

Verifying commenting conventions

Bachelor's Thesis – First Presentation, 30/03/2021

By Michael Dooley

Supervised by Pooja Rani and Dr. Nataliia Stulova

University of Bern

Motivation

- Style Guidelines cover comments
- Style Checkers check comments

...but how?

Style Guidelines

- Codification of best stylistic practices
- For comments: Cover both content and style

PEP 257 -- Docstring Conventions

PEP:	257
Title:	Docstring Conventions
Author:	David Beazley (dave@dabeazley.com) and others (see PEP 0001 for authorship)
Discussion Url:	http://www.python.org/dev/peps/pep-257/
Status:	Active
Type:	Informational
Created:	20 May 2001
Last Revised:	20 Jun 2001

Table of Contents

- Abstract
- Rationale
- Specification
- One-line Docstrings
- Multi-line Docstrings

Abstract

The PEP documents the semantics and conventions associated with Python docstrings.

Rationale

The aim of this PEP is to standardize the high-level structure of docstrings when they should contain, and how to say it (with no formatting or any markup syntax within docstrings). The PEP contains conventions, not rules or syntax.

"A universal convention suggests a lot of maintainability, clarity, consistency, and a foundation for good programming habits too. What it doesn't do is insist that you follow it against your will. That's the Python!"

—Tim Peters on comp.lang.python.2001-06-16

If you violate these conventions, the worst you'll get is some dirty looks. But some software tools, such as the [Docutils](http://www.python.org/dev/peps/pep-257/#specification) HTML docstring rendering system, will fail to parse the conventions, so following them will get you the best results.

Specification

What is a Docstring?

A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a string literal becomes the `__doc__` special attribute of that object.

All modules should normally have docstrings, and all functions and classes exported by a module should also have docstrings. Public methods (including `__init__`) normally should also have docstrings. A docstring may be documented in the module docstring when `... ..` is the first line of the docstring.

Using docstrings to document Python code may also act as documentation. They are not managed by the Python runtime, but they are used as a reference for developers (in `__doc__`), and they are used as a reference for the standard library.

1. The string literal occurring immediately after a simple assignment in the top level of a module, class, or `__init__` method is called "module docstring".

2. The string literal occurring immediately after another docstring is called "additional docstring".

Please see [PEP 257, "Docstring Specification"](http://www.python.org/dev/peps/pep-257/#specification) for a detailed description of docstring and additional docstrings.

The following docstring is "right" in all respects: `"""equal docstring. Use """" for single quotes."""` If you use the backslashes in your docstring for double quotes, use `"""double quote docstring"""`.

There are no docstrings of docstrings, even those that define their docstrings.

One-line Docstrings

One-line docstrings are the only docstrings that should rarely be one line. For example:

```
def foo(x):  
    """Returns the path of the foo root directory."""  
    global foo_root  
    if foo_root: return foo_root  
    ...
```

Rules:

