CPython Governance
Past, Present, and Future

EXIT
PYCON
PUNE
Hello,
Oπe\n!

I

Python history

Genesis

- Created in 1989 at CWI
- Publically released in 1991
- GvR joins CNRI in 1995
- Early 2000s: BeOpen.com & Zope Corporation

“If Guido was hit by a bus?”

I just returned from a meeting in which the major objection to using Python was its dependence on Guido. They wanted to know if Python would survive if Guido disappeared. This is an important issue for businesses that may be considering the use of Python in a product.

--Michael McLay 29 June, 1994

http://python-history.blogspot.com/

Python Enhancement Proposals (PEPs)

- Explanation of major feature and policy changes
- Accepted or rejected by GvR
- Lately also BDFL delegates
- Almost 500 written

Present state of the world

- Huge and growing userbase (TIOBE)
- Used in every conceivable computing domain
- No full-time core developers of CPython

:=

PEP 572
Assignment Expressions

PEP 572

```python
# before
env_base = os.environ.get("PYTHONUSERBASE", None)
if env_base:
    return env_base


# after
if env_base := os.environ.get("PYTHONUSERBASE", None):
    return env_base
```

PEP 572

```
reductor = dispatch_table.get(cls)
if reductor:
    rv = reductor(x)
else:
    reductor = getattr(x, "__reduce_ex__", None)
    if reductor:
        rv = reductor(4)
    else:
        reductor = getattr(x, "__reduce__", None)
        if reductor:
            rv = reductor()
        else:
            raise Error(
                "un(deep)copyable object")
```

```
if reductor := dispatch_table.get(cls):
    rv = reductor(x)
elif reductor := getattr(x, "__reduce_ex__", None):
    rv = reductor(4)
elif reductor := getattr(x, "__reduce__", None):
    rv = reductor()
else:
    raise Error("un(deep)copyable object")
```

June 12, 2018
BDFL “permanent vacation”

“[GvR] isn’t the Messiah, but he was a very clever boy.”

--The Economist July 19th, 2018

The sky did not fall

Meet the governance PEPs

PEP 8002: Open Source Governance Survey

- What models do other open source projects have?

- How do they resolve controversial decisions?

PEP 8010: The Technical Leader Governance Model

- Gracious Umpire Influencing Decisions Officer (GUIDO)
- Advisory council

PEP 8011: Trio of Pythonistas

group of 3 people serve for 5 years

PEP 8012: The Community Governance Model

- self-selected working groups handles PEPs

- modeled on Rust community

PEP 8013: The External Council Model

- authority delegated a council elected by core developers

- council members **can't** be core developers

PEP 8014: The Commons Governance Model

- anyone who wants can vote on a PEP

- voting process adjudicated by a Council of Elders

PEP 8015: Organization of the Python community

- Python teams empowered to own specific areas

- Steering council holding ultimate power

PEP 8016: The Steering Council model

Voting methods

PEP 13: Steering council

- Inspired by the Django project
- 5 members nominated and elected by core developers
- Serves for a single feature release (e.g., 3.7 -> 3.8)
- No three from the same company
- Wide powers exercised rarely

