What's New in Python 3.6

APUG MeetUp, 2017 January 11

Tipton Cole

Official Docs

https://docs.python.org/3.6/whatsnew/3.6.html

Brett Cannon (PyCon Canada 28 minutes) [also at PyBay]

https://www.youtube.com/watch?v=hk85RUtQsBI

Dan Bader (9 minutes)

https://www.youtube.com/watch?v=klKdMxjDaa0

Python Bits (17 minutes)

https://www.youtube.com/watch?v=-P_XUzSTjh8

Raymond Hettinger (68 minutes)

https://www.youtube.com/watch?v=p33CVV29OG8

PEPs

498, 515, 526, 525, 530

Library Module

secrets.py

Python improvements

dict [ordered - not yet reliably]

ordered **kwargs and class attributes

Standard Library

asyncio

pathlib

typing

Security

secrets, os.urandom(), hashlib() and ssl()

Windows

PEPs 528 and 529 -> UTF-8 filenames

PEPs

515: Underscores in numeric literals

```
one_billion = 1_000_000_000
color = 0x_FF_FF_FF_FF
```

498: Formatted literal strings

```
one_billion = 1_000_000_000

print(f'one billion = {one_billion:,}’)

one billion = 1,000,000,000
```

PEP

526: Syntax for variable annotations (Type Hints)

For module-level vars, class attributes (both instance vars and class vars), and local-level variables.

Can be initialized or not.

Used for static analysis by 3rd-party tools such as PyCharm, mypy and pytype.

PEP

525: Asynchronous Generators

```
import asyncio

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)

async def main():
    async for x in ticker(1,10):
        print(x)

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

PEP

530: Asynchronous Comprehensions

```
import asyncio

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)

async def main():

    mylist = [y async for y in ticker(0.1, 10)]
    print(mylist)

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Library Module

secrets.py - Secure random numbers for secrets.

For any cryptographic purpose, use secrets module instead of os.urandom().

os.urandom() may block when it cannot provide enough random bits - e.g., containers and IoT devices.

Windows

PEP 528 (Console Encoding) and
PEP 529 (Filesystem Encoding

-> UTF-8

CPython Implementation Improvements

- dict type now uses separate index
  - ordered - though not yet reliably
  - (more details later)
- order preserved for **kwargs and class attributes
  - these are the only two reliable new “orderings”

Standard Library

asyncio module now considered stable

typing module no longer provisional
NewType() for lightweight types

pathlib makes available all path-like objects across different systems (‘Pure’ paths do not involve OS interaction - as opposed to ‘Concrete’ paths)

Raymond Hettinger

Modern Dictionaries
(PyBay, Dec 8 2016)

Presentation
https://www.youtube.com/watch?v=p33CVV29OG8

Original Proposal
https://mail.python.org/pipermail/python-dev/2012-December/123028.html

For example, the dictionary:

```
    d = {'timmy': 'red', 'barry': 'green', 'guido': 'blue'}
```

is currently stored as:

```
entries = [['--', '--', '--'],
                [-8522787127447073495, 'barry', 'green'],
                ['--', '--', '--'],
                ['--', '--', '--'],
                ['--', '--', '--'],
                [-9092791511155847987, 'timmy', 'red'],
                ['--', '--', '--'],
                [-6480567542315338377, 'guido', 'blue']]
```

Instead, the data should be organized as follows:

```
    indices =  [None, 1, None, None, None, 0, None, 2]
    entries =  [[-9092791511155847987, 'timmy', 'red'],
                [-8522787127447073495, 'barry', 'green'],
                [-6480567542315338377, 'guido', 'blue']]
```

Only the data layout needs to change. The hash table algorithms would stay the same. All of the current optimizations would be kept, including key-sharing dicts and custom lookup functions for string-only dicts. There is no change to the hash functions, the table search order, or collision statistics.

The memory savings are significant (from 30% to 95% compression depending on the how full the table is). Small dicts (size 0, 1, or 2) get the most benefit.

Dictionary Size

Version	Dict Size	Dict Ordering	Notes
Python 2.7	280, 280, 280	['sarah', 'barry', 'rachel', 'tim', 'guido']	Scrambled
Python 3.5	196, 196, 196	dict_keys(['sarah', 'tim', 'rachel', 'barry', 'guido'])	Randomized
Python 3.6	112, 112, 112	dict_keys(['guido', 'sarah', 'barry', 'rachel', 'tim'])	Ordered

Dictionaries All The Way Down

Improvement in implementation of dictionaries affects all of Python because most elements of Python are implemented with dictionaries.

3.6 is faster than 2.7 … for many workloads. Or more simply, Python 3 is just as fast as Python 2. (Cannon @ 22:00)

Twisted may be 3.6 and later …
(Cannon @ 19:40)
Windows now allows binary file paths …
(Cannon @ 19:00)

Misc

Wordcode - 16 bits - no more bytecode

pyenv -> python3 -m venv

OpenSSL < 1.0.2 deprecated

Porting to Python 3.6 (at end of page)

PEP 519 - pathlib supported by os and os.path

Official Docs

https://docs.python.org/3.6/whatsnew/3.6.html

Brett Cannon (PyCon Canada 28 minutes) [also at PyBay]

https://www.youtube.com/watch?v=hk85RUtQsBI

Dan Bader (9 minutes)

https://www.youtube.com/watch?v=klKdMxjDaa0

Python Bits (17 minutes)

https://www.youtube.com/watch?v=-P_XUzSTjh8

Raymond Hettinger (68 minutes)

https://www.youtube.com/watch?v=p33CVV29OG8