Doc String and PEP 8 in python

Python docstring are the string literals that appear right after the definition of a function, method, class or module.

Example:

```
def square(n):
    '''Take in a number n, return the square of n'''
    print(n**2)

square(5)
print(square)
print(square.__doc__)
```

Python Comments vs Docstring

Python Comments

Comments are descriptions that helps programmers better understand the intent and functionality of the program. They are completely ignored by the Python interpreter.

Python Docstring

As mentioned above, Python docstring are strings used right after the definition of a function, method, class or module (like in Example). They are used to document our code.

We can access these docstring used the doc attribute.

Python Doc attribute

Whenever string literals are present just after the definition of a function, module, class or method, they are associated with the object as their doc attribute. We can letter use this attribute to retrieve this docstring

```
def square(n):
    '''Take in a number n, return the square of n'''
    print(n**2)

square(5)
print(square)
print(square.__doc__)
```

PEP 8

PEP 8 is a document that provides, guidelines and best practice on how to write python code. It was written 2001 by Guido van Rossum, Barry Warsaw, and nick coghlan. The primary focus of PEP 8 is to improve the reliability and consistency of Python code.

PEP stands for Python Enhancement proposal, and there are several of them. A PEP is a document that describes new features proposed for python and documents aspects of python. Like design and style, for the community.

The Zen of python

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

```
>>>
>>> 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!
>>>
```

Source Code

```python
def square(n):
    '''Take in a number n, return the square of n'''
    print(n**2)

square(5)
print(square)
print(square.__doc__)
```

Thank You