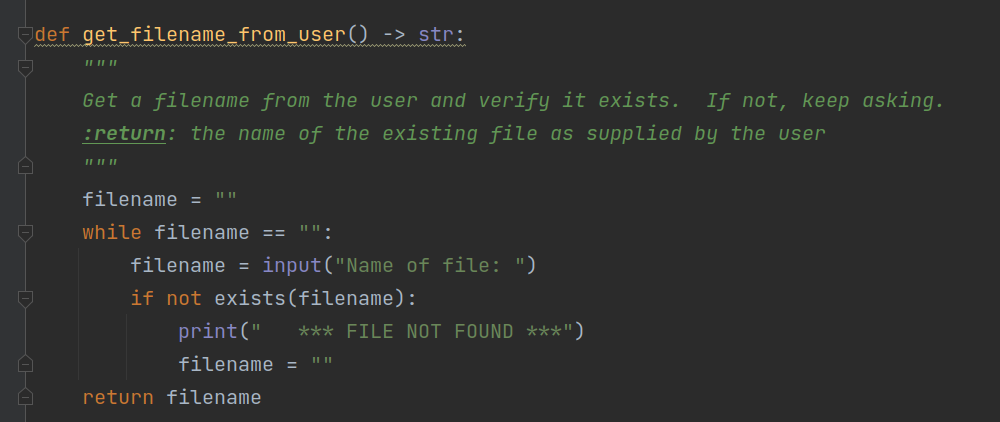
===================

CODE OVERVIEW

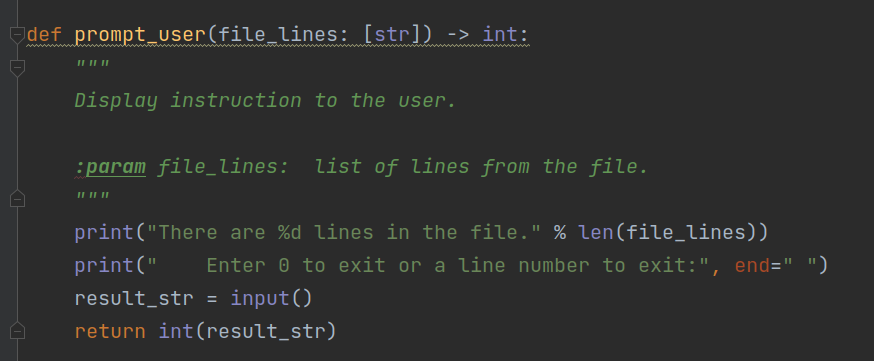
===================

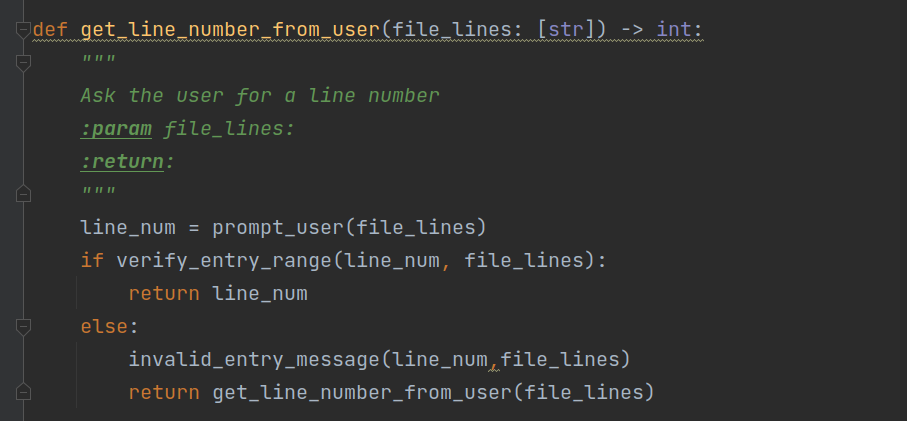
(1a) Start by writing code to prompt the user for a filename.



Note: I did verify the file exists before allowing the user to continue on.

