Functional Code vs. Professionally Ready Code

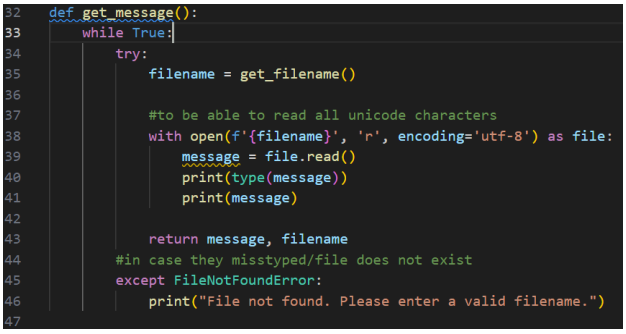
Functional Code:

- Works as intended
- Could be disorganized and disorientating for the reader
- Cannot return to a previous version easily

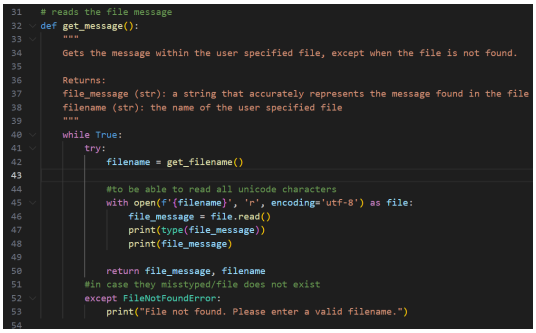
Professionally Ready Code:

- Works as intended
- Organized and correctly styled for the reader using formatting tools such as Pylint
- Version Control using Git allows for the easy return to a previous version

Below, I will go through some examples of the formatting and version maintenance that I have been doing through images and words to show how I went from functional code to professionally ready code.



First, I will demonstrate code quality analysis. Before using Pylint, my function did not have a doc string, which are informative lines of text that describe what the function does, what arguments the function takes, and what does the function returns. This is shown by the dark blue squiggly line under the function in the image above. Secondly, I had a variable called message, except I had already used that variable name in an outer scope somewhere else in my code. In order to prevent confusion, that message variable should be renamed, as shown by the orange squiggly line under the variable



message. My final revisions for the get_message function (shown in the image above) did include an informative doc string and I renamed the message variable as file_message in order to prevent any confusion about what message I was referring to through that variable.

Now I will overview my attempts to use Git and maintain some version control. First, I had to download Git, of course. Then I watched a tutorial on how to use Git within VSCode, the interpreter that I am using, and created a test repository that contains a main branch. In order to stage a change, I first had to change the file from U, which means untracked files, to A, which means an added file the Git recognizes. Commits are like a save, except you can easily go back and revert to a commit. I had my initial commit, and then a patch commit, where my initial commit referred to the first save and the patch commit referred to a significant change I had after my first save. A side by side representation of the initial version and patch versions show the differences made in the commits. And that is how I went from functional code to professionally ready code on my first project!