What's New in Python

Release 3.11.0a0

A. M. Kuchling

September 02, 2021

Python Software Foundation
Email: docs@python.org

Contents

Release 3.11.0a0

Date September 02, 2021

This article explains the new features in Python 3.11, compared to 3.10.

For full details, see the changelog.

Note: Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.11 moves towards release, so it's worth checking back even after reading earlier versions.

---

1 Summary – Release highlights

2 New Features

2.1 Enhanced error locations in tracebacks

When printing tracebacks, the interpreter will now point to the exact expression that caused the error instead of just the line. For example:

```
Traceback (most recent call last):
  File "distance.py", line 11, in <module>
    print(manhattan_distance(p1, p2))
          ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "distance.py", line 6, in manhattan_distance
    return abs(point_1.x - point_2.x) + abs(point_1.y - point_2.y)
                           ^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'x'
```

Previous versions of the interpreter would point to just the line making it ambiguous which object was `None`. These enhanced errors can also be helpful when dealing with deeply nested dictionary objects and multiple function calls,

```
Traceback (most recent call last):
  File "query.py", line 37, in <module>
    magic_arithmetic('foo')
    ^^^^^^^^^^^^^^^^^^^^^^^
  File "query.py", line 18, in magic_arithmetic
    return add_counts(x) / 25
           ^^^^^^^^^^^^^
  File "query.py", line 24, in add_counts
    return 25 + query_user(user1) + query_user(user2)
                ^^^^^^^^^^^^^^^^^
  File "query.py", line 32, in query_user
    return 1 + query_count(db, response['a']['b']['c']['user'], retry=True)
                               ~~~~~~~~~~~~~~~~~~~~^^^^^
TypeError: 'NoneType' object is not subscriptable
```

as well as complex arithmetic expressions:

```
Traceback (most recent call last):
  File "calculation.py", line 54, in <module>
    result = (x / y / z) * (a / b / c)
              ~~~~~~^~~
ZeroDivisionError: division by zero
```

See **PEP 657** for more details. (Contributed by Pablo Galindo, Batuhan Taskaya and Ammar Askar in bpo-43950.)

Note: This feature requires storing column positions in code objects which may result in a small increase of disk usage of compiled Python files or interpreter memory usage. To avoid storing the extra information and/or deactivate printing the extra traceback information, the `-X no_debug_ranges` command line flag or the `PYTHONNODEBUGRANGES` environment variable can be used.

Column information for code objects

The information used by the enhanced traceback feature is made available as a general API that can be used to correlate bytecode instructions with source code. This information can be retrieved using:

- The `codeobject.co_positions()` method in Python.
- The `PyCode_Addr2Location()` function in the C-API.

The `-X no_debug_ranges` option and the environment variable `PYTHONNODEBUGRANGES` can be used to disable this feature.

See **PEP 657** for more details. (Contributed by Pablo Galindo, Batuhan Taskaya and Ammar Askar in bpo-43950.)

3 Other Language Changes

- Asynchronous comprehensions are now allowed inside comprehensions in asynchronous functions. Outer comprehensions implicitly become asynchronous. (Contributed by Serhiy Storchaka in bpo-33346.)
- A `TypeError` is now raised instead of an `AttributeError` in `contextlib.ExitStack.enter_context()` and `contextlib.AsyncExitStack.enter_async_context()` for objects which do not support the context manager or asynchronous context manager protocols correspondingly. (Contributed by Serhiy Storchaka in bpo-44471.)
- A `TypeError` is now raised instead of an `AttributeError` in `with` and `async with` statements for objects which do not support the context manager or asynchronous context manager protocols correspondingly. (Contributed by Serhiy Storchaka in bpo-12022.)

4 Other CPython Implementation Changes

- Special methods `complex.__complex__()` and `bytes.__bytes__()` are implemented to support `typing.SupportsComplex` and `typing.SupportsBytes` protocols. (Contributed by Mark Dickinson and Dong-hee Na in bpo-24234.)

5 New Modules

- None yet.

6 Improved Modules

6.1 fractions

Support **PEP 515**-style initialization of `Fraction` from string. (Contributed by Sergey B Kirpichev in bpo-44258.)

6.2 math

- Add `math.cbrt()`: return the cube root of x. (Contributed by Ajith Ramachandran in bpo-44357.)
- The behaviour of two `math.pow()` corner cases was changed, for consistency with the IEEE 754 specification. The operations `math.pow(0.0, -math.inf)` and `math.pow(-0.0, -math.inf)` now return `inf`. Previously they raised `ValueError`. (Contributed by Mark Dickinson in bpo-44339.)

6.3 os

- On Windows, `os.urandom()` uses `BCryptGenRandom()` instead of `CryptGenRandom()` which is deprecated. (Contributed by Dong-hee Na in bpo-44611.)

6.4 sqlite3

- You can now disable the authorizer by passing `None` to `set_authorizer()`. (Contributed by Erlend E. Aasland in bpo-44491.)
- Collation name `create_collation()` can now contain any Unicode character. Collation names with invalid characters now raise `UnicodeEncodeError` instead of `sqlite3.ProgrammingError`. (Contributed by Erlend E. Aasland in bpo-44688.)
- `sqlite3` exceptions now include the SQLite error code as `sqlite_errorcode` and the SQLite error name as `sqlite_errorname`. (Contributed by Aviv Palivoda, Daniel Shahaf, and Erlend E. Aasland in bpo-16379.)

7 Removed

- `smtpd.MailmanProxy` is now removed as it is unusable without an external module, `mailman`. (Contributed by Dong-hee Na in bpo-35800.)

