**Goal**

*This section should describe, in your own words, the goal of your project. What are you trying to accomplish? This section can include things you wanted to do but weren’t able to complete.*

The premise of my project was inspired both by my interest in education technology (I am currently an education concentrator but I am also looking into double concentrating in the computer science design track) as well as my 8 year-old-brother who seems to be struggling with his basic arithmetic skills. I thought I would design a simple program for him to enhance his math ability, targeting his weak points. Although it does little beyond helping with basic arithmetic, it was overall a stimulating project that made me excited to learn more about the underlying mechanics of education technology.

**Implementation**

*This section should describe the actual implementation of your project. What design decisions did you make? What data structures did you use? What functions, classes, and methods did you develop?*

The way the program works is that it will administer a “diagnostic test” to assess the user’s current math ability. The diagnostic test consists of an equal number of addition, subtraction, multiplication, and division problems. The user’s ability is assessed by evaluating the user’s accuracy in answering these problems by taking in the user’s answer to each problem as an input, and then comparing that input to the math problem’s answer. Herein lies my first design decision: I created a math problem class with four properties: a first number, a second number, an operation, and an answer. Every diagnostic test was created with a random\_problem\_generator method that called the init method for the math problem class along with a random\_number\_generator function that generated numbers between 0 to 20. The test\_administrator method took in a problem set (a list of math problem objects) and called the print\_problem method on each element of the list.

After the user answers every problem in the diagnostic test, the program will calculate a total score using the number of problems that were correct and the total number of problems (the length of the problem set). The program will also store the number of incorrect answers for each operation in a dictionary. This operation\_incorrect\_dictionary maps every operation to an incorrect count. For example, every time the user gets a multiplication question wrong, the value mapped to the multiplication operation in the operation\_incorrect\_dictionary gets incremented by one. At the end of the diagnostic test, this dictionary is passed to a personalized\_test\_generator function. This function uses the dictionary to design a specially crafted personalized test based on the types of problems that the user gets wrong on the diagnostic test; for every incorrect division problem on the diagnostic test, the program will add a division problem to the personalized math test that follows. After the user completes the personalized test, the program ends.

