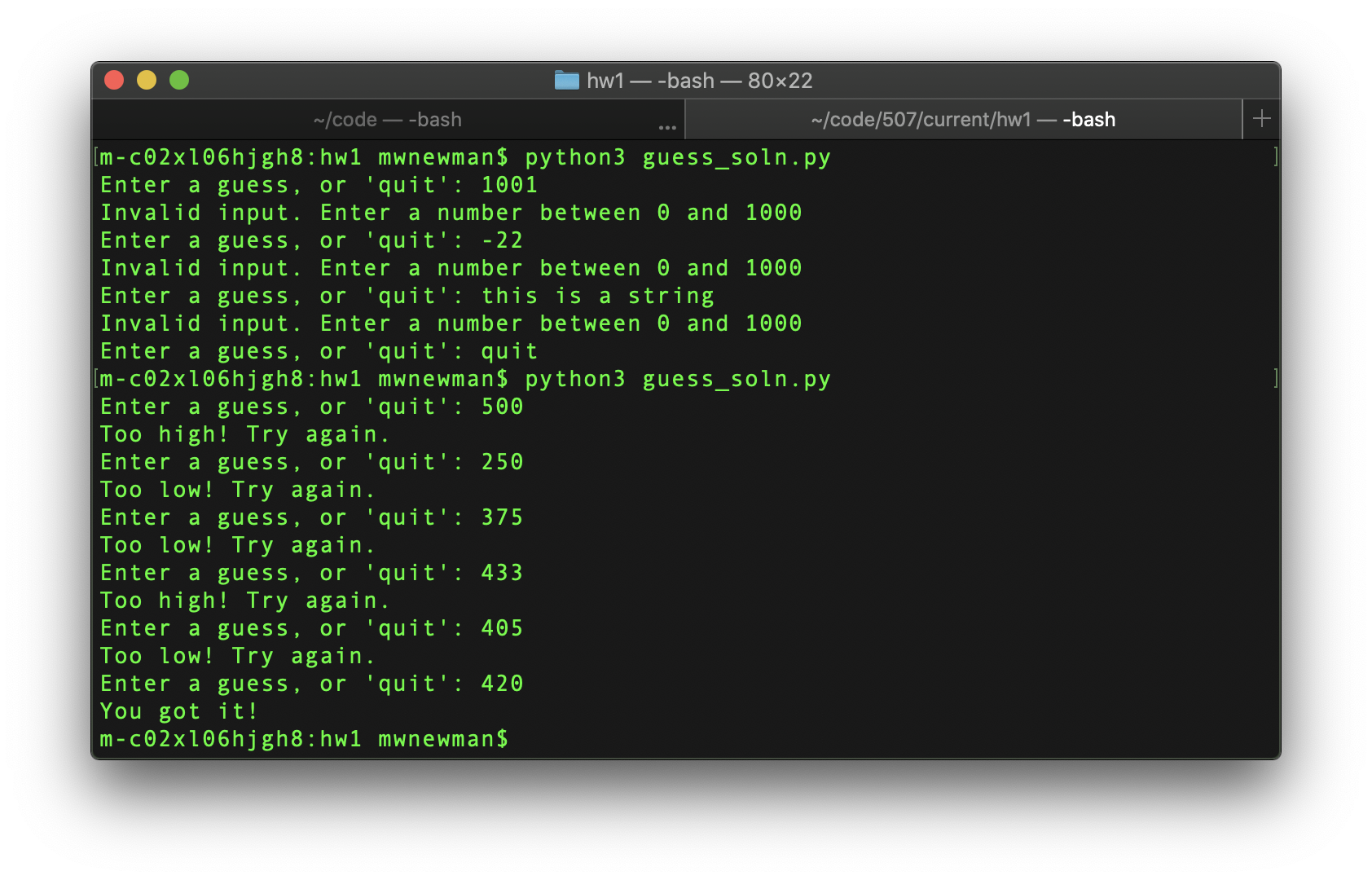
Instructions

1. Download [guess.pyPreview the document](https://umich.instructure.com/courses/343048/files/13615210/download?wrap=1) and run it in your terminal.
2. As you can see, the current version of the program isn't very fun to use. The user gets one guess, which is almost certainly wrong, and then the game is over. Your job is to change the program so that it
   1. Allows the user to keep guessing until they guess correctly or give up and type "quit"
   2. Gives the user a hint by indicating whether their guess was higher or lower than the correct answer.

Here is a sample run of the corrected program that you need to create. Your program should be able to match this output *exactly.*The sample program here is named guess\_soln.py (for guess *solution*), but yours should still be called guess.py.



Grading

|  |  |
| --- | --- |
| **Requirement** | **Points** |
| Accept user input | 20 |
| Indicate incorrect guess | 15 |
| Indicate correct guess | 15 |
| Reject input that is out of numeric range | 10 |
| Reject non-numeric input | 10 |
| Indicate whether guess is too high or too low | 10 |
| Continue until correct guess (if user does not enter "quit" and there are no max tries) | 10 |
| Program exits on "quit" | 10 |
| EC #1: Continue until 10 tries (if no  correct guess and no "bye"), display remaining tries/guesses, display secret when game ends without a correct guess | 2 |
| EC #2: User can input max value for secret range,  Max tries is set to log[base2] of max value, remaining tries, game end when guesses exhausted, and display of secret after guesses exhausted behaves as for EC #1. | 2 |
| Total | 104 |

Extra Credit

1. [+2%] Limit the user to 10 guesses, and display the number of guesses remaining after each incorrect guess. Do not deduct a guess for invalid input. If the 10th guess is wrong, tell the user they have lost the game and display the secret. (2 points)
2. [+2%] When the game starts, have the user enter the max value, which can be any integer greater than zero. Then set the max number of guesses to be the (log[base2] of the max) + 1 (rounded to  the nearest int)  that they entered, and count down from that number as described in extra credit #1. [Fun fact—there is an optimal strategy that will guarantee a correct answer in log[base2] + 1 tries. It’s called [binary search (Links to an external site.)](https://www.topcoder.com/community/competitive-programming/tutorials/binary-search) (just read the first bit to get the idea—no need to read the proof unless that’s your thing!)]

Here’s how to get the log[base2] of a number num in python:

math.log(num, 2) # the second arg is the log's base