8 Optimizations

- Compiler now optimizes simple C-style formatting with literal format containing only format codes `%s`, `%r` and `%a` and makes it as fast as corresponding f-string expression. (Contributed by Serhiy Storchaka in bpo-28307.)
- "Zero-cost" exceptions are implemented. The cost of `try` statements is almost eliminated when no exception is raised. (Contributed by Mark Shannon in bpo-40222.)
- Method calls with keywords are now faster due to bytecode changes which avoid creating bound method instances. Previously, this optimization was applied only to method calls with purely positional arguments. (Contributed by Ken Jin and Mark Shannon in bpo-26110, based on ideas implemented in PyPy.)
- `.pdbrc` is now read with `'utf-8'` encoding.

9 CPython bytecode changes

- Added a new `CALL_METHOD_KW` opcode. Calls a method in a similar fashion as `CALL_METHOD`, but also supports keyword arguments. Works in tandem with `LOAD_METHOD`.

10 Build Changes

11 Deprecated

12 Removed

- The `@asyncio.coroutine` decorator enabling legacy generator-based coroutines to be compatible with async/await code. The function has been deprecated since Python 3.8 and the removal was initially scheduled for Python 3.10. Use `async def` instead. (Contributed by Illia Volochii in bpo-43216.)
- `asyncio.coroutines.CoroWrapper` used for wrapping legacy generator-based coroutine objects in the debug mode. (Contributed by Illia Volochii in bpo-43216.)

13 Porting to Python 3.11

This section lists previously described changes and other bugfixes that may require changes to your code.

13.1 Changes in the Python API

- Prohibited passing non-`concurrent.futures.ThreadPoolExecutor` executors to `loop.set_default_executor()` following a deprecation in Python 3.8. (Contributed by Illia Volochii in bpo-43234.)

14 C API Changes

- Add a new `PyType_GetName()` function to get type's short name. (Contributed by Hai Shi in bpo-42035.)
- Add a new `PyType_GetQualName()` function to get type's qualified name. (Contributed by Hai Shi in bpo-42035.)

14.1 New Features

14.2 Porting to Python 3.11

- The old trashcan macros (`Py_TRASHCAN_SAFE_BEGIN`/`Py_TRASHCAN_SAFE_END`) are now deprecated. They should be replaced by the new macros `Py_TRASHCAN_BEGIN` and `Py_TRASHCAN_END`.

  A tp_dealloc function that has the old macros, such as:

```
static void
mytype_dealloc(mytype *p)
{
    PyObject_GC_UnTrack(p);
    Py_TRASHCAN_SAFE_BEGIN(p);
    ...
    Py_TRASHCAN_SAFE_END
}
```

  should migrate to the new macros as follows:

```
static void
mytype_dealloc(mytype *p)
{
    PyObject_GC_UnTrack(p);
    Py_TRASHCAN_BEGIN(p, mytype_dealloc)
    ...
    Py_TRASHCAN_END
}
```

  Note that `Py_TRASHCAN_BEGIN` has a second argument which should be the deallocation function it is in.

  To support older Python versions in the same codebase, you can define the following macros and use them throughout the code (credit: these were copied from the `mypy` codebase):

```
#if PY_MAJOR_VERSION >= 3 && PY_MINOR_VERSION >= 8
#  define CPy_TRASHCAN_BEGIN(op, dealloc) Py_TRASHCAN_BEGIN(op, dealloc)
#  define CPy_TRASHCAN_END(op) Py_TRASHCAN_END
#else
#  define CPy_TRASHCAN_BEGIN(op, dealloc) Py_TRASHCAN_SAFE_BEGIN(op)
#  define CPy_TRASHCAN_END(op) Py_TRASHCAN_SAFE_END(op)
#endif
```

- The `PyType_Ready()` function now raises an error if a type is defined with the `Py_TPFLAGS_HAVE_GC` flag set but has no traverse function (`PyTypeObject.tp_traverse`). (Contributed by Victor Stinner in bpo-44263.)
- Heap types with the `Py_TPFLAGS_IMMUTABLETYPE` flag can now inherit the **PEP 590** vectorcall protocol. Previously, this was only possible for static types. (Contributed by Erlend E. Aasland in bpo-43908)

14.3 Deprecated

14.4 Removed

- `PyFrame_BlockSetup()` and `PyFrame_BlockPop()` have been removed. (Contributed by Mark Shannon in bpo-40222.)
- Deprecate the following functions to configure the Python initialization:
  - `PySys_AddWarnOptionUnicode()`
  - `PySys_AddWarnOption()`
  - `PySys_AddXOption()`
  - `PySys_HasWarnOptions()`
  - `Py_SetPath()`
  - `Py_SetProgramName()`
  - `Py_SetPythonHome()`
  - `Py_SetStandardStreamEncoding()`
  - `_Py_SetProgramFullPath()`

  Use the new `PyConfig` API of the Python Initialization Configuration instead (**PEP 587**). (Contributed by Victor Stinner in bpo-44113.)
- The following deprecated functions and methods are removed in the `gettext` module: `lgettext()`, `ldgettext()`, `lngettext()` and `ldngettext()`.

  Function `bind_textdomain_codeset()`, methods `output_charset()` and `set_output_charset()`, and the *codeset* parameter of functions `translation()` and `install()` are also removed, since they are only used for the `l*gettext()` functions. (Contributed by Dong-hee Na and Serhiy Storchaka in bpo-44235.)
- The behavior of returning a value from a `TestCase` and `IsolatedAsyncioTestCase` test methods (other than the default `None` value), is now deprecated.

Index

E

P