Rubric to Check Clean Code

In the final project one of the things we will be grading quite heavily is on clean coding practices. Clean code is essentially a way of writing code that is easily readable and understandable. Here we will outline some of the things we will be looking for in your final project and a break down as to how we will grade them.

1. **Clear function and variable names**
    a. Whenever you create a function or assign a variable to something make the names of these meaningful.
    b. For example if you are making a function that adds two numbers you may consider calling the function add_two_num()
    c. If you are storing the name of a file to a variable, you may consider storing it in the variable name filename.

2. **White spaces between code**
    a. Put white spaces in between different lines of your code. This may be to break up different sections of your functions. Having spaces before and after loop or if statements are good tips to follow to increase the readability of your code.

3. **Comments and Docstrings**
    a. However, sometimes it is not enough to just have clear variable names and functions. When you run into a situation where your code is doing something that is not intuitive or is hard to follow even with clear variable names you want to use comments and docstrings to remind yourself and others what is going on.
    b. Comments are simple 1-2 lines of general information about what is going on in your code. Be sure to use comments in areas of your code that is hard to understand or is unclear if someone else reads it.
    c. Docstrings are used in functions to provide you and someone reading your code with basic information about the function.
    d. Whenever you make a function we expect to see a docstring outlining these criterias
        i. Function's purpose, if you are doing a non-intuitive calculation you may want to specify the equation you are using in the calculation here
        ii. Parameters that the function takes in.

            1. This means the name of the parameters you need to pass into the function as well as the data type of those parameters.
        iii. What the function returns
            1. The name and what is the data type that it returns

4. **Make function for similar code**
    a. If you see yourself copy and pasting a lot of code that is similar but changes only slightly it is highly encouraged to write a function.
    b. You want to write the function that to have the same structure as the code you are copy and pasting and takes as its parameters the things that changes

5. **DO NOT HARDCODE!!!!**
    a. Hardcoding is something you never want to do when you make a program. The reason being is that code should be tailored to handle all/most cases of a problem you are tasked with.
    b. If you need to include a number into a calculation be sure to store that into a variable.
    c. If we look at project 1 and project 2 for example, we should be able to play mastermind with any guesses and numbers generated. We should also be able to take any spectra that was collected in the same manner as project 2 and reproduce the end result.
    d. In short hardcoding only tailors to specific type of inputs when in reality we could put into the functions any arbitrary inputs.

6. **Good logic of the program**
    a. By this we mean that it is easy to see what you are attempting to do.
    b. There is a clear logic into the way you laid out your code and we can follow it well with or without the use of comments

We will grade using a check, check+, check++ and check- rubric. By default we will assign everyone to have a check which is a 70%. If we see good effort upholding the rubric laid out above we will bump it up to a check+ and if it is really well written awarded a check++. If no effort is made to make the code clean and legible then we will give a check- which is a failing grade for this portion.