New versions

September 17, 2024

1 Python versions

A new major version of Python has historically been released every 18 months. Starting with Python 3.9, the schedule is changed to every 12 months, with 3.9 due 6 October 2020.

1.0.1 Support

- 1½ year of full support,
- 3½ more years of security fixes

After the release of Python 3.X.0, the 3.X series is maintained for five years:

During the first eighteen months (1½ year) it receives bugfix updates and full releases (sources and installers for Windows and macOS) are made approximately every other month. For the next forty two months (3½ years) it receives security updates and source-only releases are made on an as-needed basis (no fixed cadence). The final source-only release is made five years after 3.X.0.

2 Python 3.6

New syntax features:

- PEP 498, formatted string literals.

[1]:
```
import decimal
name = 'Fred'
print(f'He said his name is {name}.')
width = 10
precision = 4
value = decimal.Decimal('12.34567')
print(f'result: {value:{width}.{precision}}')  # nested fields
```

```
He said his name is Fred.
result:      12.35
```

- PEP 515, underscores in numeric literals.

[2]:
```
a = 1_000_050_000
print(a)
print('{:_}'.format(a))
print(f'{a:_}')
```

```
1000050000
1_000_050_000
1_000_050_000
```

- PEP 526, syntax for variable annotations.

[3]:
```
from typing import List, Dict
primes: List[int] = []

captain: str  # Note: no initial value!

class Starship:
    stats: Dict[str, int] = {}
```

- PEP 525, asynchronous generators.

[4]:
```
async def ticker(delay, to):
    """Yield numbers from 0 to *to* every *delay* seconds."""
    for i in range(to):
        yield i
        await asyncio.sleep(delay)
```

- PEP 530: asynchronous comprehensions.

[4]:
```
async def aiter():
    for i in range(10):
        yield i

result = [i async for i in aiter() if i % 2]
print(result)

result = [await fun() for fun in funcs if await condition()]
```

```
[1, 3, 5, 7, 9]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_13554/3441334263.py in <cell line: 8>()
      6 print(result)
      7
----> 8 result = [await fun() for fun in funcs if await condition()]

/tmp/ipykernel_13554/3441334263.py in <listcomp>(.0)
      6 print(result)
      7
----> 8 result = [await fun() for fun in funcs if await condition()]

NameError: name 'condition' is not defined
```

New library modules:

- secrets: PEP 506 – Adding A Secrets Module To The Standard Library.

CPython improvements:

- better dict performance (more compact, borrowed from PyPy)

Standard library improvements:

- The asyncio module has received new features, significant usability and performance improvements, and a fair amount of bug fixes. Starting with Python 3.6 the asyncio module is no longer provisional and its API is considered stable.
- A new file system path protocol has been implemented to support path-like objects. All standard library functions operating on paths have been updated to work with the new protocol.
- The datetime module has gained support for Local Time Disambiguation.
- The typing module received a number of improvements.
- The tracemalloc module has been significantly reworked and is now used to provide better output for ResourceWarning as well as provide better diagnostics for memory allocation errors. See the PYTHONMALLOC section for more information.

3 Python 3.7

New syntax features:

- PEP 563, postponed evaluation of type annotations.

Backwards incompatible syntax changes:

- async and await are now reserved keywords.

New library modules:

- contextvars: PEP 567 – Context Variables - used with async
- dataclasses: PEP 557 – Data Classes

[ ]:
```
from dataclasses import dataclass

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

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

- importlib.resources

New built-in features:

- PEP 553, the new breakpoint() function.

Python data model improvements:

- PEP 562, customization of access to module attributes.
- PEP 560, core support for typing module and generic types.
- the insertion-order preservation nature of dict objects has been declared to be an official part of the Python language spec.
- PEP 552: Hash-based .pyc Files

Significant improvements in the standard library:

- The asyncio module has received new features, significant usability and performance improvements.
- The time module gained support for functions with nanosecond resolution.

4 Python 3.8

4.0.1 Assignment expressions

There is new syntax *:=* that assigns values to variables as part of a larger expression. It is affectionately known as "the walrus operator" due to its resemblance to the eyes and tusks of a walrus.

In this example, the assignment expression helps avoid calling len() twice:

[ ]:
```
a = [1, 1]
if (n := len(a)) > 10:
    print(f"List is too long ({n} elements, expected <= 10)")
```

A similar benefit arises during regular expression matching where match objects are needed twice, once to test whether a match occurred and another to extract a subgroup:

[ ]:
```
import re
advertisement = '18% discount'
discount = 0.0
if (mo := re.search(r'(\d+)% discount', advertisement)):
    discount = float(mo.group(1)) / 100.0
```

The operator is also useful with while-loops that compute a value to test loop termination and then need that same value again in the body of the loop:

[ ]:
```
# Loop over fixed length blocks
while (block := f.read(256)) != '':
    process(block)
```

Another motivating use case arises in list comprehensions where a value computed in a filtering condition is also needed in the expression body:

[ ]:
```
[clean_name.title() for name in names
                    if (clean_name := normalize('NFC', name)) in allowed_names]
```

Try to limit use of the walrus operator to clean cases that reduce complexity and improve readability.

4.0.2 Positional-only parameters

[ ]:
```
def f(a, b, /, c, d, *, e, f):
    print(a, b, c, d, e, f)

