What are the most common Python docstring formats? [closed]

Asked 12 years, 1 month ago Modified 1 year ago Viewed 588k times

1135
votes

Closed. This question is opinion-based. It is not currently accepting answers.

Closed 4 years ago.

The community reviewed whether to reopen this question 7 months ago and left it closed:

Original close reason(s) were not resolved

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

I have seen a few different styles of writing docstrings in Python, what are the most popular styles?

python coding-style documentation docstring

Share

edited Nov 27, 2021 at 23:49
MatthewMartin
31.7k 32 108 163

asked Oct 10, 2010 at 1:10
Noah McIlraith
14k 7 28 35

6 python.org/dev/peps/pep-0008 there's a whole section devoted to documentation strings – mechanical_meat Oct 10, 2010 at 1:15

39 I think that this question was not clear enough because PEP-257 and PEP-8 are the establishing only the base for docstrings, but how about `epydoc`, `doxygen`, `sphinx`? Does anyone have any statistics, is one of them going to replace the others, in cases like this too many options can hurt. – sorin Dec 16, 2011 at 13:22

1 @sorin, I also would like to know what markup, if any, is most common. But I think the answer is that none of them is really all that common: people tend to prefer to look at the Python source directly, rather than converted to html. So, it's most useful to just be consistent but in a way that's optimized for human readability, and no explicit markup. – poolie Oct 21, 2012 at 7:31

3 PyCharm autocompletes in a rather interesting way, which I think is a nice implementation of the instructions needed to run it: `def foo(self, other):\n\t"""\n\t(blank line)\n\t:param other: \n\t:return:\n\t"""` – Matteo Ferla Mar 25, 2016 at 20:12

2 Which of these answers is the one that works by default with the VS Code documentation parser? – William Entriken Nov 9, 2019 at 4:13

Comments disabled on deleted / locked posts / reviews |

6 Answers

Sorted by: Highest score (default)

1390
votes

Formats

Python docstrings can be written following several formats as the other posts showed. However the default Sphinx docstring format was not mentioned and is based on **reStructuredText (reST)**. You can get some information about the main formats in this blog post.

Note that the reST is recommended by the PEP 287

There follows the main used formats for docstrings.

- Epytext

Historically a **javadoc** like style was prevalent, so it was taken as a base for Epydoc (with the called `Epytext` format) to generate documentation.

Example:

```
"""
This is a javadoc style.

@param param1: this is a first param
@param param2: this is a second param
@return: this is a description of what is returned
@raise keyError: raises an exception
"""
```

- reST

Nowadays, the probably more prevalent format is the **reStructuredText** (reST) format that is used by Sphinx to generate documentation. Note: it is used by default in JetBrains PyCharm (type triple quotes after defining a method and hit enter). It is also used by default as output format in Pyment.

Example:

```
"""
This is a reST style.

:param param1: this is a first param
:param param2: this is a second param
:returns: this is a description of what is returned
:raises keyError: raises an exception
"""
```

- Google

Google has their own format that is often used. It also can be interpreted by Sphinx (ie. using Napoleon plugin).

Example:

```
"""
This is an example of Google style.

Args:
    param1: This is the first param.
    param2: This is a second param.

Returns:
    This is a description of what is returned.

Raises:
    KeyError: Raises an exception.
"""
```

Even more examples

- Numpydoc

Note that Numpy recommend to follow their own numpydoc based on Google format and usable by Sphinx.

```
"""
My numpydoc description of a kind
of very exhautive numpydoc format docstring.

Parameters
----------
first : array_like
    the 1st param name `first`
second :
    the 2nd param
third : {'value', 'other'}, optional
    the 3rd param, by default 'value'

Returns
-------
string
    a value in a string

Raises
------
KeyError
    when a key error
OtherError
    when an other error
"""
```

Converting/Generating

It is possible to use a tool like Pyment to automatically generate docstrings to a Python project not yet documented, or to convert existing docstrings (can be mixing several formats) from a format to an other one.

Note: The examples are taken from the Pyment documentation

Share

edited Oct 13, 2019 at 3:30
LightCC
8,674 4 46 86

answered Jun 24, 2014 at 11:10
daouzli
15.3k 1 17 17

25 I might add that reST is what's used by default in JetBrains PyCharm, Just type triple quotes after defining your method and hit enter.jetbrains.com/pycharm/help/creating-documentation-comments.html – Felipe Feb 9, 2016 at 4:35

28 Most comprehensive answer, includes a sense of history and current best practices. Now all we need is some sense of community motion toward a new "best" format and some additional community effort toward creating migration tools from all the othersto the new one, so we could actually evolve best practice. – BobHy Jun 28, 2016 at 11:25

3 yo @daouzli, google style link is 404. I belive this one is correct. You can add sphinx google style example as well. Great answer btw. EDIT: I edited your answer by myself. – voy Jul 7, 2016 at 11:09

5 good answer. I dare say where you can change default docstring format in PyCharm (JetBrains): Settings --> Tools --> Python Integrated Tools --> Docstring format. Good luck! – Jackssn Mar 30, 2017 at 11:08

5 I'm surprised noone commented about the first text line: currently it's strictly speaking correct but i feel like the prefered way is to place it on the first line just after the triple quotes. PEP 8 and PEP 257 does it in almost all of their examples. PEP 287 does it your way, but in my experience it is not that common. – Lapinot Nov 14, 2017 at 20:59

354 votes

The Google style guide contains an excellent Python style guide. It includes conventions for readable docstring syntax that offers better guidance than PEP-257. For example:

```
def square_root(n):
    """Calculate the square root of a number.

    Args:
        n: the number to get the square root of.
    Returns:
        the square root of n.
    Raises:
        TypeError: if n is not a number.
        ValueError: if n is negative.

    """
    pass
```

I like to extend this to also include type information in the arguments, as described in this Sphinx documentation tutorial. For example:

```
def add_value(self, value):
    """Add a new value.

       Args:
           value (str): the value to add.
    """
    pass
```

Share

edited May 24, 2021 at 13:17
TylerH
20.4k 60 75 94

answered Nov 13, 2011 at 3:14
Nathan
5,220 2 24 27

49 I find the "signature in docstrings"-style awfully redundant and verbose. For Python 3+, Function annotations are a much cleaner way to do this. Even worse if it uses pseudo-strong-types: Python is way better with duck typing. – Evpok Jun 26, 2012 at 11:49

29 yeah, but at least it gives a hint of what sort of duck is expected, and majority of devs aren't on Python 3 yet – Anentropic Jul 26, 2012 at 11:47

4 @Evpok personally, I don't like function annotations. To use classes in them you might have to do unnecessary imports, to use strings in them you might run out of horizontal space very quickly describing them. So far I haven't seen the point of using them for anything. – OdraEncoded May 10, 2014 at 0:44

7 @Nathan, Google's style guide recommends comments that are descriptive rather than declarative, e.g. "Fetches rows from a Bigtable" over "Fetch rows from a Bigtable." Thus, changing "Calculate..." to "Calculates..." would make your example more consistent with the rest of the comment, i.e. "Returns" and "Raises". – jds Oct 28, 2014 at 12:30

3 nit: Following Google style, use descriptive rather than imperative form, i.e. "Calculates ..." and "Adds ..." – sbeliakov Jun 22, 2017 at 11:45

244
votes

Docstring conventions are in PEP-257 with much more detail than PEP-8.

However, docstrings seem to be far more personal than other areas of code. Different projects will have their own standard.

I tend to always include docstrings, because they tend to demonstrate how to use the function and what it does very quickly.

I prefer to keep things consistent, regardless of the length of the string. I like how to code looks when indentation and spacing are consistent. That means, I use:

```
def sq(n):
    """
    Return the square of n.
    """
    return n * n
```

Over:

```
def sq(n):
    """Returns the square of n."""
    return n * n
```

And tend to leave off commenting on the first line in longer docstrings:

```
def sq(n):
    """
    Return the square of n, accepting all numeric types:

    >>> sq(10)
    100

    >>> sq(10.434)
    108.86835599999999

    Raises a TypeError when input is invalid:

    >>> sq(4*'435')
    Traceback (most recent call last):
      ...
    TypeError: can't multiply sequence by non-int of type 'str'

    """
    return n*n
```

Meaning I find docstrings that start like this to be messy.

```
def sq(n):
    """Return the squared result.
    ...
```

Share

edited Dec 29, 2015 at 16:25
kzh
19.3k 13 71 96

answered Oct 10, 2010 at 5:36
Tim McNamara
17.7k 4 51 82

100 Note that PEP-8 specifically says that the docstrings should be written as commands/instructions, rather than descriptions, eg. `"""Return the squared result"""` rather than `"""Returns the squared result"""`. Although personally, I write mine how Tim's are here, despite what the PEP says. – Cam Jackson Aug 24, 2011 at 1:30

69 I also don't agree with that advice (using the imperative tense) because it starts sounding awkward for anything longer than one sentence. Furthermore, you are *describing* a function, not telling the reader what to do. – mk12 Aug 25, 2012 at 18:17

14 *Note: The specification for prescriptive rather than descriptive docstrings actually appears in PEP-257, not PEP-8.* I've come from a tradition of Java, where I was describing functions, but I finally began using the imperative tense when my programming paradigm switched from object-oriented to procedural. And when I began using pycco to generate literate-programming-style documentation, it became very apparent why the imperative tense was suggested. You should choose based on your paradigm. – karan.dodia Jun 5, 2013 at 20:56

28 The imperative is a grammatical *mood*. (Sorry.) – Dawn Drescher Jul 28, 2014 at 17:56

7 @Mk12 Git commit messages should also be written as commands instead of descriptions. And they are also "*describing*" a code change, "not telling the reader what to do". So I think it is just convention to write descriptions as commands. – onepiece Jul 22, 2015 at 12:50

67
votes

As apparently no one mentioned it: you can also use the **Numpy Docstring Standard**. It is widely used in the scientific community.

