Lab 3: Math Quiz Lab

Lauryn Martin and Joshua Pollock

Lab Section 1

September 27, 2016

**Problem Statement:**

For this lab we are given a piece of code. From this code we need to add on a few constraints that are given to us. The constraints include:

* A selectable amount of problems by user input
* A selectable difficulty that works no matter the case the user gives (upper or lower)
* Increasing range of numbers as difficulty increases
* Print functions based on how many problems the user got correct.

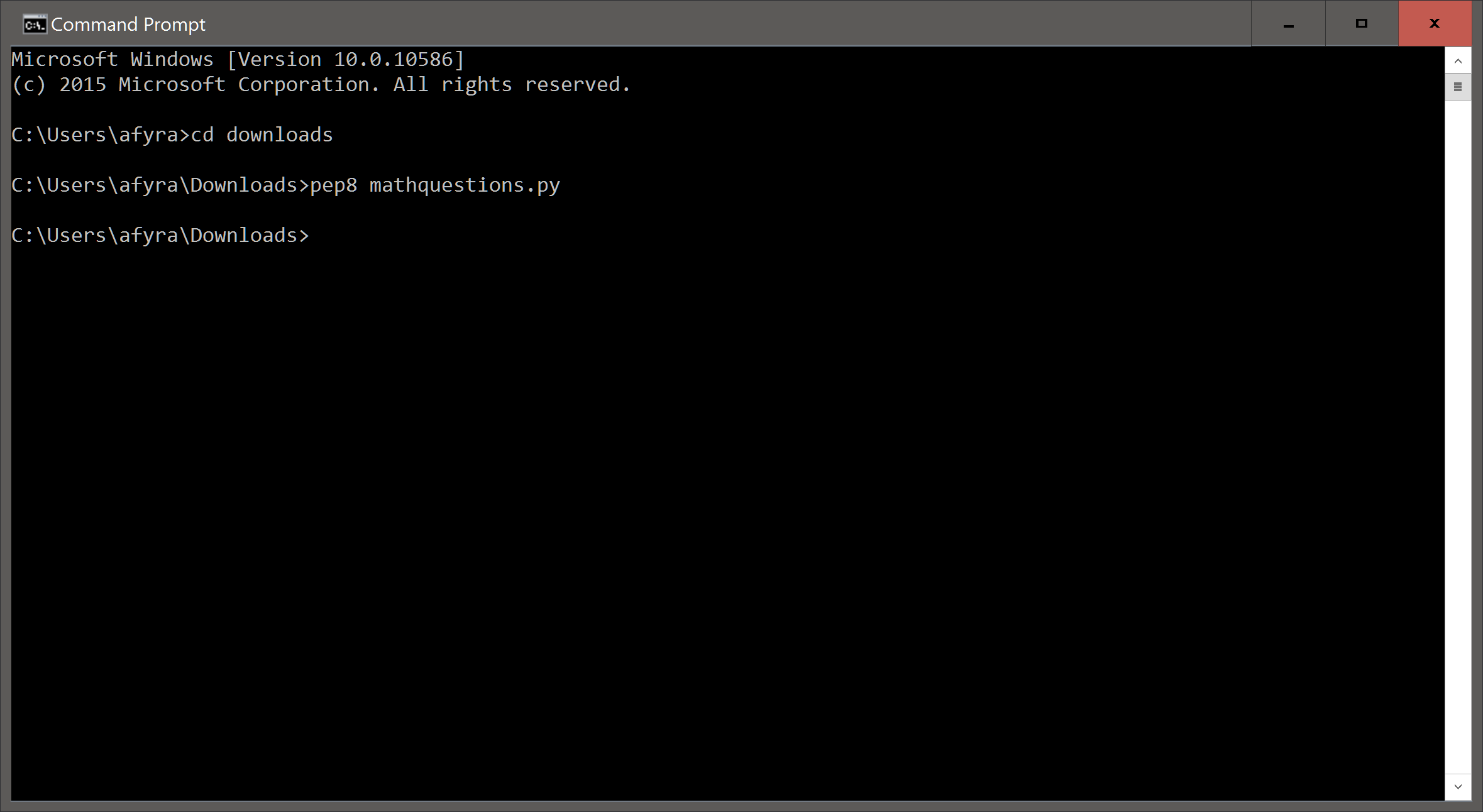
An important feature of this code is that we are allowed to use the randint function from importing random. This will make it easier to randomize each problem that the user receives.