January steering council election

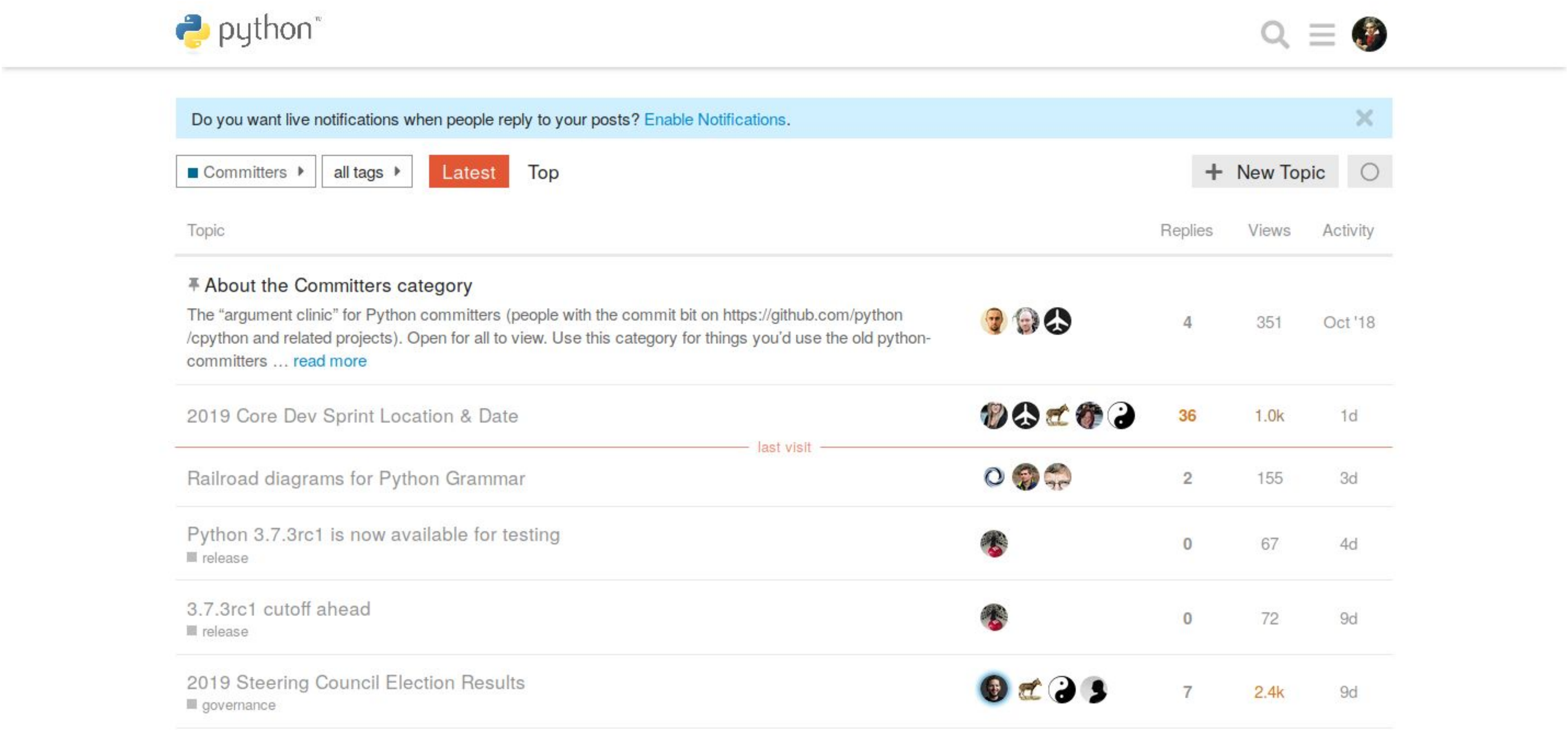
- 67 voters
- 17 candidates

The first steering council

- Guido van Rossum
- Carol Willing
- Nick Coghlan
- Brett Cannon
- Barry Warsaw

Speculation

Large scale discussions



Do you want live notifications when people reply to your posts? Enable Notifications.

Topic	Replies	Views	Activity
About the Committers category The “argument clinic” for Python committers (people with the commit bit on https://github.com/python/cpython and related projects). Open for all to view. Use this category for things you'd use the old python-committers … read more	4	351	Oct '18
2019 Core Dev Sprint Location & Date	36	1.0k	1d
Railroad diagrams for Python Grammar	2	155	3d
Python 3.7.3rc1 is now available for testing release	0	67	4d
3.7.3rc1 cutoff ahead release	0	72	9d
2019 Steering Council Election Results governance	7	2.4k	9d

Future of discussions

- Discourse
  - Replacing python-committers@
  - Better moderation
- GitHub for bugs?

Python 3

Python 2

https://python3statement.org/

Let’s not do that again.

Future of the core language

- Now under the committee
- Impossible to replace GvR
- Less important than ever

That’s all!

benjamin@python.org