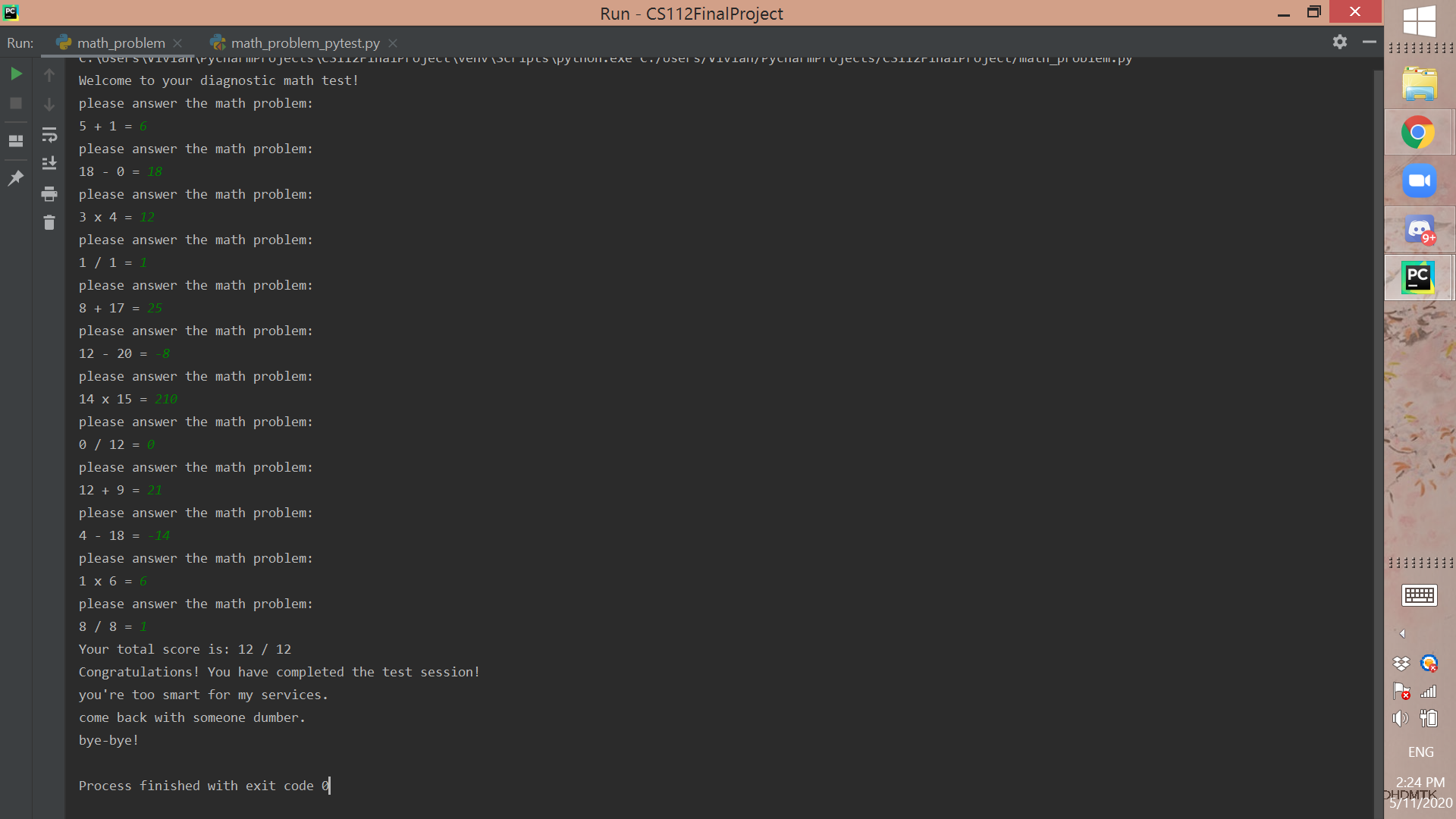
**Results**

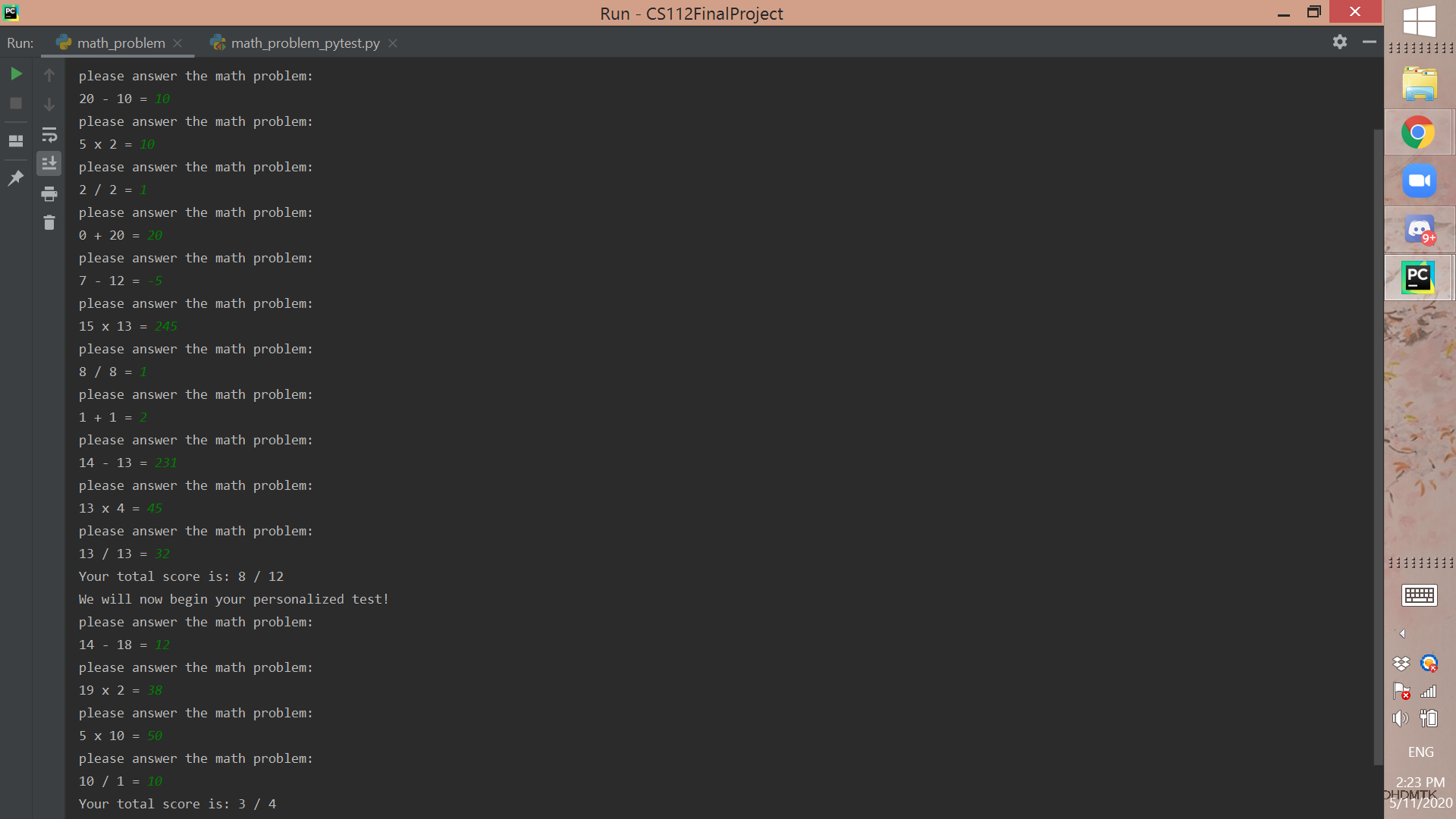
*This section should describe the results of your hard work. Include any measurements you did, example program inputs and outputs, or screenshots. This is also the place to tell us about testing: how did you ensure that your code is correct?*