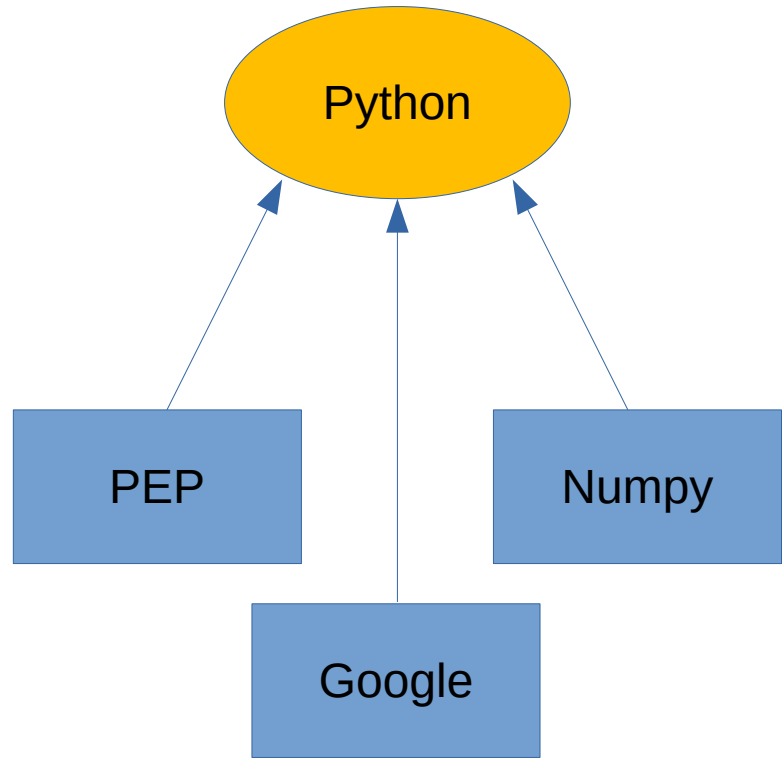
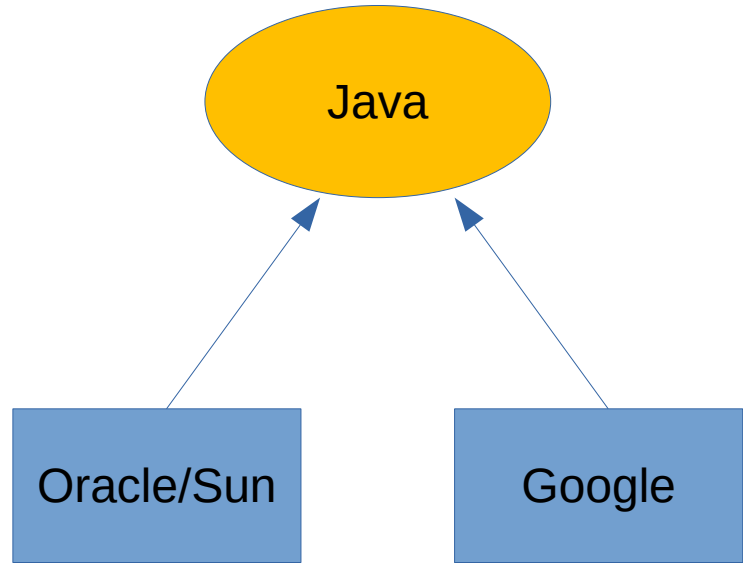
- Triple quotes are used even though the string fits on one line. This makes it easy to copy and paste.
- The closing quote is on the same line as the opening quote. This holds better for one lines.
- There's no blank line either before or after the docstring.
- The docstring is a plain string to be printed. It specifies the function or method without an annotation ("foo(x)", "foo(x)=", "foo(x)= ...", "foo(x)= ...").
- The one-line docstring is not a "signature" indicating the function/method parameters (which can be obtained by inspecting the object).

Multi-line Docstrings

Multi-line docstrings consist of a summary line just like a one-line docstring, followed by a blank line, followed by a more elaborate description. The summary line may be used by automatic indexing tools. It is important that the summary line itself is separated from the rest of the description by a blank line. The summary line may be the same line as the opening quote or on the next line. The closing docstring is indented the same as the opening and is the first line (see example below).

Don't blank line after docstrings (one line or multi line) that document a class, generally speaking, the class's methods are separated from each other by a single blank line, and the docstrings for the other methods are separated by a blank line.

```
def foo(x):  
    """Returns the path of the foo root directory."""  
    ...  
  
def foo(x):  
    """Returns the path of the foo root directory."""  
    ...
```



Style Guideline differences

PEP	Google
Use 4 spaces per indentation level.	Indent your code blocks with <i>4 spaces</i> .

Style Guideline differences

PEP	Google
Unless the entire docstring fits on a line, place the closing quotes on a line by themselves.	-

Style Guideline differences

PEP	Google
<p>Limit all lines to a maximum of 79 characters. For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.</p>	<p>Maximum line length is 80 characters.</p>