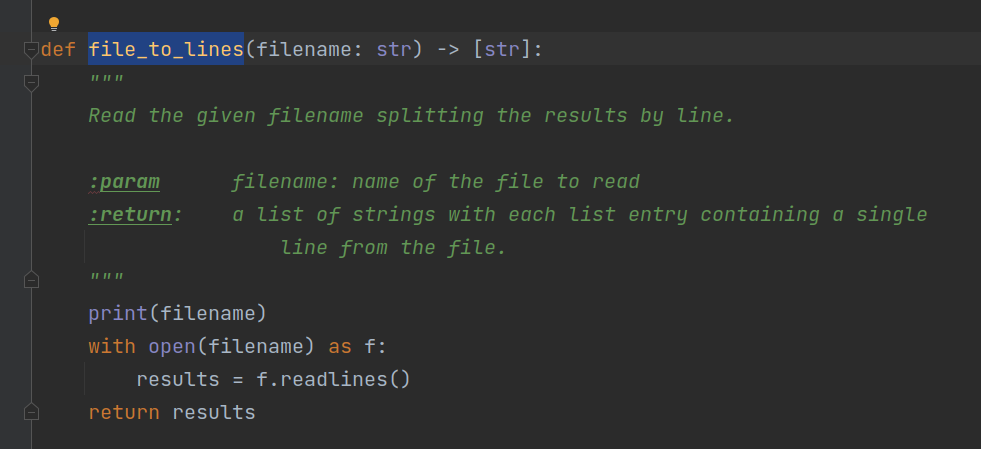
(1b) Also, write similar code to get a line number from the user.



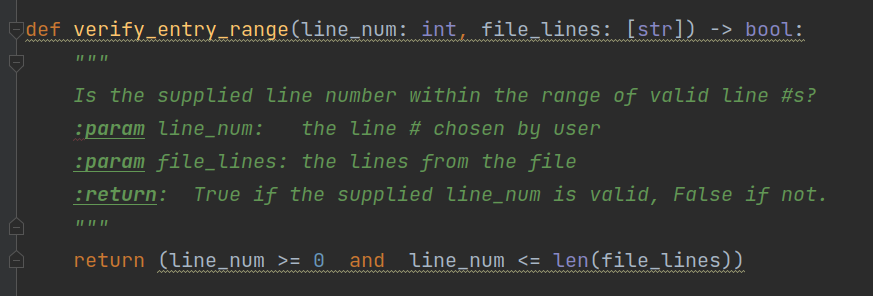


Note: error handling for non-numeric input is not supplied to (1) simplify the code here and (2) because the class has not covered exceptions thus far.

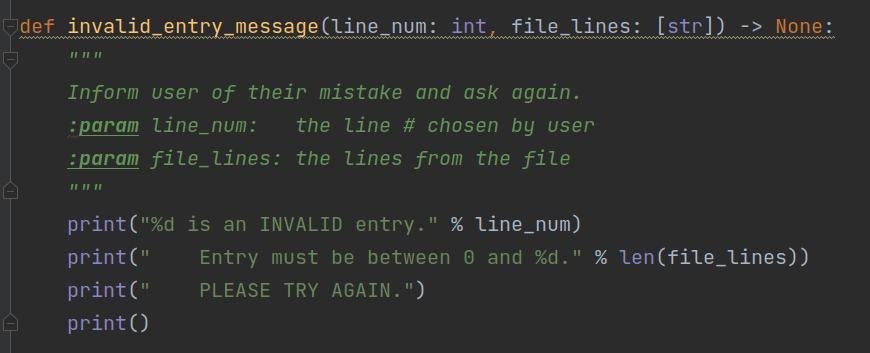
(2) Now, it is time to read the file into a list of strings:



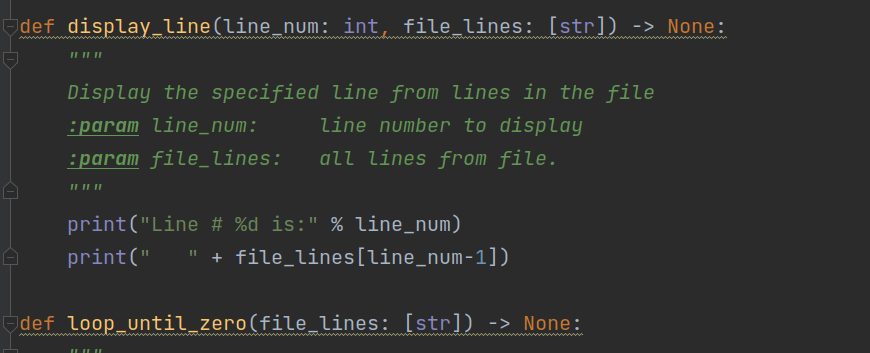
(3) I wrote a routine to verify the entry range. Not strictly necessary as the verification is simple, but if things were to become more complex, this will make the check obvious to the next programmer.



