Rough Style Guidelines

Commenting

Comments in Python are created using the pound sign (#) and should be brief statements that help clarify your code for the reader. Sometimes it is necessary to include longer comments. When this is the case, the comment should NOT exceed 72 characters to a single line. For example,

```
def hello_long_world():
    # A very long statement that just goes on and on and on and on and
    # never ends until after it's reached the 80 char limit
    print("Hellooooooooooooooooooooooooooooooooooooooooooooooooo World")
```

This is how longer comments should be treated. Please do NOT do the following:

```
def hello_long_world():
    # A very long statement that just goes on and on and on and on and never ends until after it's reac
    print("Hellooooooooooooooooooooooooooooooooooooooooooooooooo World")
```

Documenting Code

Docstrings allow you to provide an overview of your code to readers. Providing adequate documentation of your code allows readers to call Python's help() function on any objects you define which makes your code more accessible and less of a headache for everyone who has to use it. In Python, you can create docstrings by proper placement of multi-line comments. In all cases, your docstrings should use the triple quote (""") string format. Here I will outline how to Document your classes and their methods but if you would like to know more you should check out this tutorial which covers documentation for classes, scripts, and Python modules.

Class docstrings should contain the following information:

— A brief summary of its purpose and behavior

— Any public methods, along with a brief description

— Any class properties (attributes)

Class constructor parameters should be documented within the ___init___method docstring. Individual methods should be documented using their individual docstrings. When documenting class methods you should include the following:

— A brief description of what the method is and what it's used for

— Any arguments (both required and optional) that are passed including keyword arguments

— Label any arguments that are considered optional or have a default value

— Any side effects that occur when executing the method

— Any exceptions that are raised

— Any restrictions on when the method can be called

Here I provide a generic example of the docstring for some class named MyClass. This example is not perfect but if you look at **lab1.py**, the class **Node** has a docstring provided to you and it helps demonstrate proper documentation for a class.

```
class MyClass:
    """
    A brief summary of the class purpose or behavior

    ...

    Attributes
    ----------
    atr1 : type
        brief summary of atr1
    atr2 : type
        brief summary of atr2 (default is 4)

    Methods
    -------
    my_method(arg1=None)
        Summary of my_method
    """

    def __init__(self, arg1, arg2=4):
        """
        Parameters
        ----------
        arg1 : type
            Description of arg1
        arg2 : type, optional
            Description of arg2 (default is 4)
        """

        self.atr1 = arg1
        self.atr2 = arg2

    def my_method(self, arg1=None):
        """Summary of my_method.

        If further elaboration is necessary put it here, one line spacing from
        the method summary.

        Parameters
        ----------
        arg1 : type, optional
```

```
42            Description of arg1 (default is None)
43
44        Raises
45        ------
46        AssertionError
47            Explain why an exception was made
48            (In this case its an arbitrary decision)
49        """
50
51        try:
52                assert(self.atr1 == 2)
53        except:
54                raise AssertionError ("false")
55
56        return None
```