Style Checkers

Programs, that check code for consistent style

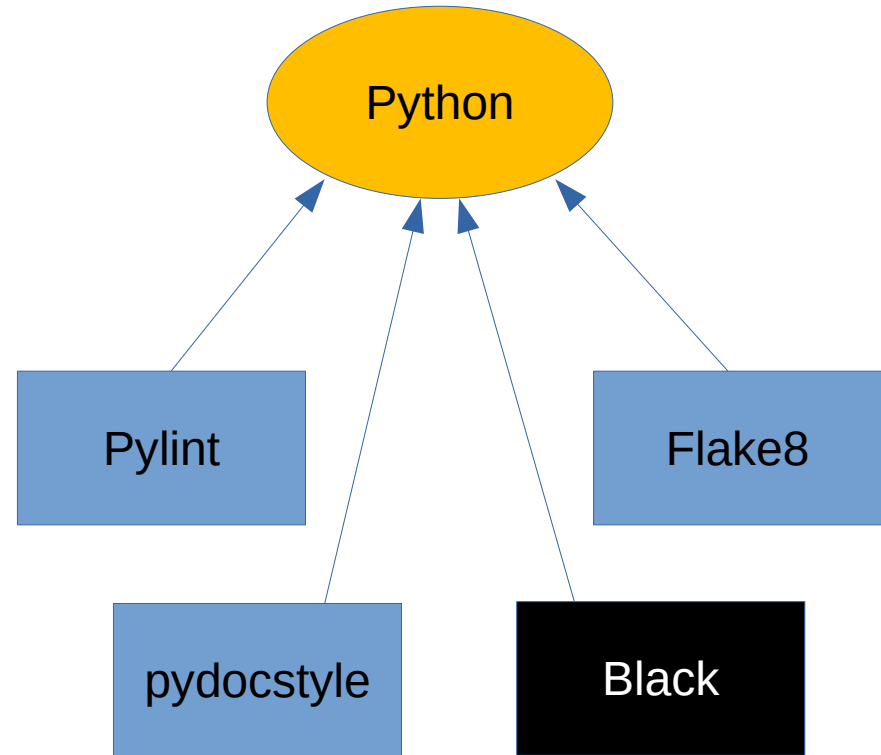
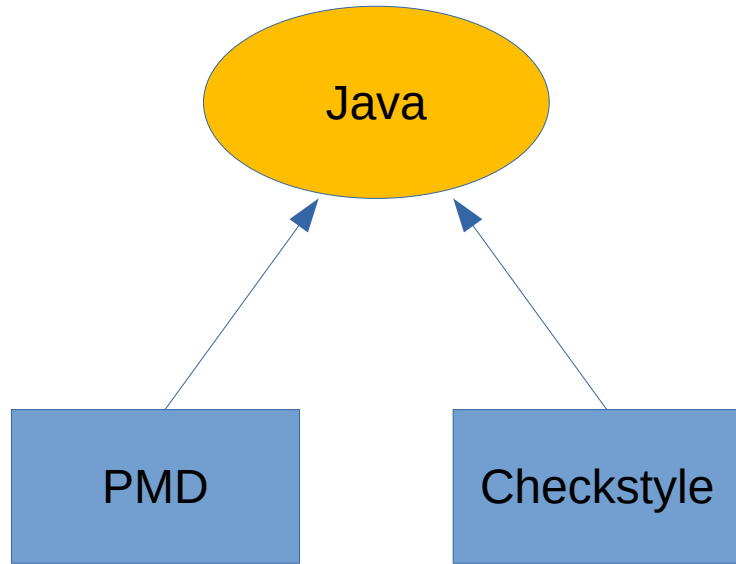
Style Checker Rules

Example (Pylint)

empty-docstring (C0112)

Used when a module, function, class or method has an empty docstring (it would be too easy ;).

Rules are often toggle-able or configurable



Question 1

What parts of style guidelines do style checkers cover with regards to documentation comments?

Related: How do style checkers differ compared to each other?

Example (PEP)

The docstring is a phrase ending in a period.

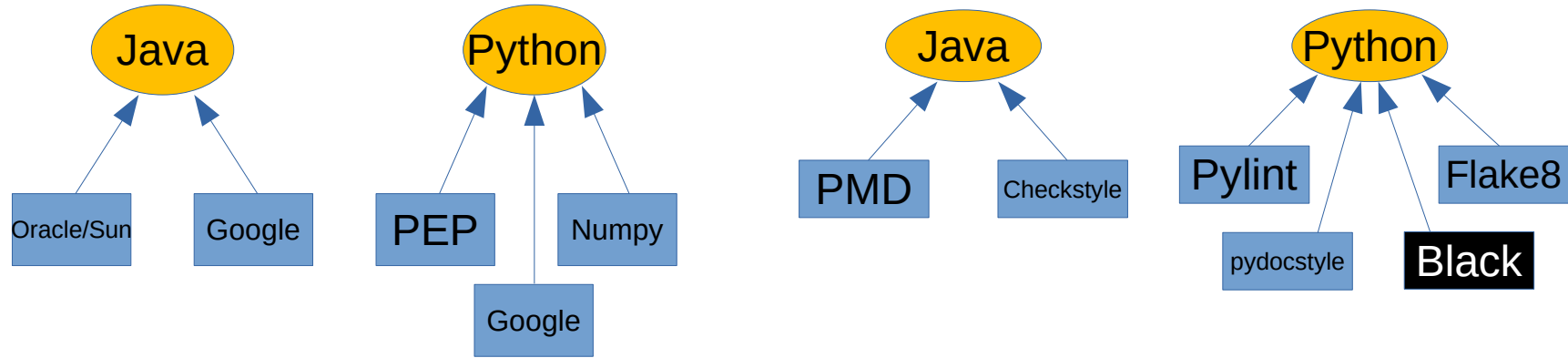
Question 2

What style checkers are used in open-source projects and how are they configured for comments?

Pipeline

1. Q1: Select guidelines and style checkers
2. Q1: Create a rule mapping
3. Q2: Mine projects and see if they use style checkers
4. Q2: Look at config files and compare them to defaults
5. Document and interpret results

Summary



What parts of style guidelines do style checkers cover with regards to documentation comments?

What style checkers are used and how are they configured for comments?