One aspect of my program that was difficult to test was the test\_administrator method because it required an actual user to interact with the program in order to function. As testing a method with user inputs is beyond the scope of this class and my present ability, I was, unfortunately, unable to test my test\_administrator method. However, there are still potential errors that I wished to account for even if I couldn’t test them explicitly. For example, I wanted every answer to be an integer because oftentimes the program would deem an answer incorrect because the float wasn’t the same value as the integer. Hence, I tweaked the init method a little bit such that the answer to every division problem would always be a whole number. Although I wasn’t able to test this feature using the test\_administrator, I was still able to test it within my init method test. I also added a feature to the test\_administrator that will end the program (with a kind message!) should the user get a 100% on the diagnostic test.

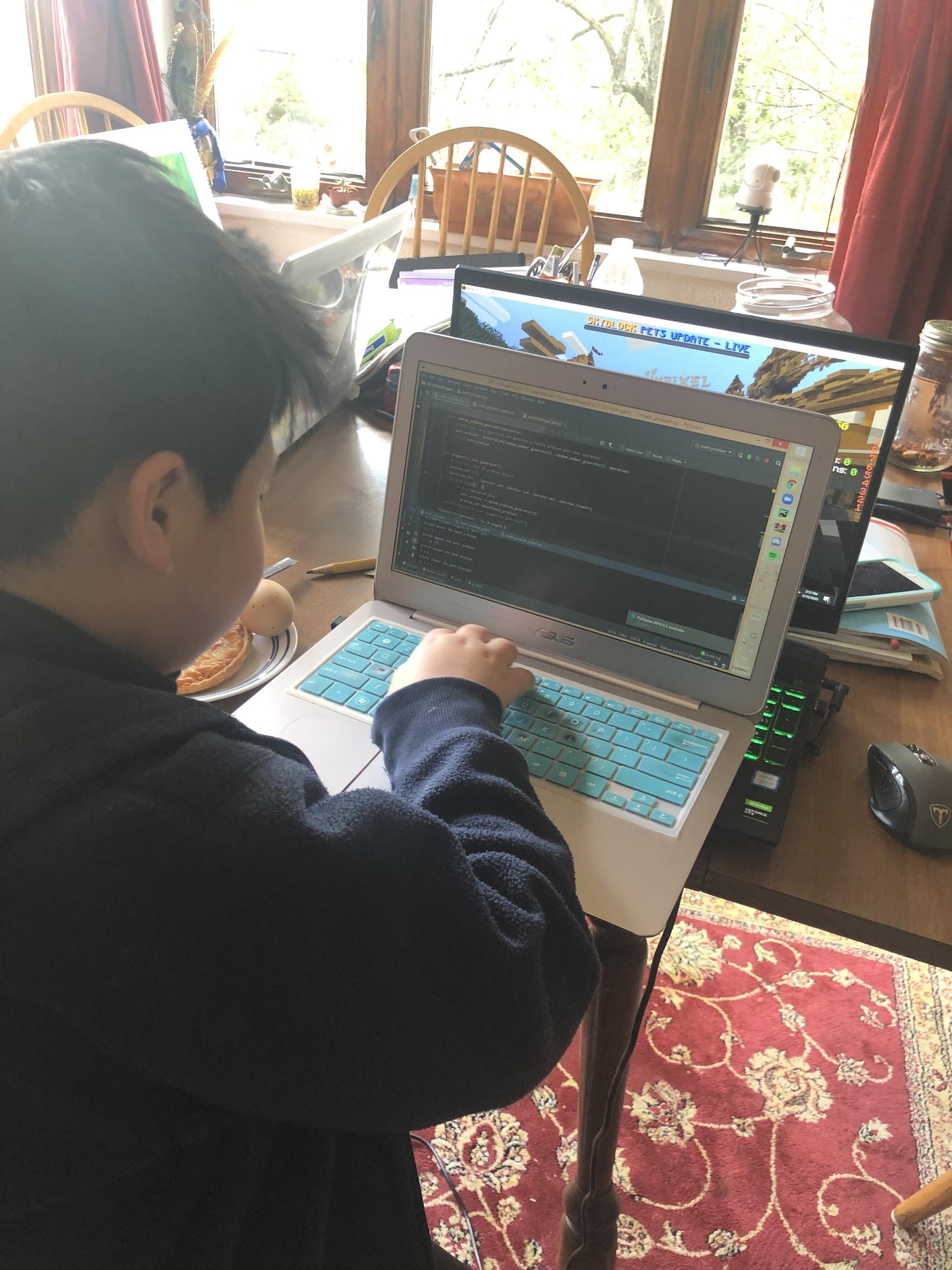
I tested the other aspects of my code in several ways. The most distinctive test cases were in my init method for my math problem class because I had to watch out for innate math technicalities such as dividing by zero. I also tested the case in which the operation passed into the init method wasn’t actually a math operation (it throws an error). For my diagnostic\_test\_generator, I checked to make sure that it had an equal number of addition, subtraction, multiplication, and division problems. In contrast, for my personalized\_test\_generator, I tested it to make sure that it had a different number of addition, subtraction, multiplication, and division problems based on the operation\_incorrect\_dictionary: the contents of the personalized test should mirror the operation\_incorrect\_dictionary exactly.

