The Zen of Docstrings

"""

"""

Safe Hammad
Python Northwest
16th December 2010

Is this a question?

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
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.
Now is better than never.
Although never is often better than *right* now.
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.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

The Docstring

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

“INTROSPECTION”

- Compare with comments which can only be accessed by viewing the source code.

__doc__

mymod.py

```
#!/usr/bin python

"""This is my module."""

class MyClass(object):
    """This is my class."""

    def my_method(self, a, b):
        """This is my method."""

        # I am a comment which
        # can only be accessed
        # and viewed right here!
        return a + b
```

Interactive interpreter

```
>>> import mymod

>>> mymod.__doc__
'This is my module.'

>>> mymod.MyClass.__doc__
'This is my class.'

>>> m = MyClass()
>>> m.my_func.__doc__
'This is my method.'

>>>
```

Formatting

The major source of docstring fanaticism is the format of the function or method docstring.

PEP Talk

PEP 8

- Write docstrings for all public modules, functions, classes and methods.
- PEP 257 describes good docstring conventions. Most importantly, the “”” that ends a multiline docstring should be on a line by itself, preferably preceded by a blank line, e.g.:

```
"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""
```

- For one-liners, it's okay to keep the closing “”” on the same line.

PEP Talk

PEP 257

- Always use triple-quotes, even for one-liners.
- Prescribe a method's effect as a command e.g. “Do this”, “Return that”.
- Don't reiterate a method's signature.
- List each method parameter on its own line.

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real – the real part (default 0.0)
    imag – the imaginary part (default 0.0)
    """
```

- In a multi-line docstring, the BDFL recommends inserting a blank line between the last paragraph and the closing quotes.

Methodism

```
def complex(real=0.0, image=0.0):
    """Form a complex number.

    Keyword arguments:
    real – the real part (default 0.0)
    imag – the imaginary part (default 0.0)
    """
```

```
def complex(real=0.0, image=0.0):
    """Form a complex number.

    :Parameters:
    - `real`: The real part (default 0.0)
    - `imag`: The imaginary part (default 0.0)
    """
```

Methodism (cont.)

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    :Param real:
        The real part (default 0.0)
    :Param imag:
        The imaginary part (default 0.0)
    """
```

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    @param real: The real part (default 0.0)
    @type real: float
    @param imag: The real part (default 0.0)
    @type real: float
    """
```

The Benefits of Introspection

Through the power of introspection, a docstring can be put to use outside of the context of reading source code.

1. DOCUMENTATION
2. TESTING VIA DOCTEST

Other Benefits

TDD vs. DDD

Docstring Driven Development can sometimes be as effective as Test Driven Development!

Automated External Documentation

- Epydoc:
  - Geared toward automated api documentation directly from source code.
  - Understands several styles of annotation including reStructuredText, Javadoc and Epytext.
- Sphinx:
  - Designed primarily for documentation generated from non-source files but includes directives to auto-document source code.
  - Uses and extends reStructuredText.

Doctest

```
def factorial(n):
    """Return the factorial of n, an exact integer >= 0.

    If the result is small enough to fit in an int, return an int.
    Else return a long.

    >>> [factorial(n) for n in range(6)]
    [1, 1, 2, 6, 24, 120]
    >>> [factorial(long(n)) for n in range(6)]
    [1, 1, 2, 6, 24, 120]
    """

    … implementation …


if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

What about attribute docstrings?

```
class Foo(object):

    bar = 5
    """The barness of the foo"""
```

- PEP 224 attempted to introduce them but was rejected:

  “It's not the implementation, it's the syntax. It doesn't convey a clear enough coupling between the variable and the doc string.”

- Some documentation tools will recognise and parse them.

Function Annotations

```
def complex(real: “Real part”, imag: “Imaginary part”) -> “A
complex number”:
```

- Introduced with Python 3 as part of PEP 3107.
- “... makes no attempt to introduce any kind of standard semantics, even for the built-in types.”
- Can be introspected:

```
>>> complex.func_annotation
{'real': 'Real part', 'imag': 'Imaginary part', 'return': 'A
complex number}
```

- External documentation and testing is a possible use!

Rules of thumb

- Explain what the method does.
- Explain what the method returns.
- Explain the expected arguments.
- List the exceptions thrown.

The Zen of Docstrings

```
>>> import that
The Zen of Docstrings, by Safe Hammad

- Internal consistency is more important than blindly
following a prescribed rule.

- But blindly following a prescribed rule is better
than following no rule at all.

- Some docstring is better than no docstring.

- But no docstring is much better than a misleading or
incorrect docstring.

- Above all, follow the rule of least surprise.  If
your method does something surprising or non-obvious,
then document it.
>>>
```

Thanks!

Safe Hammad
http://safehammad.com