- The specification of the format from numpy together with an example
- You have a sphinx extension to render it: numpydoc

- And an example of how beautiful a rendered docstring can look like: http://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html

The Napolean sphinx extension to parse Google-style docstrings (recommended in the answer of @Nathan) also supports Numpy-style docstring, and makes a short comparison of both.

And last a basic example to give an idea how it looks like:

```
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Parameters
    ----------
    arg1 : int
        Description of arg1
    arg2 : str
        Description of arg2

    Returns
    -------
    bool
        Description of return value

    See Also
    --------
    otherfunc : some related other function

    Examples
    --------
    These are written in doctest format, and should illustrate how to
    use the function.

    >>> a=[1,2,3]
    >>> print [x + 3 for x in a]
    [4, 5, 6]
    """
    return True
```

Share

edited Nov 8, 2019 at 4:07
ali14
124 1 7

answered Apr 21, 2014 at 0:01
joris
127k 35 241 202

3 NumPy format IMHO takes too much vertical space which is scarce on widescreen monitors (except you use one turned by 90 degrees, but i guess most people don't) So, IMHO Google Format is a good choice with regard to readability and features. – Semanino Nov 6, 2018 at 10:00

6 I suppose it is somewhat subjective. Once you have a more complex docstring (with different sections, with examples, etc, so taking a lot of vertical space anyhow regardless of the format), I find the numpydoc format easier to read / better structured. – joris Nov 6, 2018 at 15:07

2 Personally I feel such a long docstring is better located in the documentation, not the source code, if it is so long they end up impeding readability of the module. – Jonathan Hartley Jan 7, 2019 at 16:41

---

12 votes

It's Python; anything goes. Consider how to *publish your documentation*. Docstrings are invisible except to readers of your source code.

People really like to browse and search documentation on the web. To achieve that, use the documentation tool Sphinx. It's the de-facto standard for documenting Python projects. The product is beautiful - take a look at https://python-guide.readthedocs.org/en/latest/ . The website Read the Docs will host your docs for free.

Share

edited Dec 31, 2014 at 10:13

answered Jan 11, 2013 at 19:29

Colonel Panic
129k 84 389 457

24 I routinely use `ipython` to test-drive a library, and it makes reading docstrings dead simple — all I have to type is `your_module.some_method_im_curious_about?` and I get a every nice printout, including docstring. – Thanatos Mar 4, 2013 at 7:59

9 The users of a *library* or of an *API* or who are writing a *plugin* are all likely to look at the code and need to make sense of it. I find comments far more crucial in Python than in Java or C# because types are not declared. It helps a lot if the comments give an idea of roughly what kinds of ducks are being passed and returned. (Otherwise, you have to actually walk all the code and tally up that a given parameter must... be iterable over here... support indexing over there... support numeric subtraction at the end... Aha! It's basically an int array. A comment would've helped!) – Jon Coombs Mar 31, 2014 at 18:47

2 Eh, no. Docstrings are **not** invisible and that's a bit of the point. You get to see the docstring if you run the `help` function on the documented function/method/class (and that you can do even if you only have access to the compiled module). Personally I think one should keep this in mind when choosing docstring convention (ie that it's intented to be read as is). – skyking May 26, 2019 at 8:21

NB: The linked video has been removed. – wovano May 27, 2021 at 8:09

8
votes

I suggest using Vladimir Keleshev's pep257 Python program to check your docstrings against PEP-257 and the Numpy Docstring Standard for describing parameters, returns, etc.

pep257 will report divergence you make from the standard and is called like pylint and pep8.

Share

answered Sep 11, 2014 at 15:34
Finn Årup Nielsen
5,758 1 29 40

1 Mentioning PEP-257 in the context of "how should i properly document parameters, return values, exceptions raised etc" is a JOKE - it says not a single word about them (although a code example shows some). NumPy format IMHO takes too much vertical space which is scarce on widescreen monitors (except you use one turned by 90 degrees, but i guess most people don't) So, IMHO Google Format is a good choice with regard to readability and features. – Semanino Nov 6, 2018 at 10:00

1 @Semanino I am mentioning the Numpy Docstring Standard in the context of the pep257 program, - not PEP-257. That program is now called pydocstyle. pydocstyle allows you to do some numpydoc checks, e.g., `pydocstyle --select=D4 tmp.py` checks for a range of docstring content issues including section naming. – Finn Årup Nielsen Nov 7, 2018 at 13:34