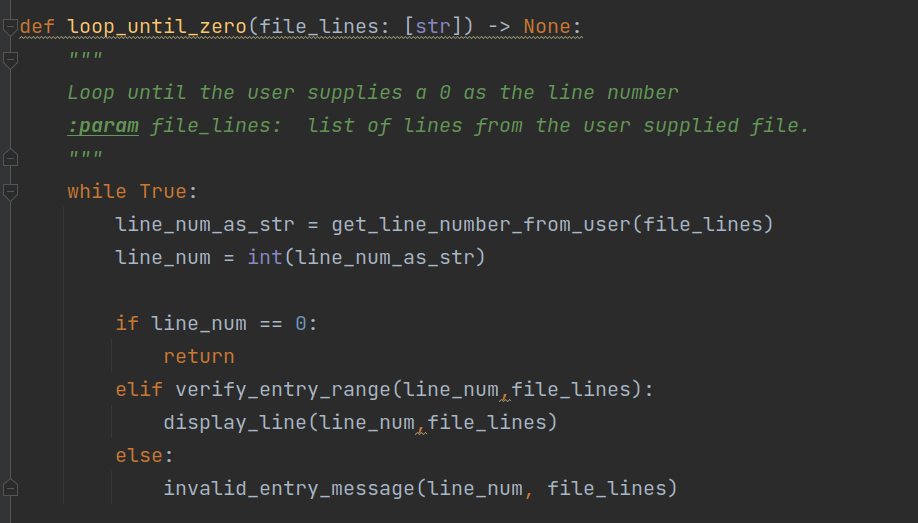
(4) Write a routine that displays the user a message if he enters an invalid line number.



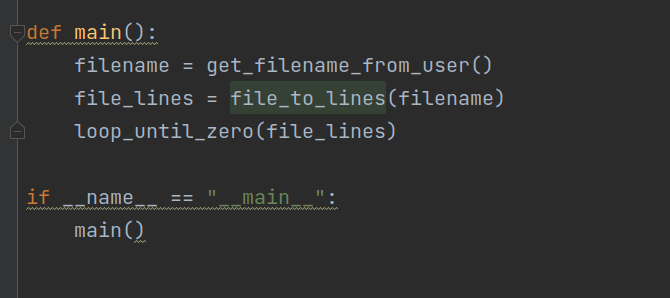
(5) Write a routine to display the line the user has selected.



(6) Getting near completion. We must write the infinite loop requested by the assignment. It will exit if a 0 is received, display a line if a valid line number is received, or display an error message if an invalid line number is received.



