Python PEP Talk

Wisdom from the Python Steering Council

Presented by
Kevin Woodman

A Brief History of Python

Highlights:

- 1991: Unveiled by Dutch programmer Guido van Rossum.
- 1994: Version 1.0 released.
- 1995: Guido dubbed “Benevolent Dictator for Life” by his team.
- 2000: First Python Enhancement Proposal codified.

A Brief History of Python

Highlights (continued):

- 2000: Version 2.0 released.
- 2008: Version 3.0 released.
- 2018: Guido steps down as BDFL after making a controversial decision that resulted in acrimonious public comments.
- 2019: Five-person Steering Council elected to fill the leadership void.
- 2024: Python 3.13 released.

PEP 20: The Zen of Python

Abstract

Long time Pythoneer Tim Peters succinctly channels the BDFL's guiding principles for Python's design into 20 aphorisms, only 19 of which have been written down.

PEP 20: The Zen of Python

Highlights

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.

PEP 20: The Zen of Python

Highlights (continued)

- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one – and preferably only one – obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

PEP 8: Style Guide for Python Code

Zen Points:

- Beautiful is better than ugly.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.

PEP 8: Style Guide for Python Code

A Foolish Consistency is the Hobgoblin of Little Minds

- Code is read much more often than it is written.
- Consistency with this style guide is important.
- Consistency within a project is more important.
- Consistency within one module or function is the most important.
- Know when to be inconsistent.

PEP 8: Style Guide for Python Code

Code Lay-out

- Use 4 spaces per indentation level.
- Spaces are the preferred indentation method.
- Tabs should be used solely to remain consistent with code that is already indented with tabs.
- Limit all lines to a maximum of 79 (or 99) characters.
- For flowing long blocks of text (docstrings or comments), the line length should be limited to 72 characters.

PEP 8: Style Guide for Python Code

Comments

- Comments that contradict the code are worse than no comments.
- Paragraphs inside a block comment are separated by a line containing a single #.
- Use inline comments sparingly.
- Write docstrings for all public modules, functions, classes, and methods.

PEP 8: Style Guide for Python Code

Naming Conventions

Language Element	Naming Style
Packages	lowercase
Modules	lowercase or lower_case_with_underscores
Classes	CapitalizedWords
Type Variables	CapitalizedWords
Exceptions	CapitalizedWords
Constants	UPPER_CASE_WITH_UNDERSCORES
Variables	lower_case_with_underscores
Functions	lower_case_with_underscores
Class Members	lower_case_with_underscores

PEP 484: Type Hints

Zen Points:

- Explicit is better than implicit.
- In the face of ambiguity, refuse the temptation to guess.

PEP 484: Type Hints

Highlights:

- Type hints are a specialized case of function annotations.
- Function annotations have no effect on execution and are intended for use with external tools.
- Type hints aren’t and will probably never be mandatory because Python is a dynamically-typed language.
- Type hints help linting tools (like Pylint) determine when a function argument is of an incorrect type.

PEP 484: Type Hints

Function Annotations

- Function annotations add metadata to function definitions.
- Type hints specify the type of each parameter as well as the returned type.

```
# Example of a function with no annotations
def fish_license(name = "Eric"):
    return None

# Example of a function with annotations
def fish_license(name: pet_id = "Eric") -> license:
    return None

# Example of a function with type hints
def fish_license(name: str = "Eric") -> None:
    return None
```

PEP 484: Type Hints

Type Hints

- Simple types include **int**, **float**, **bool**, **str**, **complex**, **Any**, and **None**.
- Compound types include **List**, **Union**, **Optional**, and **Callable**.

```python
from typing import Optional, Union

# Examples of compound & simple type hints
def breakfast(spam: Optional[bool] = True) -> str:
    return "Spam" if spam else "Egg and Bacon"

def lob_grenade(count: Union[int, float]) -> None:
    if count == 3:
        snuff_naughty_enemy()

# Example of a class type hint
from england import Comedy

def play_sketches(show: Comedy) -> None:
    pass
```

PEP 257: Docstring Conventions

Zen Points:

- Explicit is better than implicit.
- Readability counts.
- In the face of ambiguity, refuse the temptation to guess.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

PEP 257: Docstring Conventions

Highlights:

- Docstrings start & end with three quotation marks.
- Docstrings can be one-line or multiline.
- The first line is an imperative-voice executive summary.
- Subsequent lines are for all the details that a code user would need to use that function or module.
- PEP 257 does *not* mandate what details to include or what format to use.

PEP 257: Docstring Conventions

Module Docstrings:

- Should list everything that’s exported, each with a one-line executive summary beside it.

```
"""
Provide all the Monty Python movies as objects.

Exports:

and_now_for_something_completely_different -- Not!
monty_python_and_the_holy_grail -- The best one!
life_of_brian -- Banned for years in many places
the_meaning_of_life -- surprisingly, it’s not 42
"""
```

PEP 257: Docstring Conventions

Function Docstrings:

- Should (when applicable) summarize behaviour, describe arguments & return values, list side effects, list exceptions raised, and mention restrictions on when it can be called.

```
def never_be_rude(minority: str) -> None:
    """
    Warn people not to be rude to a minority.

    Being rude to other people can be hazardous to
    your health.  Just don't do it.

    Parameters:
    minority: str -- the people you're being rude to

    Raises:
    Bullet -- when someone has had enough
    """

    pass
```

PEP 257: Docstring Conventions

Class Docstrings:

- Should (when applicable) summarize behaviour, list public members, include notes for descendant classes.

```
class WoolamalooProfessor:
    """
    A philosophy professor at an Aussie university.

    Enjoys lager.  Knows cricket.  Sings about
    philosophers' drinking habits.

    Members:
    name
    department
    list_faculty_rules()
    """
```

PEP 257: Docstring Conventions

Good to Know:

- Docstrings are for code users; comments are for maintenance programmers.
- Documentation generator tools (like **pydoc**) use docstrings for their sources, but will ignore comments.