**Planning:**

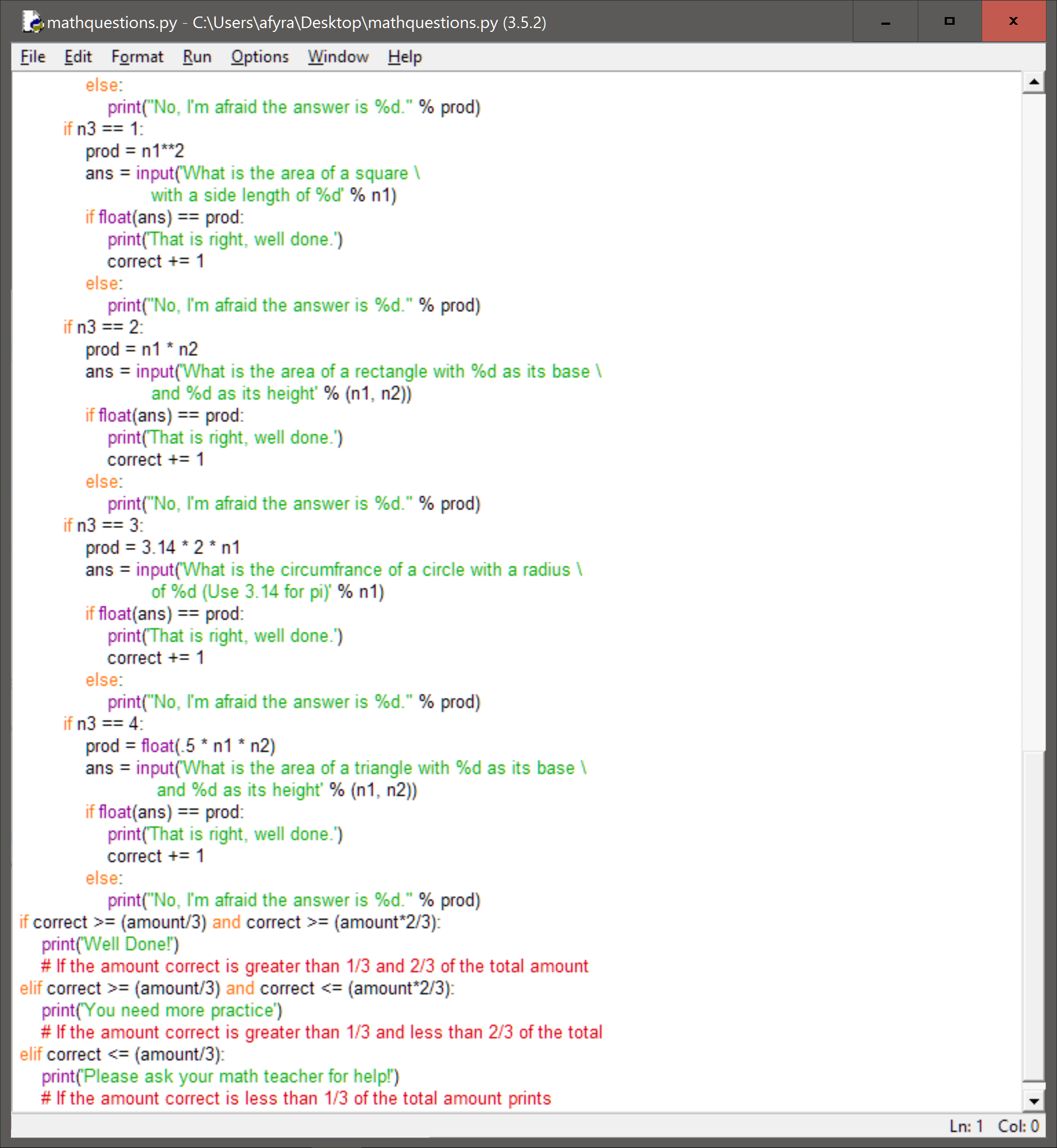
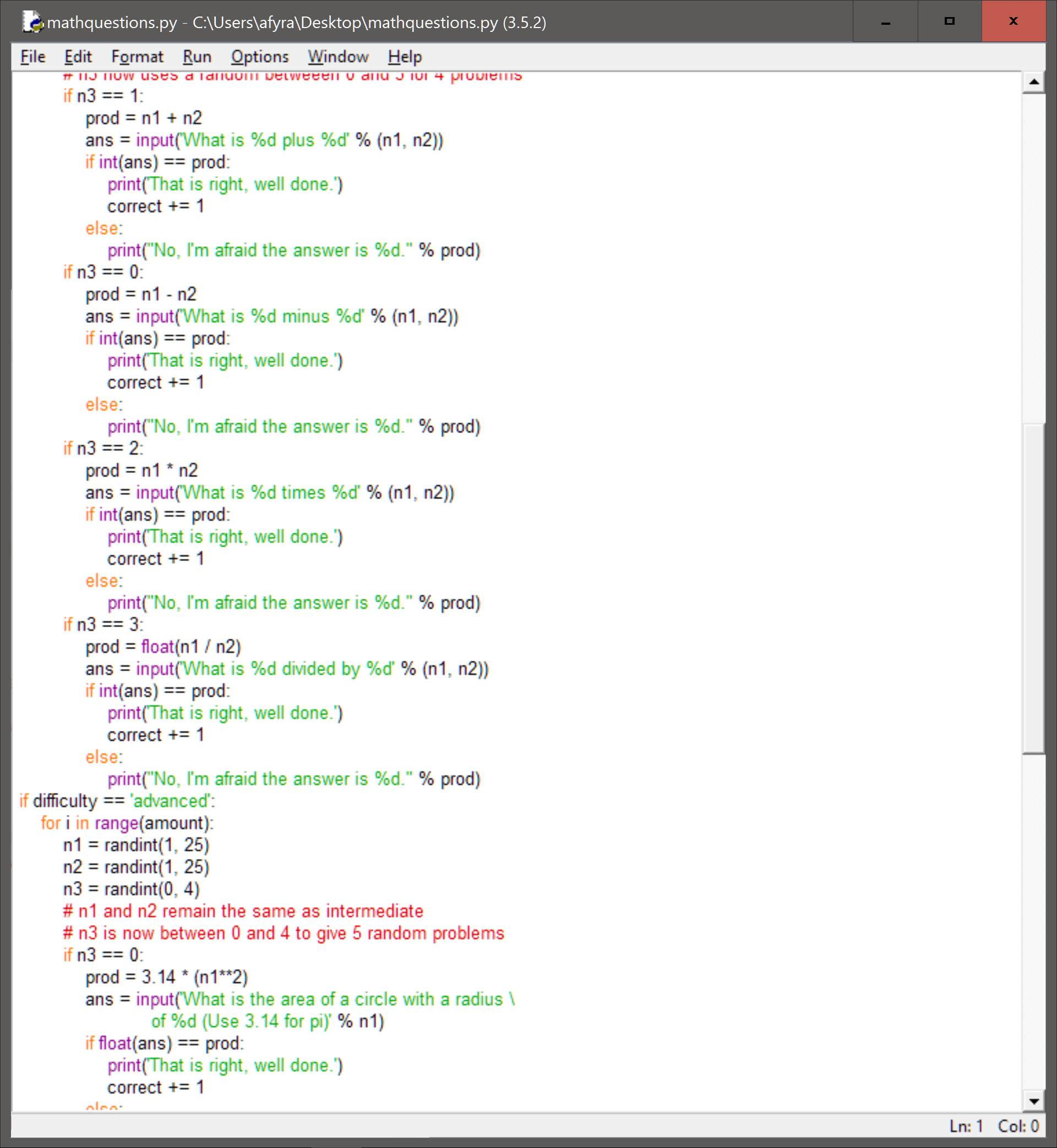
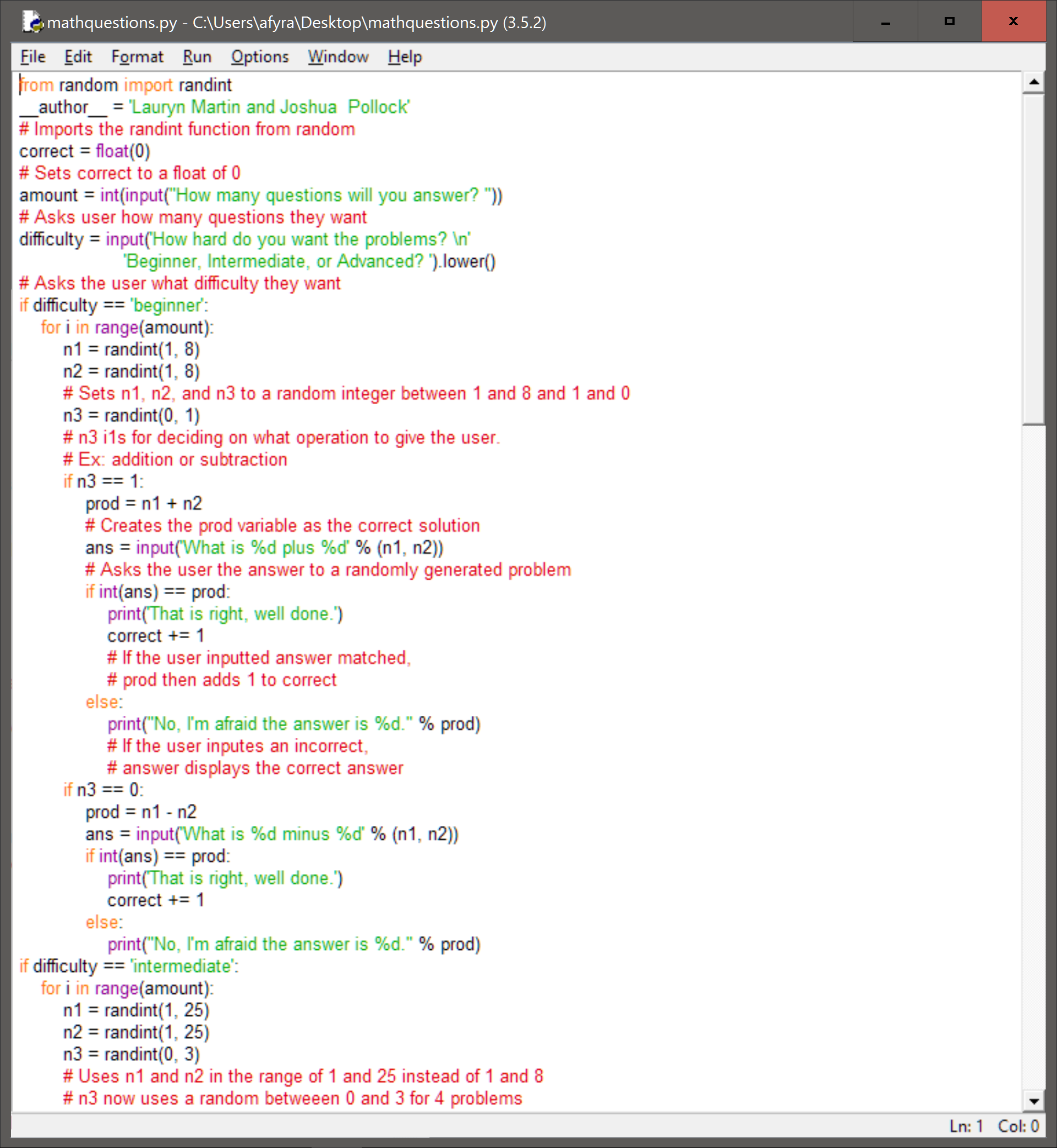
We started off by typing out the given code into IDLE. From this we were able to get a gist on what the code is doing and what needs to be changed in order to meet the requirements. We also needed to come up with 5 problems to input for the Advanced difficulty. These questions were on our pre-lab and were easy to come up with. For this code we want to use the randiint function and if statements in order to come up with the questions that would be given to the user. We will also need to use print functions and float functions in order to fill the constraints, such as the constraint of printing certain strings based on the amount of answers correct.