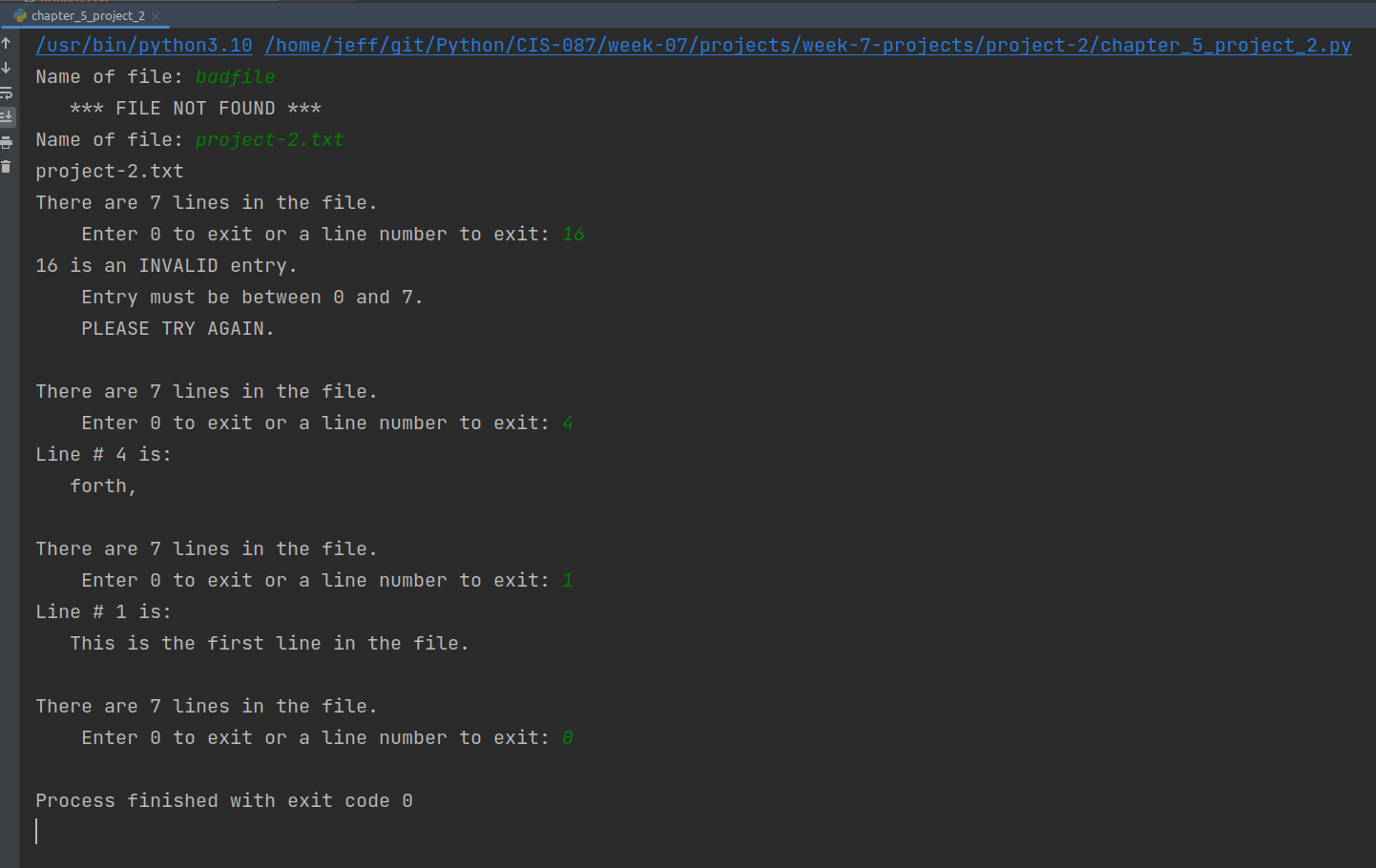
(7) Wrap it all in a main routine to allow it to be called from the command line (or similar tool).



===================

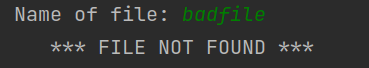
EXECUTION

===================

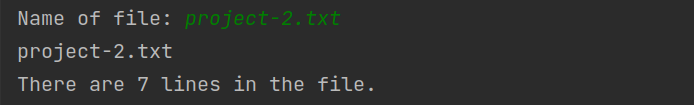
Complete session

(1) Start the program.

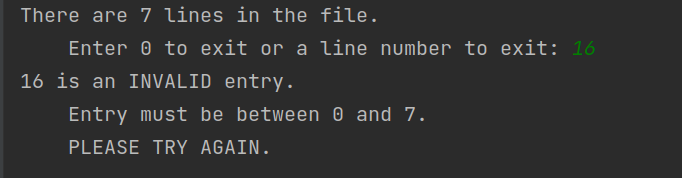
(2) Supply a bad filename to test verification of file exists checks:



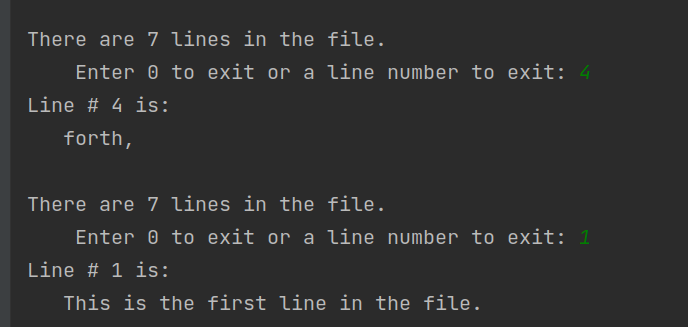
(3) Supply the correct filename and verify the line count. Note: real world text files should end with a new line character. This causes the count to look like it has an extra line. This would be handled by verifying the last line is empty and reporting an error if not. Alternatively, we could strip the newline characters from all lines and allow an incorrectly formed text file.



(4) Supply an invalid line # to verify range check:



(5) Supply a valid line # and verify the correct line is displayed (2 checks provided):



(6) Supply a 0 and verify program exits:

