussi `cgi.parse()` et `cgi.parse_multipart()` puisqu'elles utilisent les fonctions affectées à l'interne. Pour plus de détails, voir leur documentation respective (contribution de Adam Goldschmidt, Senthil Kumaran et Ken Jin dans [bpo-42967](#)).

15 Notable changes in Python 3.9.3

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](#))

16 Notable changes in Python 3.9.5

16.1 urllib.parse

La présence de caractères de saut de ligne ou de tabulation dans une URL permet certaines formes d'attaques. En conformité avec la spécification du *WHATWG* qui met à jour la [RFC 3986](#), les caractères *ASCII* de saut de ligne `\n`, `\r` et de tabulation `\t` sont retirés des URL par l'analyseur dans `urllib.parse` pour contrer ces attaques. Le jeu de caractères à retirer est contrôlé par une nouvelle variable de module `urllib.parse._UNSAFE_URL_BYTES_TO_REMOVE` (voir [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.)

Index

P

Python Enhancement Proposals

- PEP 393, [20](#), [21](#)
- PEP 442, [16](#)
- PEP 523, [20](#)
- PEP 573, [3](#), [19](#)
- PEP 584, [3](#), [4](#)
- PEP 585, [3](#), [4](#)
- PEP 586, [22](#)
- PEP 590, [3](#), [13](#)
- PEP 593, [3](#), [12](#)
- PEP 596, [3](#)
- PEP 602, [3](#)
- PEP 614, [3](#), [5](#)
- PEP 615, [3](#), [6](#)
- PEP 616, [3](#), [4](#)
- PEP 617, [3](#), [5](#), [15](#)
- PEP 706, [23](#)

PYTHONCASEOK, [17](#)

R

RFC

- RFC 2640, [17](#)
- RFC 3986, [23](#)

V

variable d'environnement

- PYTHONCASEOK, [17](#)