**Implementation and Testing:**

While testing the code we found that we would need to create a third variable in order to make the program truly random. We created the variable n3 with the intention that it would act as the randomizer. This meant that n3 would ocilate in between the amount of different problems and select one at random. We decided to change from using math.pi to using 3.14. This would make it easier on everday users and would make it unneccesary to use the round command.

The code gave out errors on the Advanced and Intermediate level of difficulty due to the introduction of decimals. We had to change many numbers to floats instead of integers in order to make the code run properly. This also applied to the end numbers as 2/3 and 1/3 create decimals. Once changing everything to floats the code ran properly.

While scanning the code with pep8 originally there were a large ammount of errors in the code. These errors were simple formatting errors and were quickly corrected. The other issue with out code was that some lines were too long. This was corrected by splitting up the lines with \. After these corrections the code was found to be pep8 compliant.



**Reflection:**

This code is quite long, which would make it harder to refactor. The ending statements use only if statements. This may cause errors in the code. For our testing it did not show any errors and displayed properly. Overall, the code was quite easy to come up with. Creating the code just took longer than previous labs. Using n3 to randomize the problems given was an easy way to randomize the code and follow given constraints. We were able to fill the constraints by using the randint function, if statements, and basic algebra functions. The most important aspects of our code was our if statements, our randint statements, and our input statements. We also used the float statement so the user of our code would be able to enter a decimal without issue. Our if statements were the most important in our code because we utilized it the most in our code. The randint made our code meet the requirements of the lab. The input statements helped us better organize our code and allowed the user to have some options when moving through our code. The way we did our code, while long, was the best and most understandable way to do it with the information and lessons we had to work with.