Below are screenshots of user inputs and how the code responds to those inputs:





Here is also a photo of my brother using the program:



**Challenges**

*Describe any challenges you faced. What was difficult? Did you have any false starts in implementing your project? If you had goals that you weren’t able to accomplish, tell us why!*

One challenge that I had, in the beginning, was figuring out how to use python’s built-in math functions as “objects.” Prior to this project, I only knew how to call the math functions for adding, subtracting, multiplying, and dividing with +, -, \*, and / respectively. However, since I would be using them as an object property as well as an input for different functions, I needed a way to “access” these functions in a different manner. I asked professor Woos and he recommended that I use the package “operator” which would allow me to apply my math operations as though they were functions. For example, operator.add( 5, 4 ) returns 9. I was also able to pass these functions as arguments in my methods.

Another challenge I had was that the program would deem an answer wrong simply because it expected a float instead of an integer. To counter this issue, I just got rid of fractions as answers entirely (having the answer to the math problem 5/7 be 5/7 is anticlimatic anyways). My brother hasn’t worked much with fractions anyway so this change ended up serving his purposes better too.

**Future work**

*If you had as much time as you wanted to keep working on this project, what would you try to do? Be creative!*

If I had more time to work on my project, I would like to add more variety to my problems, such as problems with more than just two numbers. I was also considering adding a way for the user to “pre-personalize” their test by allowing them to control the length and contents of their tests. Perhaps they would like to have a longer personalized test for increased help. Or maybe they already know that they are really good at adding so they would want to remove the addition problems all together from the diagnostic test. Or maybe they have all operations with numbers below 20 down pat and would like to focus on arithmetic with bigger numbers. The possibilities are endless. I was also thinking about ways in which I could categorize problems by more than just their operation type. Perhaps I could tweak the program such that it would pick on the fact that the user has particular difficulty with numbers above 15, or that they have trouble multiplying numbers by 7. It would be interesting to see how I could make the personalized test more “personal.” Of course, the ultimate ideal would be to be able to move towards problems that aren’t just simple arithmetic but other more advanced problems or maybe even expanding to other subjects.

Overall, I had a lot of fun with the project and I am satisfied with what I created for my brother.