Rejected Markup Languages

XML/SGML Family:

- Verbose.
- Difficult to type.
- Too cluttered to read comfortably in source.

```
"""
<h1>It's the Mind</h1>

<p>Good evening.  Tonight on <b>"It's the Mind"</b>
we examine the phenomenon of <b>Déjà Vu</b> -- that
strange feeling we sometimes get that we've lived
through something before.</p>

<p><i>[Telephone rings]</i></p>
<hr>
<p><a href="https://www.imdb.com/title/tt0650969/">
IMDb</a></p>
"""
```

Rejected Markup Languages

TeX:

- Not very easy to write.
- Not easy for the uninitiated to read.

```
"""
\section*{It's the Mind}

Good evening.  Tonight on \textbf{"It's the Mind"}
we examine the phenomenon of \textbf{Déjà Vu} –
that strange feeling we sometimes get that we've
lived through something before.

\textit{[Telephone rings]}

\hrulefill

\href{https://www.imdb.com/title/tt0650969/}{IMDb}
"""
```

Rejected Markup Languages

Perl POD:

- POD (Plain Old Documentation) syntax “takes after Perl itself in terms of readability.”

```
"""
=head1 It's the Mind

Good evening.  Tonight on B<"It's the Mind"> we
examine the phenomenon of B<Déjà Vu> -- that strange
feeling we sometimes get that we've lived through
something before.

I<[Telephone rings]>

=begin text
---------------------------------------------------
=end text

L<IMDb|https://www.imdb.com/title/tt0650969/>
```

Rejected Markup Languages

JavaDoc:

- Uses HTML tags for most markup – thus it shares the readability problems of HTML.

```
"""
<h1>It's the Mind</h1>

<p>Good evening.  Tonight on <b>"It's the Mind"</b>
we examine the phenomenon of <b>Déjà Vu</b> -- that
strange feeling we sometimes get that we've lived
through something before.</p>

<p><i>[Telephone rings]</i></p>
@see
<a href="https://www.imdb.com/title/tt0650969/">
IMDb</a>
@author Graham Chapman
"""
```

PEP 216: Docstring Format

StructuredText:

- Lacks “essential” constructs that people want to use in their docstrings.
- There’s no way to escape markup characters.

```
"""
It's the Mind

    Good evening.  Tonight on **"It's the Mind"** we
    examine the phenomenon of **Déjà Vu** -- that
    strange feeling we sometimes get that we've
    lived through something before.

    *[Telephone rings]*

    "IMDb":https://www.imdb.com/title/tt0650969/
"""
```

PEP 287: reStructuredText Docstring Format

Zen:

- Beautiful is better than ugly.
- Readability counts.

PEP 287: reStructuredText Docstring Format

reStructuredText:

- Has markup for linking to an identifier’s source documentation.
- Is extensible.

```
"""
It's the Mind
=============

Good evening.  Tonight on **"It's the Mind"** we
examine the phenomenon of **Déjà Vu** -- that
strange feeling we sometimes get that we've lived
through something before.

*[Telephone rings]*

----

`IMDb <https://www.imdb.com/title/tt0650969/>`_
"""
```

PEP 287: reStructuredText Docstring Format

Highlights:

- It’s the most widely-used markup language in the Python development community.
- It’s powerful but complex and takes time to learn.
- **Sphynx** is the documentation generator that recognizes reStructuredText.

PEP 287: reStructuredText Docstring Format

Alternatives:

- **Markdown** is recognized by **MkDocs** and **pdoc**.
- The **Google** style guide for Python documentation is plain-text but recognized and prettified by **pdoc**.
- The **NumPy/SciPy** format (which blends reStructuredText and Google's style guide) is recognized by **Sphynx** (with a plugin).
- **Epytext** is based on JavaDoc and was used widely before PEP 287 was accepted but is now a legacy format.

What’s In Python’s Future?

Your Guess Is as Good as Mine!

- PEP 703: Making the Global Interpreter Lock Optional in CPython.
- PEP 744: JIT Compilation.
- Guido has recently re-emerged from the shadows and has adopted the title “Benevolent Dictator for Life Emeritus”.

PEP 666: Reject Foolish Indentation

Highlights:

- Settles arguments about indenting with tabs & spaces.
- “This proposal, if accepted, will probably mean a heck of a lot of work for somebody. But since I don’t want it accepted, I don’t care.”

PEP 801: Reserved

Notes:

- Wait – the built-package format in PEP 427 *isn't* the true wheel?
- And neither is the binary distribution format on the PyPA specs pages?