f(10, 20, 30, d=40, e=50, f=60)       # Valid
f(10, b=20, c=30, d=40, e=50, f=60)   # Invalid: b cannot be a keyword argument
f(10, 20, 30, 40, 50, f=60)           # Invalid: e must be a keyword argument
```

4.0.3 f-strings support = for self-documenting expressions and debugging

4.0.4 Improved fstring syntax "="

[17]:
```
from datetime import date
user = 'eric_idle'
member_since = date(1975, 7, 31)
f'{user=} {member_since=}'
```

[17]:
```
"user='eric_idle' member_since=datetime.date(1975, 7, 31)"
```

4.1 Underscores in numeric literals

[7]:
```
2_000_000
```

[7]:
```
2000000
```

[13]:
```
5_00_000  # 5 Lakh
```

[13]:
```
500000
```

[14]:
```
12_00_00_000  # 12 Crore
```

[14]:
```
120000000
```

[9]:
```
2_00_0
```

[9]:
```
2000
```

[10]:
```
2_1_3_95_1_345
```

[10]:
```
213951345
```

4.1.1 asyncio.run() has graduated from the provisional to stable API.

4.1.2 cProfile

The cProfile.Profile class can now be used as a context manager. Profile a block of code by running:

[ ]:
```
import cProfile

with cProfile.Profile() as profiler:
      # code to be profiled
      ...
```

5 Python 3.9

New syntax features:

- PEP 584, union operators added to dict;
- PEP 585, type hinting generics in standard collections;
- PEP 614, relaxed grammar restrictions on decorators.

New built-in features:

- PEP 616, string methods to remove prefixes and suffixes.

New features in the standard library:

- PEP 593, flexible function and variable annotations;
- os.pidfd_open() added that allows process management without races and signals.

Interpreter improvements:

- PEP 573, fast access to module state from methods of C extension types;
- PEP 617, CPython now uses a new parser based on PEG;
- a number of Python builtins (range, tuple, set, frozenset, list, dict) are now sped up using PEP 590 vectorcall;
- garbage collection does not block on resurrected objects;
- a number of Python modules (_abc, audioop, _bz2, _codecs, _contextvars, _crypt, _functools, _json, _locale, math, operator, resource, time, _weakref) now use multiphase initialization as defined by PEP 489;
- a number of standard library modules (audioop, ast, grp, _hashlib, pwd, _posixsubprocess, random, select, struct, termios, zlib) are now using the stable ABI defined by PEP 384.

New library modules:

- PEP 615, the IANA Time Zone Database is now present in the standard library in the zoneinfo module;
- an implementation of a topological sort of a graph is now provided in the new graphlib module.

Release process changes:

- PEP 602, CPython adopts an annual release cycle.

6 Python 3.10

New syntax features:

- PEP 634, Structural Pattern Matching: Specification
- PEP 635, Structural Pattern Matching: Motivation and Rationale
- PEP 636, Structural Pattern Matching: Tutorial
- bpo-12782, Parenthesized context managers are now officially allowed.

New features in the standard library:

- PEP 618, Add Optional Length-Checking To zip.

Interpreter improvements:

- PEP 626, Precise line numbers for debugging and other tools.

New typing features:

- PEP 604, Allow writing union types as X | Y
- PEP 612, Parameter Specification Variables
- PEP 613, Explicit Type Aliases
- PEP 647, User-Defined Type Guards

Important deprecations, removals or restrictions:

- PEP 644, Require OpenSSL 1.1.1 or newer
- PEP 632, Deprecate distutils module.
- PEP 623, Deprecate and prepare for the removal of the wstr member in PyUnicodeObject.
- PEP 624, Remove Py_UNICODE encoder APIs
- PEP 597, Add optional EncodingWarning

7 Python 3.11

Python 3.11 is between 10-60% faster than Python 3.10. On average, we measured a 1.25x speedup on the standard benchmark suite. See Faster CPython for details.

New syntax features:

- PEP 654: Exception Groups and except*

New built-in features:

- PEP 678: Exceptions can be enriched with notes

New standard library modules:

- PEP 680: tomllib — Support for parsing TOML in the Standard Library

Interpreter improvements:

- PEP 657: Fine-grained error locations in tracebacks
- New -P command line option and PYTHONSAFEPATH environment variable to disable automatically prepending potentially unsafe paths to sys.path

New typing features:

- PEP 646: Variadic generics
- PEP 655: Marking individual TypedDict items as required or not-required
- PEP 673: Self type
- PEP 675: Arbitrary literal string type
- PEP 681: Data class transforms

Important deprecations, removals and restrictions:

- PEP 594: Many legacy standard library modules have been deprecated and will be removed in Python 3.13
- PEP 624: Py_UNICODE encoder APIs have been removed
- PEP 670: Macros converted to static inline functions

8 Python 3.12

New grammar features:

- PEP 701: Syntactic formalization of f-strings

Interpreter improvements:

- PEP 684: A Per-Interpreter GIL

New typing features:

- PEP 688: Making the buffer protocol accessible in Python
- PEP 692: Using TypedDict for more precise **kwargs typing
- PEP 695: Type Parameter Syntax
- PEP 698: Override Decorator for Static Typing

Important deprecations, removals or restrictions:

- PEP 623: Remove wstr from Unicode
- PEP 632: Remove the distutils package. See the migration guide for advice on its replacement.

9 Python 3.13

https://docs.python.org/3.13/whatsnew/3.13.html

[ ]: