Coding Expectations

On this page, I outline my primary expectations for your submitted programming assignments at various points in the quarter. You may find it useful to check the expectations for submitted assignments for concrete instructions on how to turn in work in this course.

Programming is a creative endeavor, and there are usually multiple good ways to solve a problem. As programmers, we have the responsibility of not only solving the problem at hand but also doing so in a way that is logical, performant, documented, and easy to read. Users of your code—likely including your future self—will appreciate time you put into polishing your code.

Beyond pure correctness, there are three primary axes along which you should evaluate your code.

1) **Style**: is my code as concise as it reasonably can be? Are any parts of my code redundant? Am I unnecessarily repeating any computational operations? Would someone who has never seen my code before be able to read my program? If I do need to perform the same task multiple times, can I wrap it inside a function or a for-loop? Am I making full appropriate use of the tools covered in the course so far?

2) **Documentation**: does my code include helpful comments to guide the reader? Do my functions and classes possess docstrings? When the assignment requires, have I included sufficient surrounding text to explain my thought process?

3) **Robustness**: does my program check user inputs and produce informative error messages if inappropriate inputs are supplied? Does my code include unit tests that guarantee the correct operation of important components?

In this course, we will place the most emphasis on style and documentation. Expectations on these axes will be in force on most assignments. Certain assignments will ask you to focus on robustness as well.

My expectations of you will evolve as you develop your Python programming skills. Here's where I expect you to be at various checkpoints in the quarter. These expectations stack, so the expectations at Week 7 also include those from earlier weeks.

Week 1

You are taking your first steps in Python, and are still figuring out syntax, data types, and control flow.

1) **Style:** You can identify which parts of your code are necessary and which are unnecessary. You use loops to avoid duplicating code.

2)**Documentation:** You use many comments to explain to your reader the purpose of most code blocks. When required, you use surrounding text in the Jupyter Notebook to explain the workings of your solution.

3) **Robustness:** You can predict how your code might behave when various kinds of input are supplied. You can test your predictions through simple experiments.

Week 2

1) **Style:** You use functions to reuse important functionality in your programs.

2) **Documentation:** You write thorough docstrings for your functions The docstring includes a summary of the purpose of the function, the types of the expected inputs, and a description of the output. Your docstring should be sufficient to explain to a user what your function does, even if they have never seen it before.

3) **Robustness:** When required, you raise appropriate and descriptive exceptions at key points in your code. These include checking user inputs and helpfully describing failure states of function calls.

Week 4

1) **Style:** You define custom classes and instantiate objects in order to create programs with memory. You minimize the use of global variables in your code.

2)**Documentation:** You write helpful docstrings for your custom classes and all of their methods, with the possible exception of very simple **init**() definitions. Your docstrings make clear to new users the purpose of your class and how it behaves.

3) **Robustness:** When required, you implement simple, automated unit tests to check your functions and classes for correctness.

Week 7

1) **Style:** You use array indexing, especially boolean indexing, to retrieve and modify sections of arrays and data frames. You use vectorized functions to compute on portions of arrays and data frames. You do not use for-loops to iterate over the elements of an array or data frame.

2) **Documentation:** As in previous weeks.

3) **Robustness:** As in previous weeks.

In [ ]: