Justin Phillips

cs362 winter2020

assignment 4

2/14/2020

**Unit Testing**

I started off the unit testing by creating the test\_classroom function where we will initiate the values needed for the student function. The first test that was created, was to test if the \_\_init\_\_ created the correct values for the student, then I ran an assertEqual to verify that the values for the Student class are what they are supposed to be.

The next function that is tested is the get\_full\_name function, in this test I first created the student object. I then set the first\_name and last\_name to their own independent variables. I then called the get\_full\_name function with the student object as the parameters and set the results to a variable called fullname. Then ran an assertEqual comparing the fullname to the (fn, “ “, ln)

Test\_submit\_assignment was the next function that I tested. I started this off by creating a student object where it was set to the variable of student, then created an empty assignments list and appended “test” to it. I then called the get\_assignments function, also called the get\_assignment and set the results to name. After calling these I then called the submit\_assignment(name) then used the assertEqual and compared the length of the list to 2 checking that the assignment was added to the list.

Next test was on the get\_assignments, the first things that needed to be done was to create an object for student and assignment. I then added the assignment to the student object and ran an assertEqual comparing the assignment object was equal to the assignment that was adding in the get\_assignment function call.

The test\_get\_average function was probably the most difficult to create the test for, I started off by creating the student object and created 2 assignment objects. I then assigned grades to the objects, then added those grades together and then divided them get an average. The results were set that average to a new variable. I then appended the assignment to the assignment list, called the get\_average function set those results to a variable. I then called the assertEqual function to test if the results of the average variables were equal.

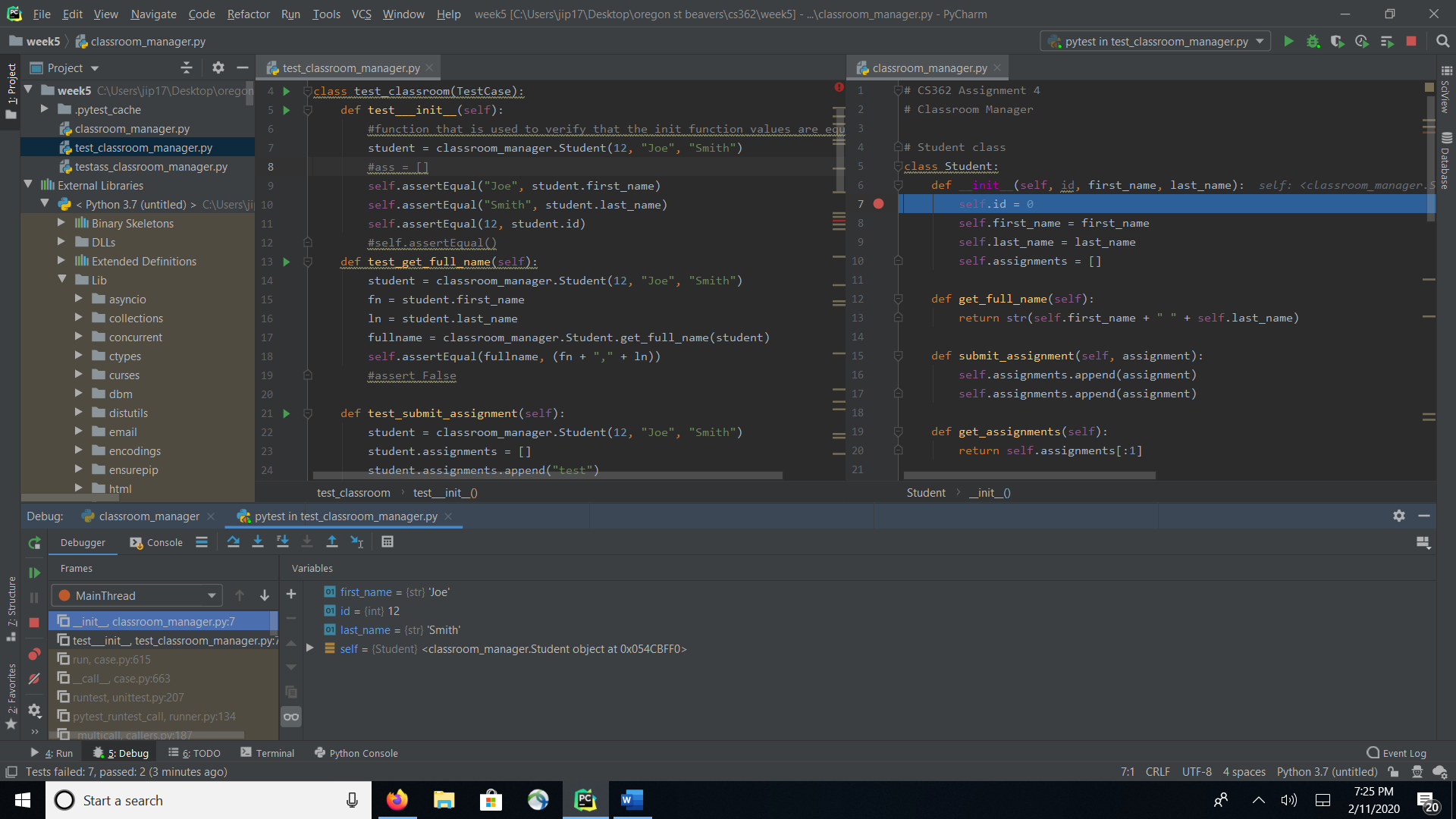
The last function for the student class is test\_remove\_assignment, first step was to create the student object followed by creating a list for the object. Next, I needed to create the assignment object to add to the list, appended that assignment to the list and then called the remove function. To check the results, I used assertEqual comparing the length of the list to 0.

The student functions are finished now I needed to test the assignment class. The first test was to test that the class is initiated properly, so to test this I created an assignment object and created a grade for this object. I then ran assertEqual on the items that were added to the object.

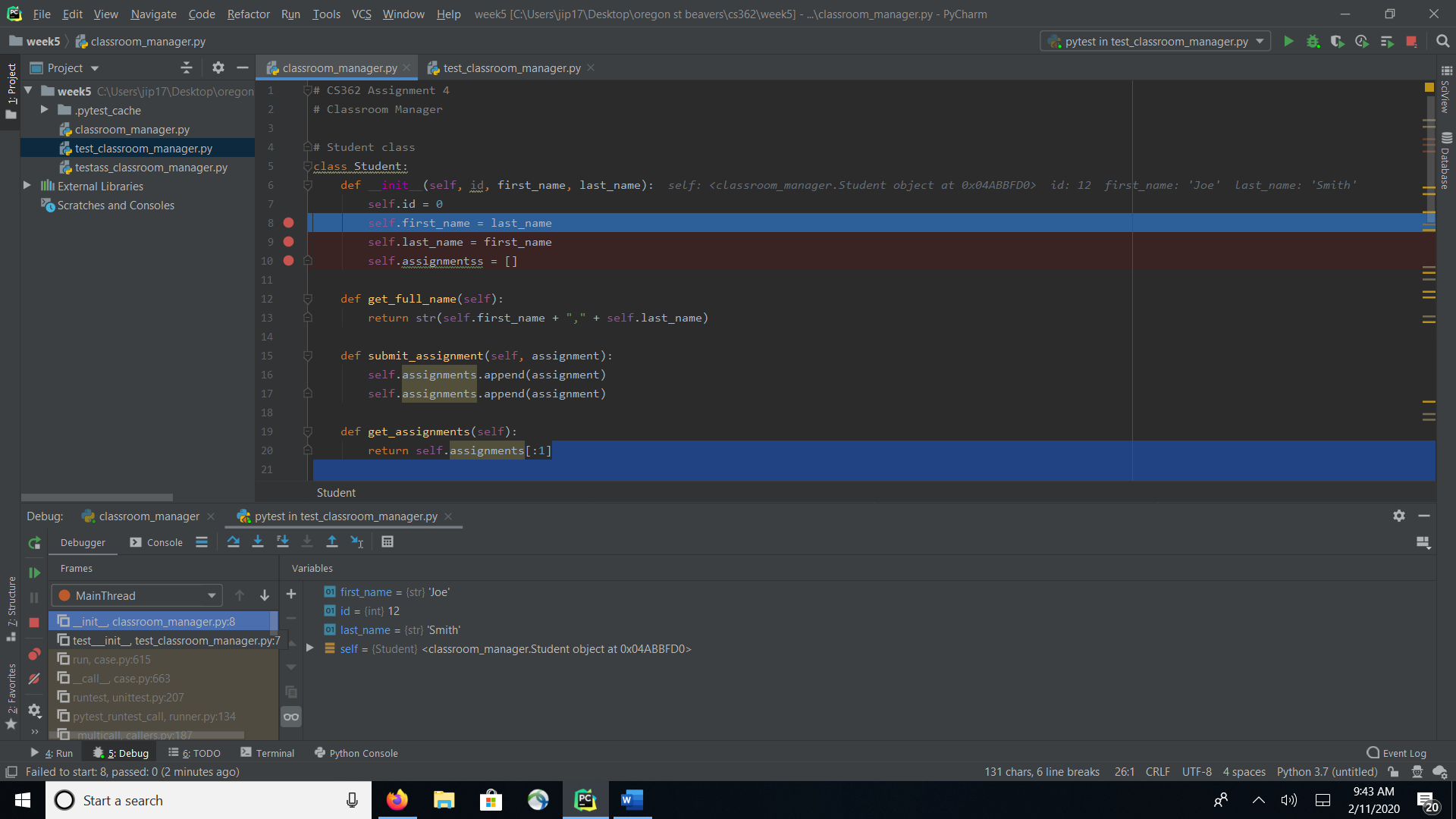
The last function that I created unit test for was test\_asign\_grade for the assignment class. I created the assignment object, then called the assign grade function. In this test case I had a boundary case since if the grade was over the max grade then the grade was none. The first assertEqual was just to check if the grade in the object was equal to the grade that got assigned. I then called assign\_grade again with a number greater than the max, called assertEqual and compared it what the results should be which is none.

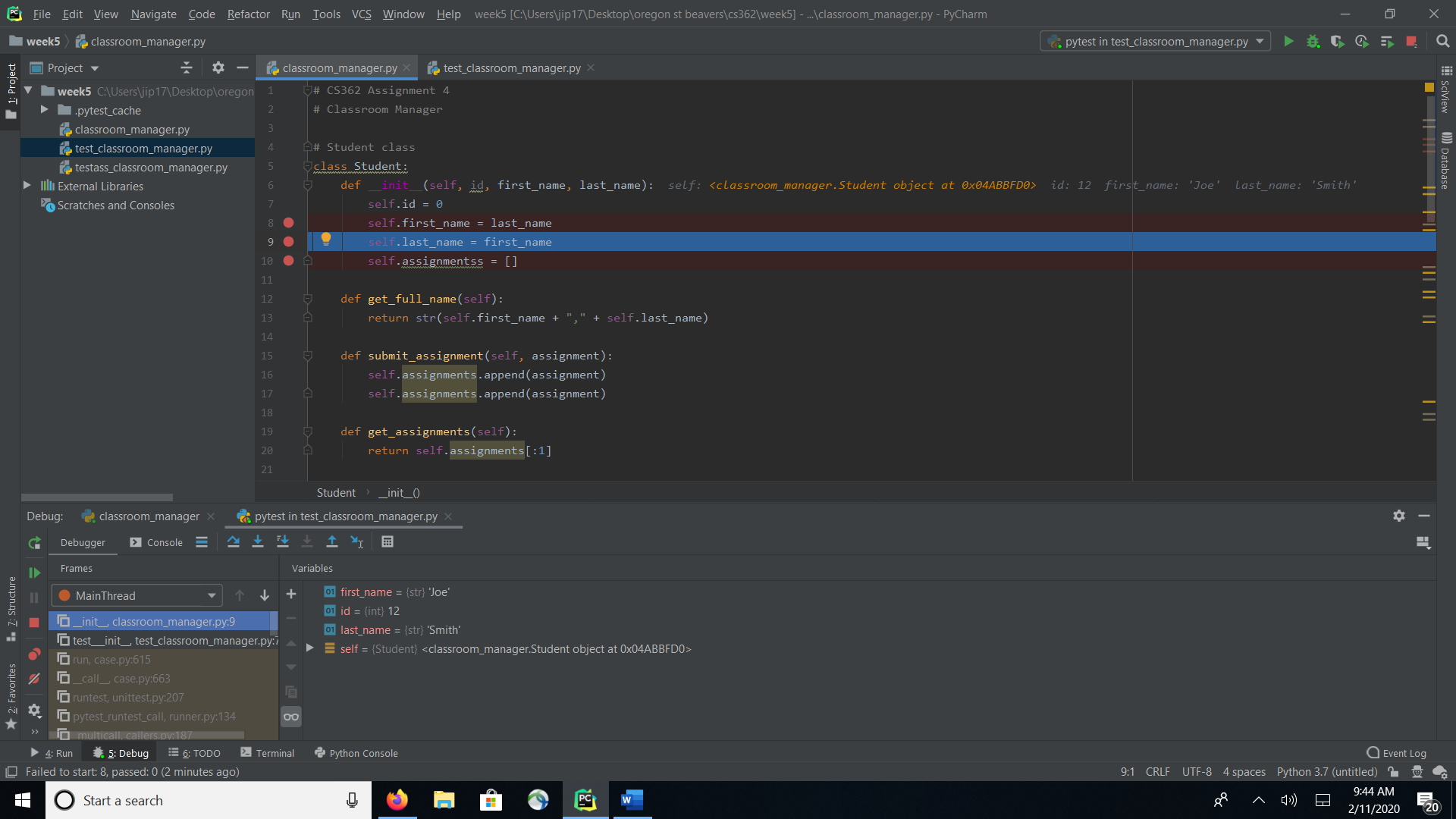
**Debugging**

The first bug I started with was in the \_\_inti\_\_ function the original code had id set = 0 and it should be self.id = id.



The next bug I found was also in \_\_init\_\_ where the arguments being passed were flipped and had been assigned to the wrong name. They had self.first\_name = last\_name and self.last\_name = first\_name.

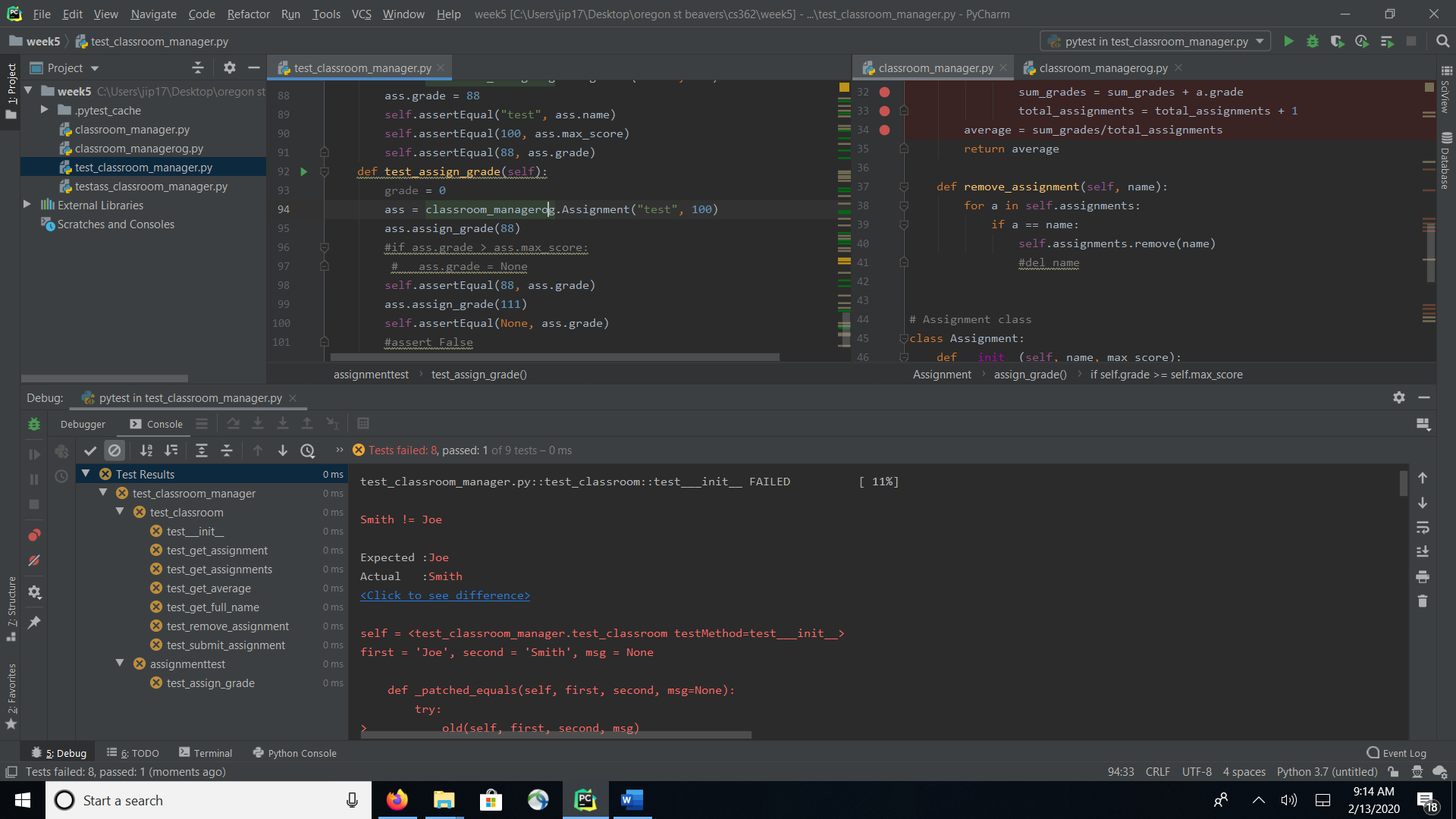




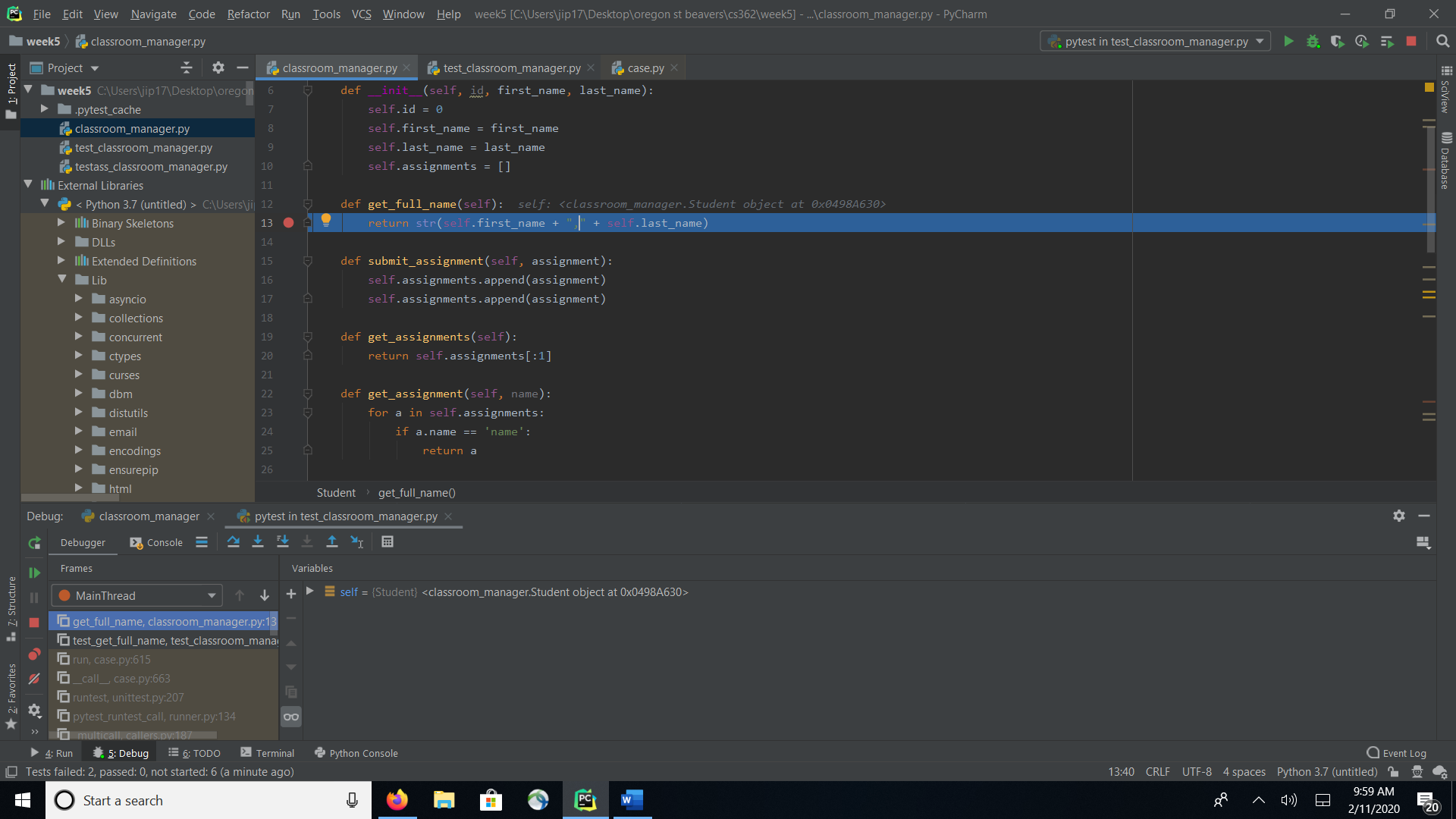
I found a bug in a misspelling of assignments; this was an easy find since PyCharm highlights variables when they aren’t being used. I tried to make an assignment list the word “assignment” was highlighted.

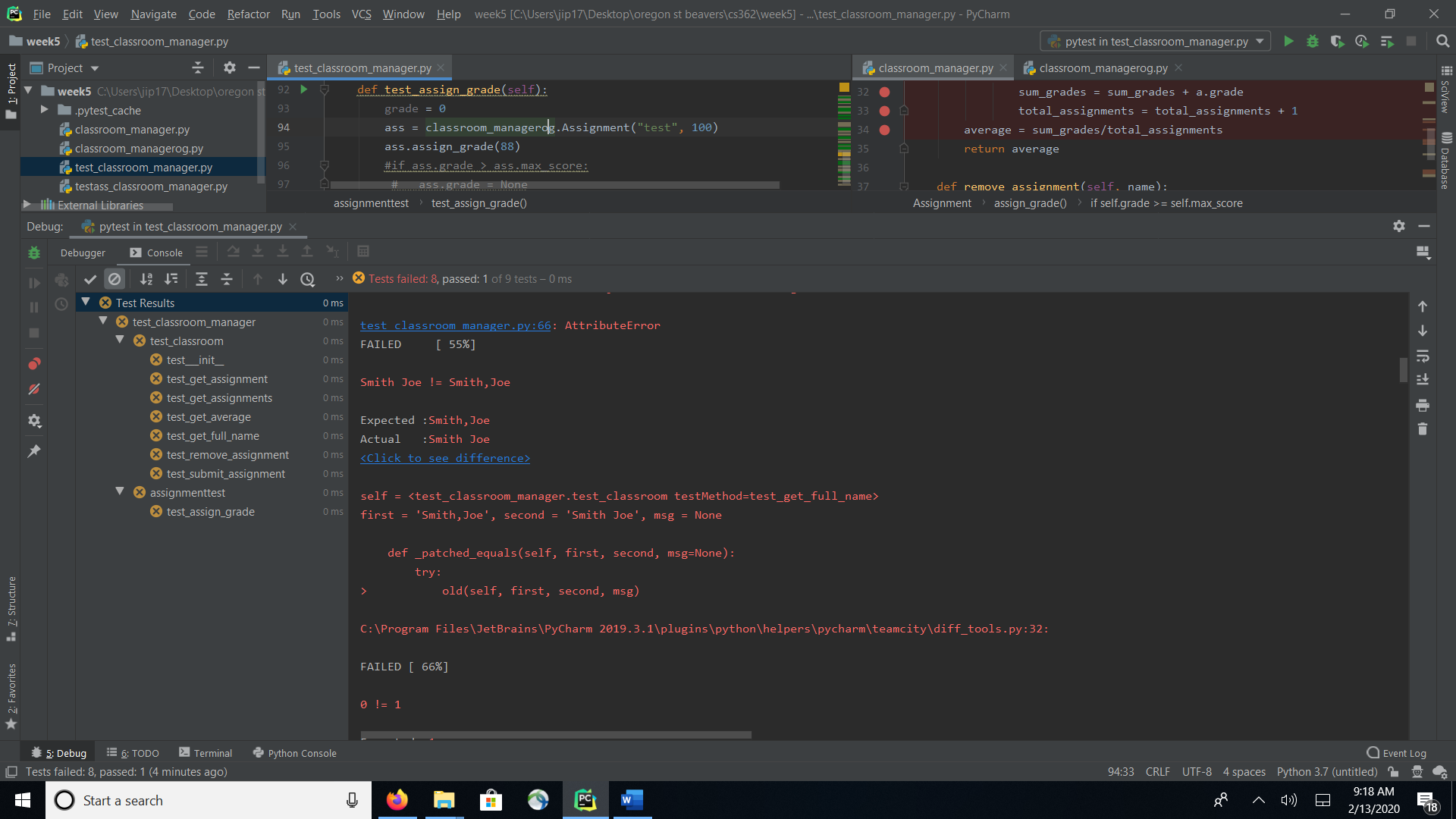


The failed test showed how the name was wrong which also gave a hint at the bug

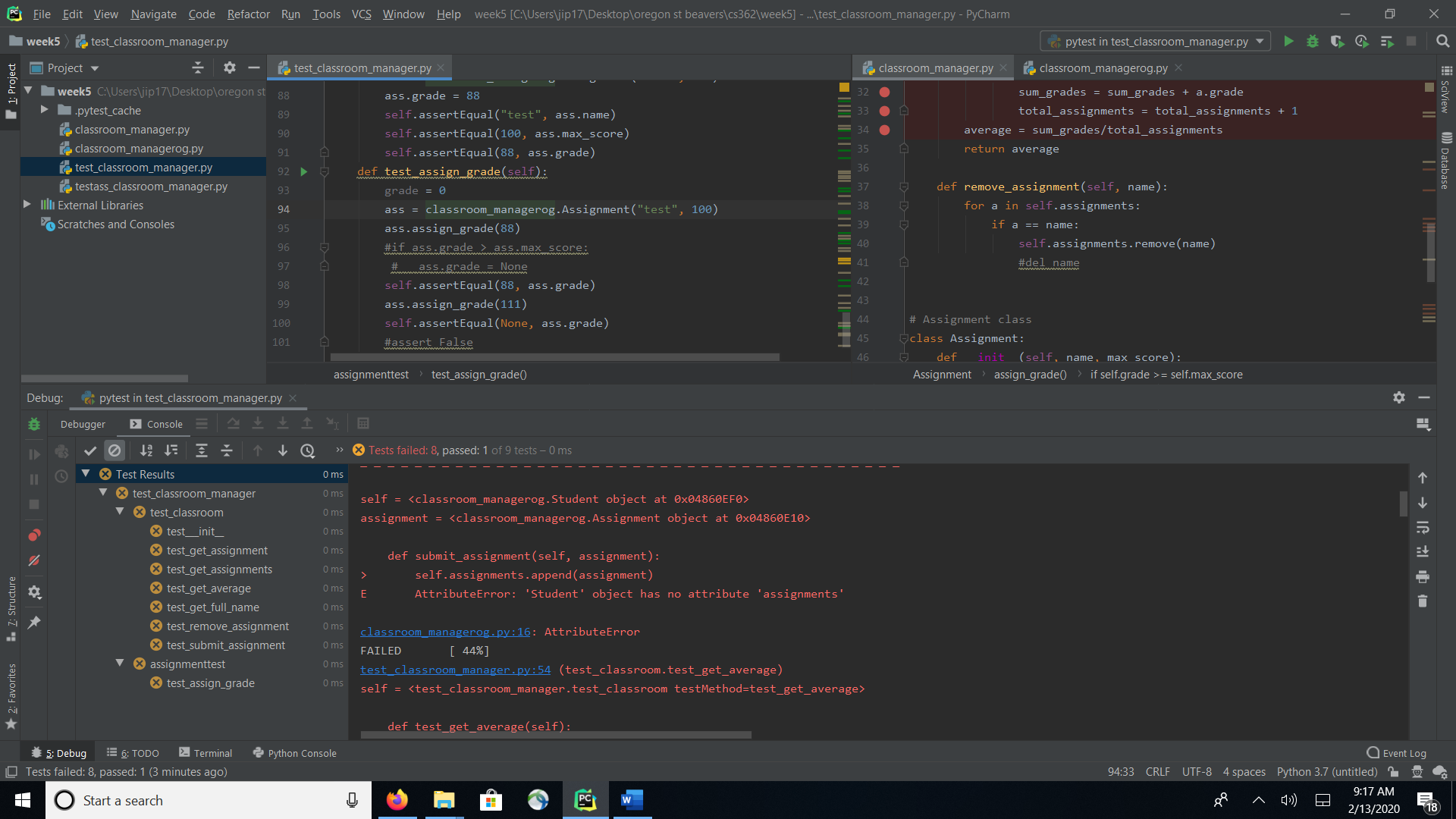


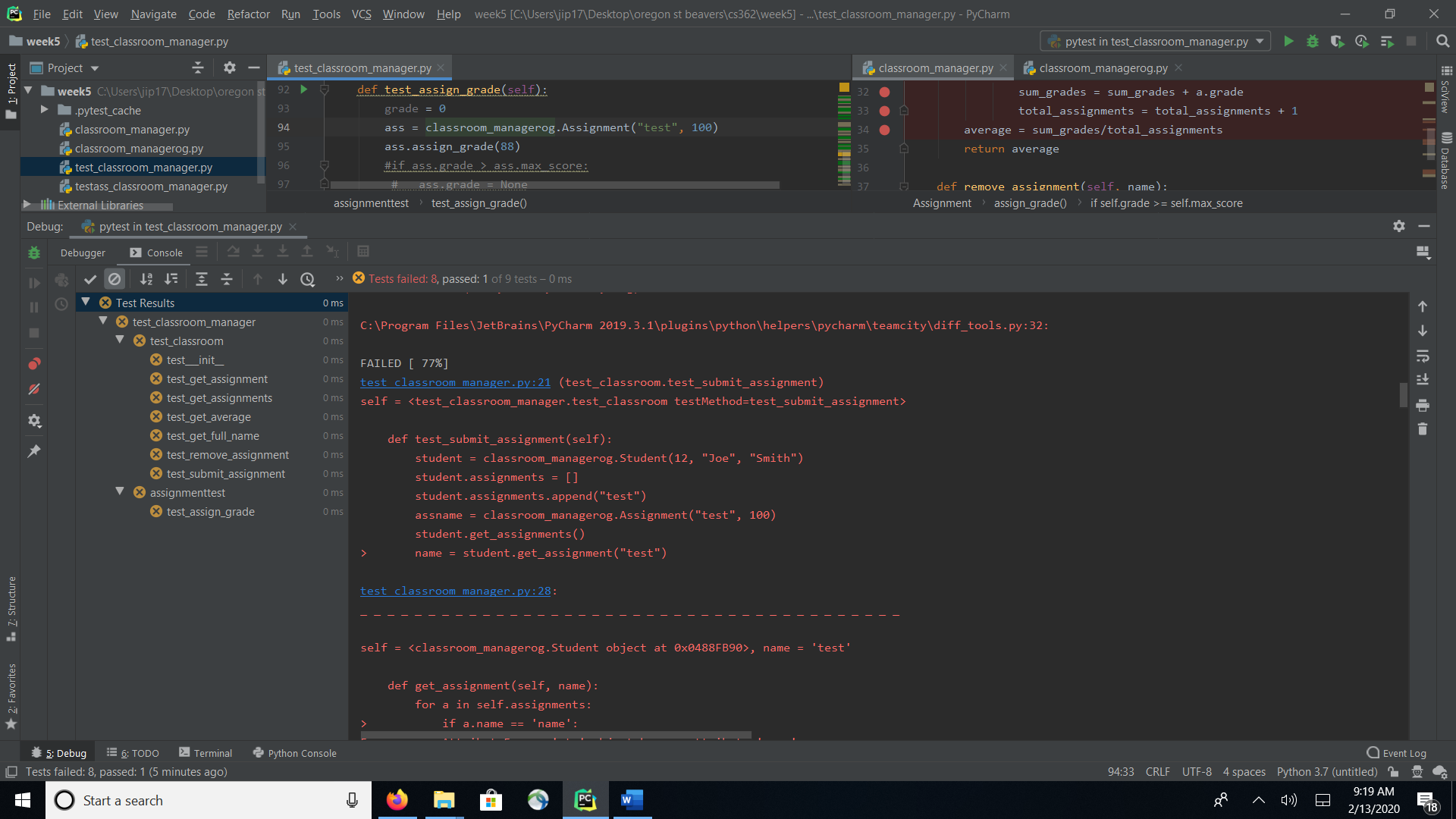
The next bug I discovered was in the get\_full\_name function in the instruction it calls for there to be a whitespace between the first and last names, but the function had a “,” so I just fixed it.



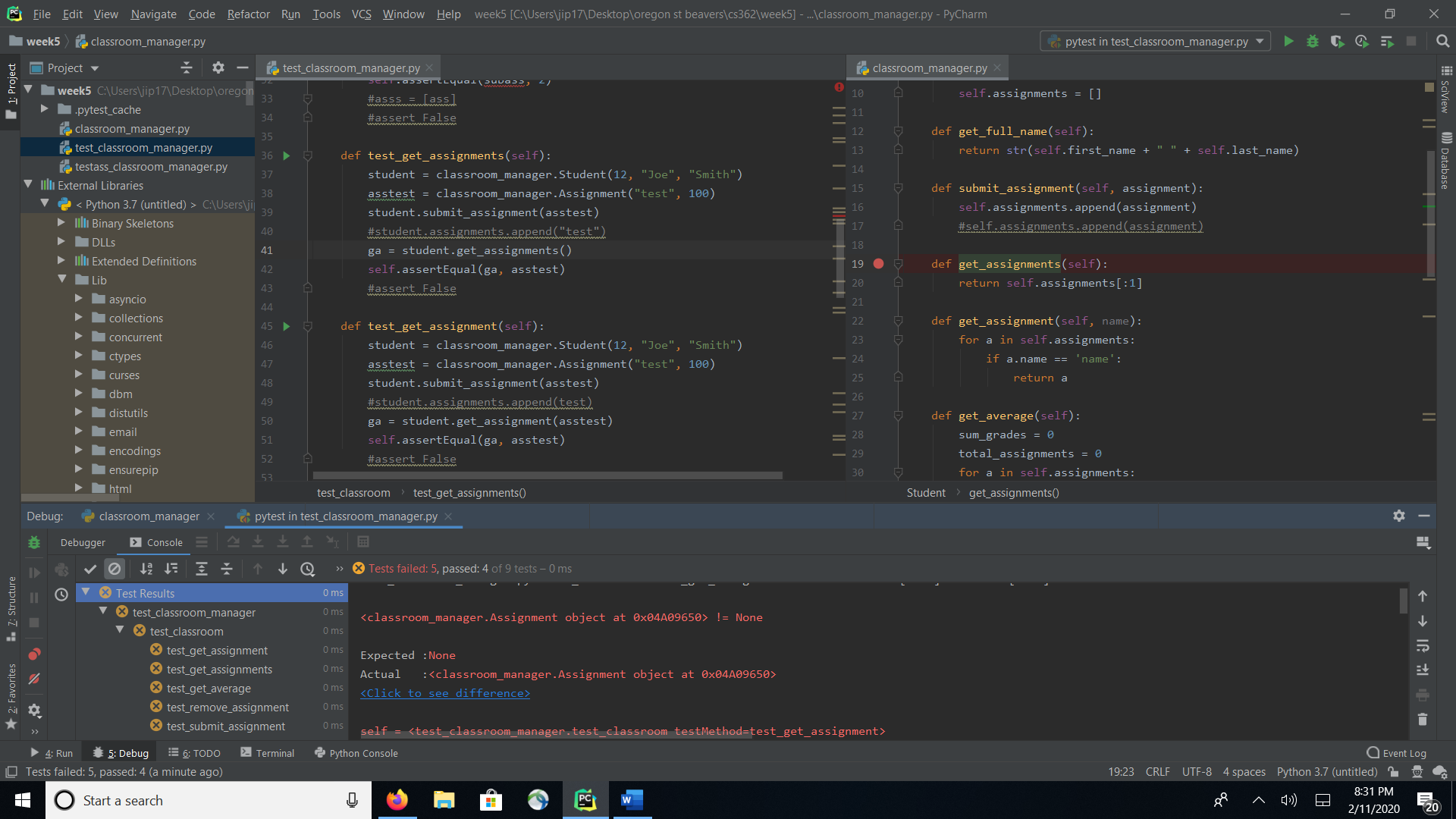


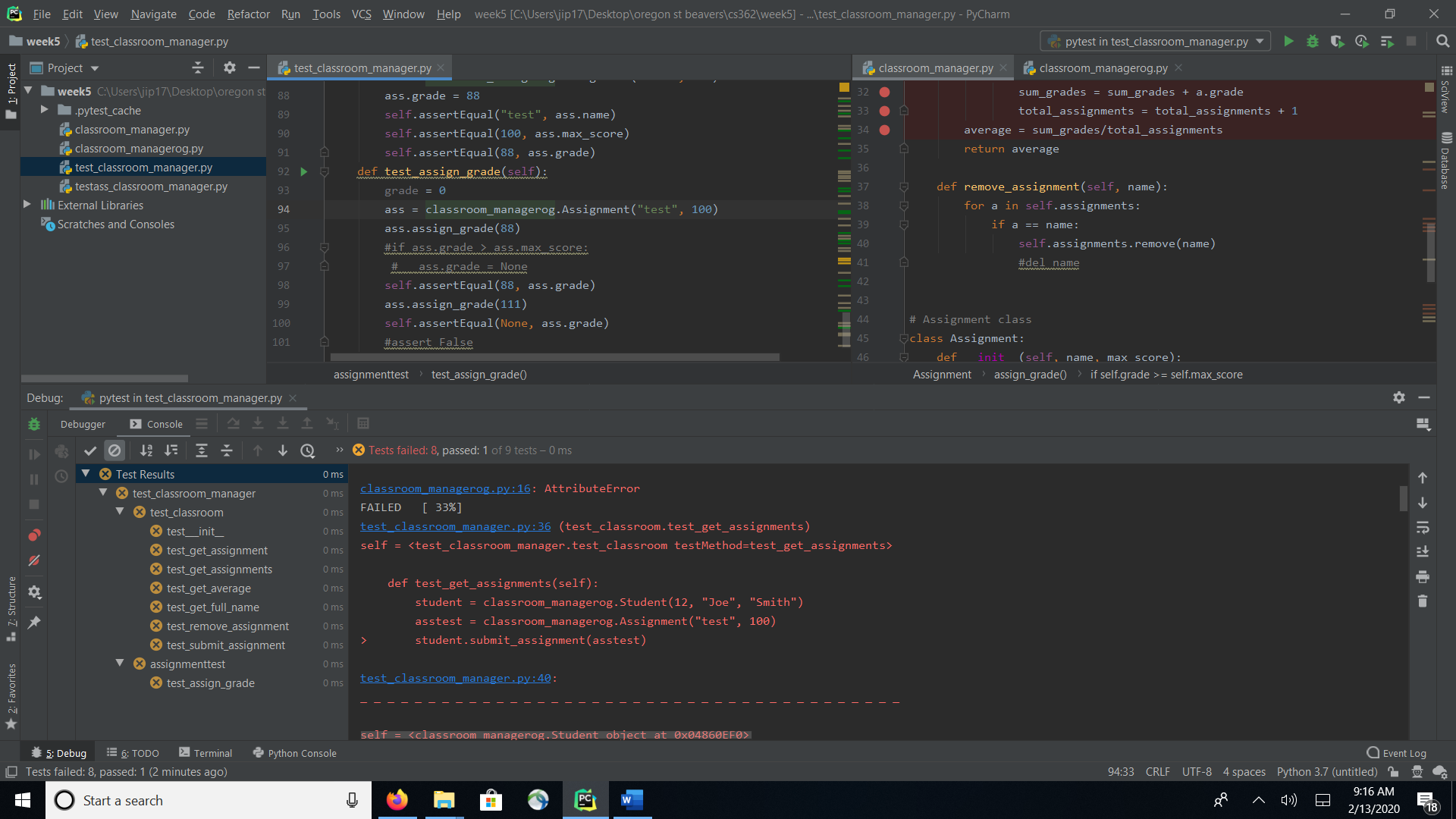
In test\_submit\_assignments it seemed that it submitted the assignment 2 times, so I deleted one of them.



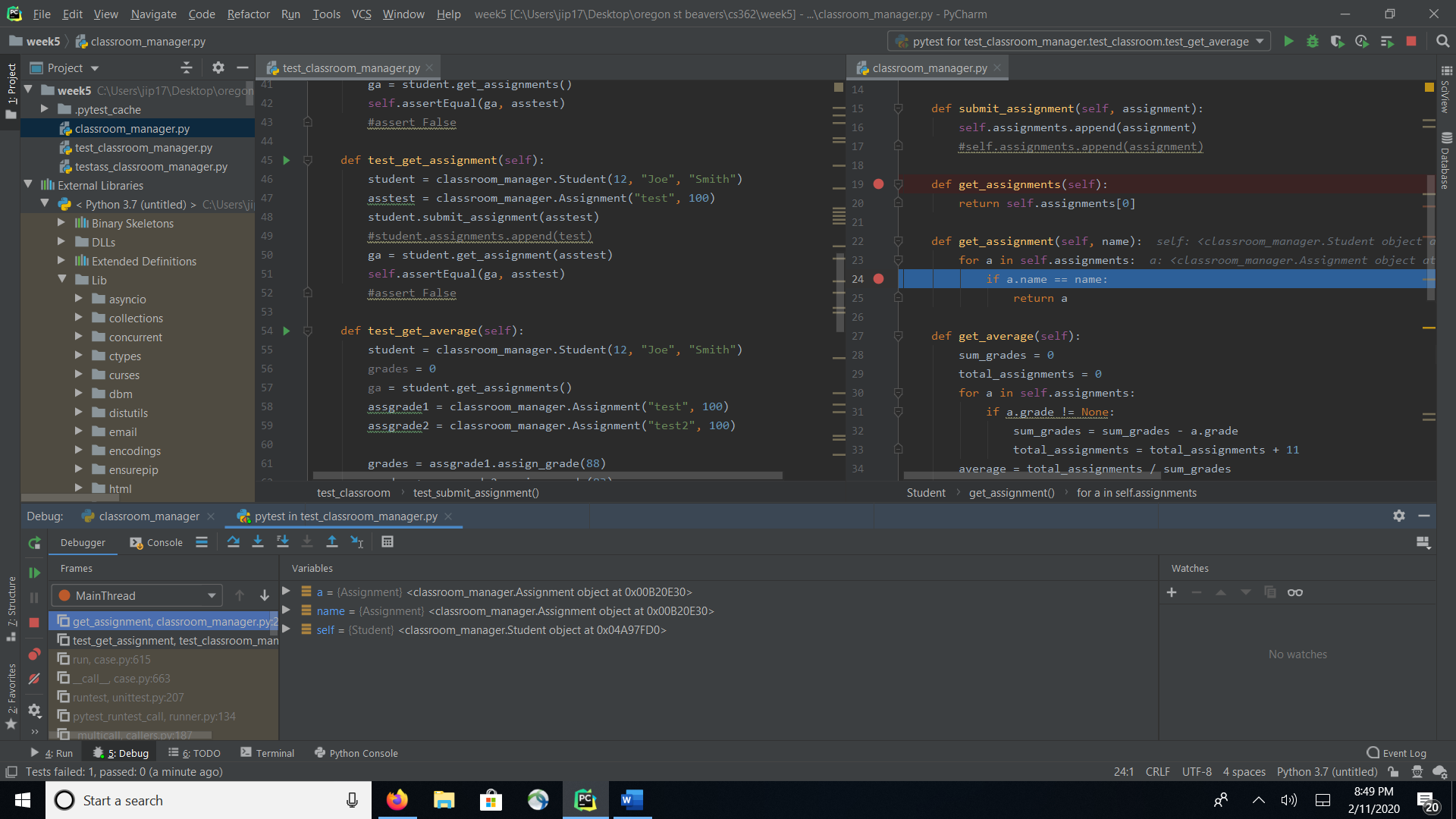


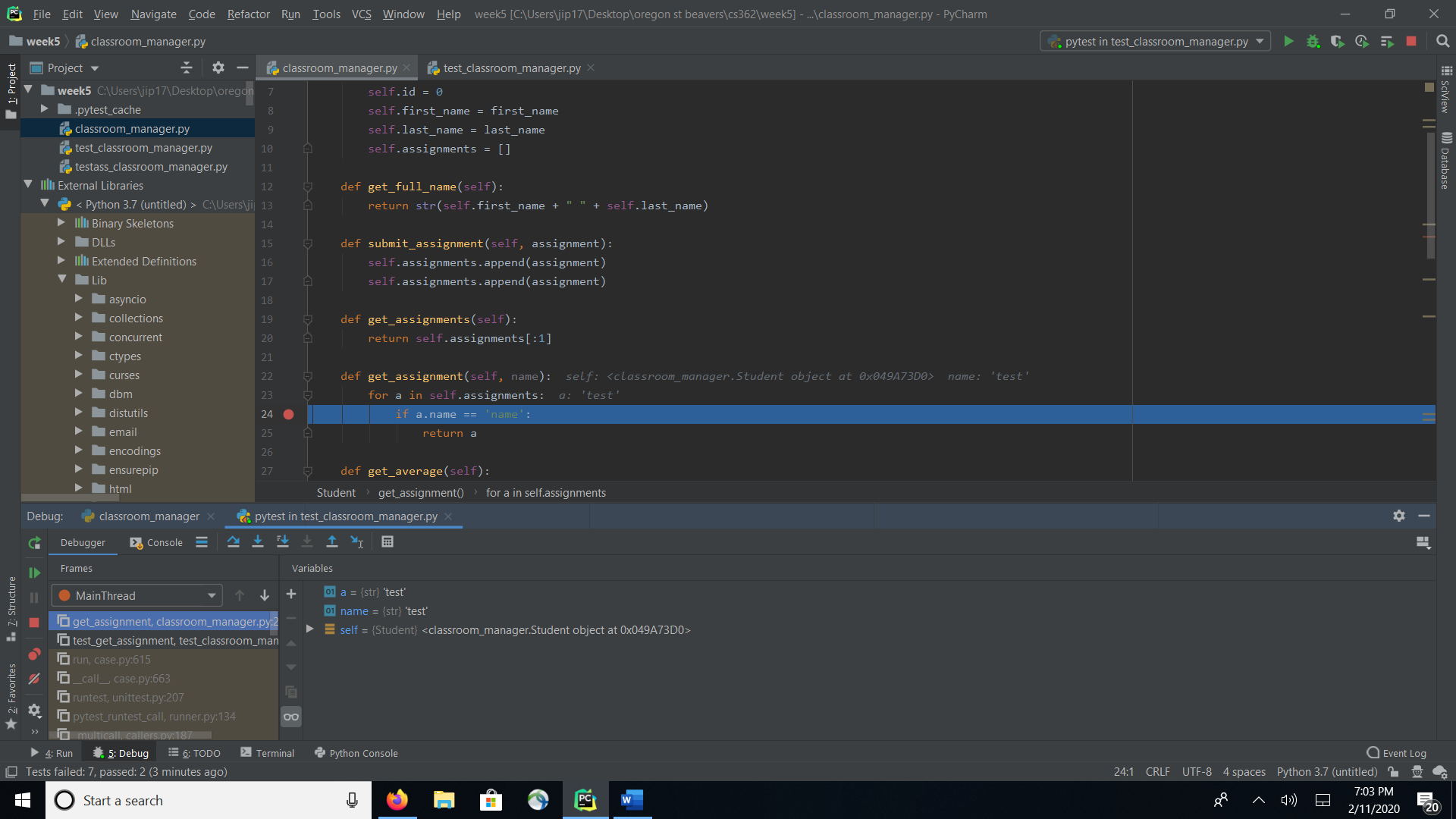
The bug I found here was that the return is get\_assignments was returning assignments [:1], I fixed this to return assignments [0].

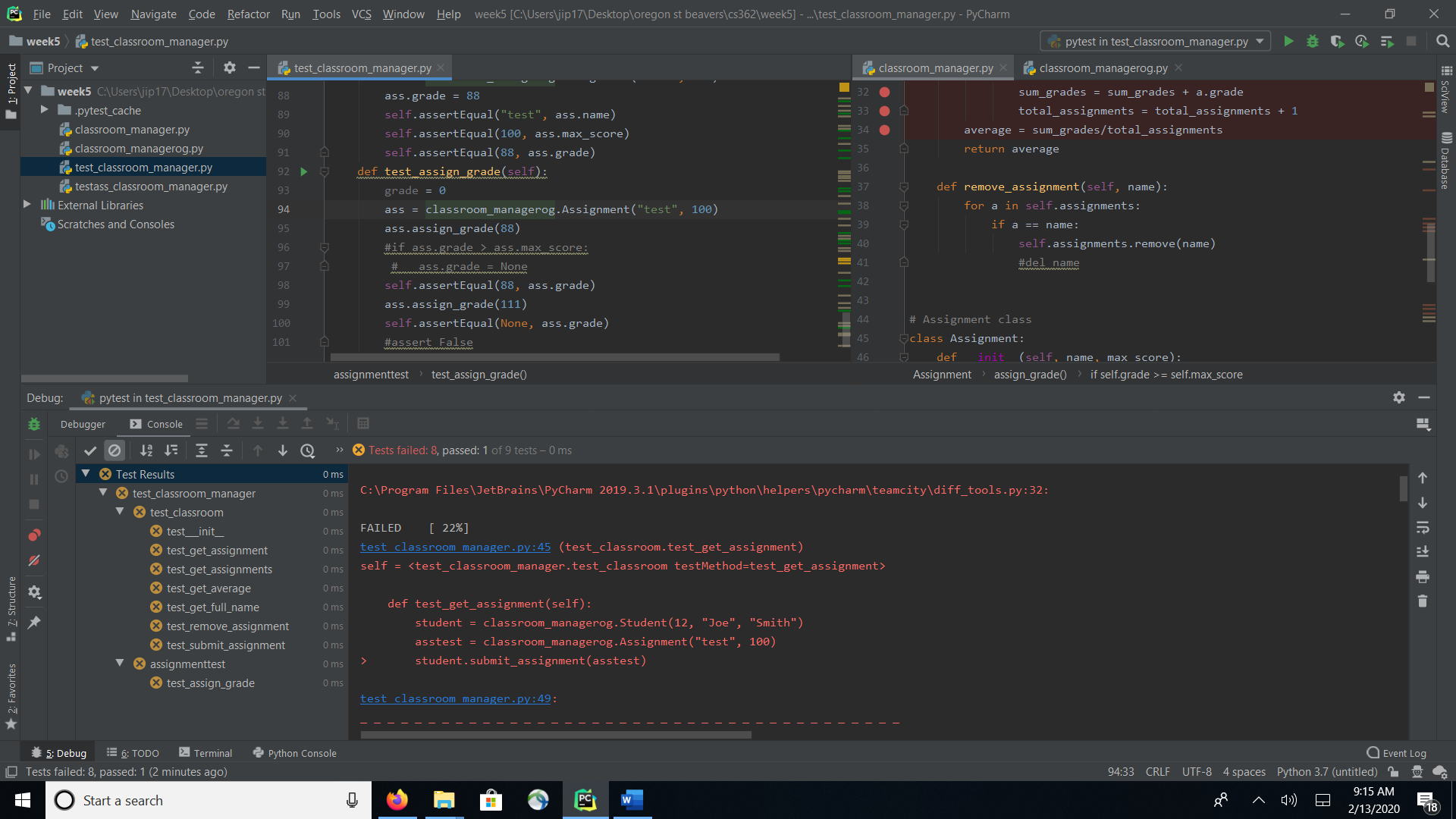


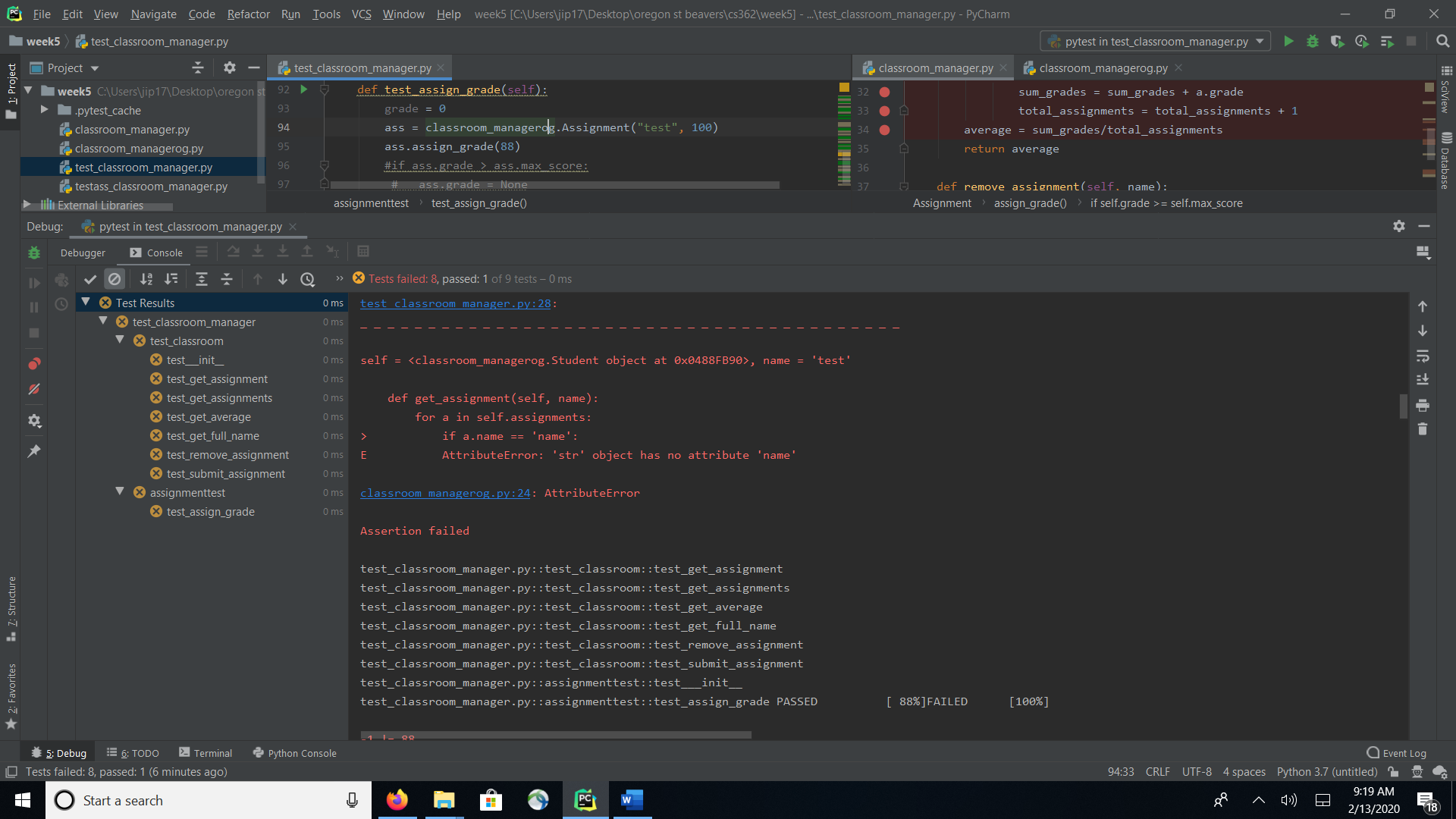


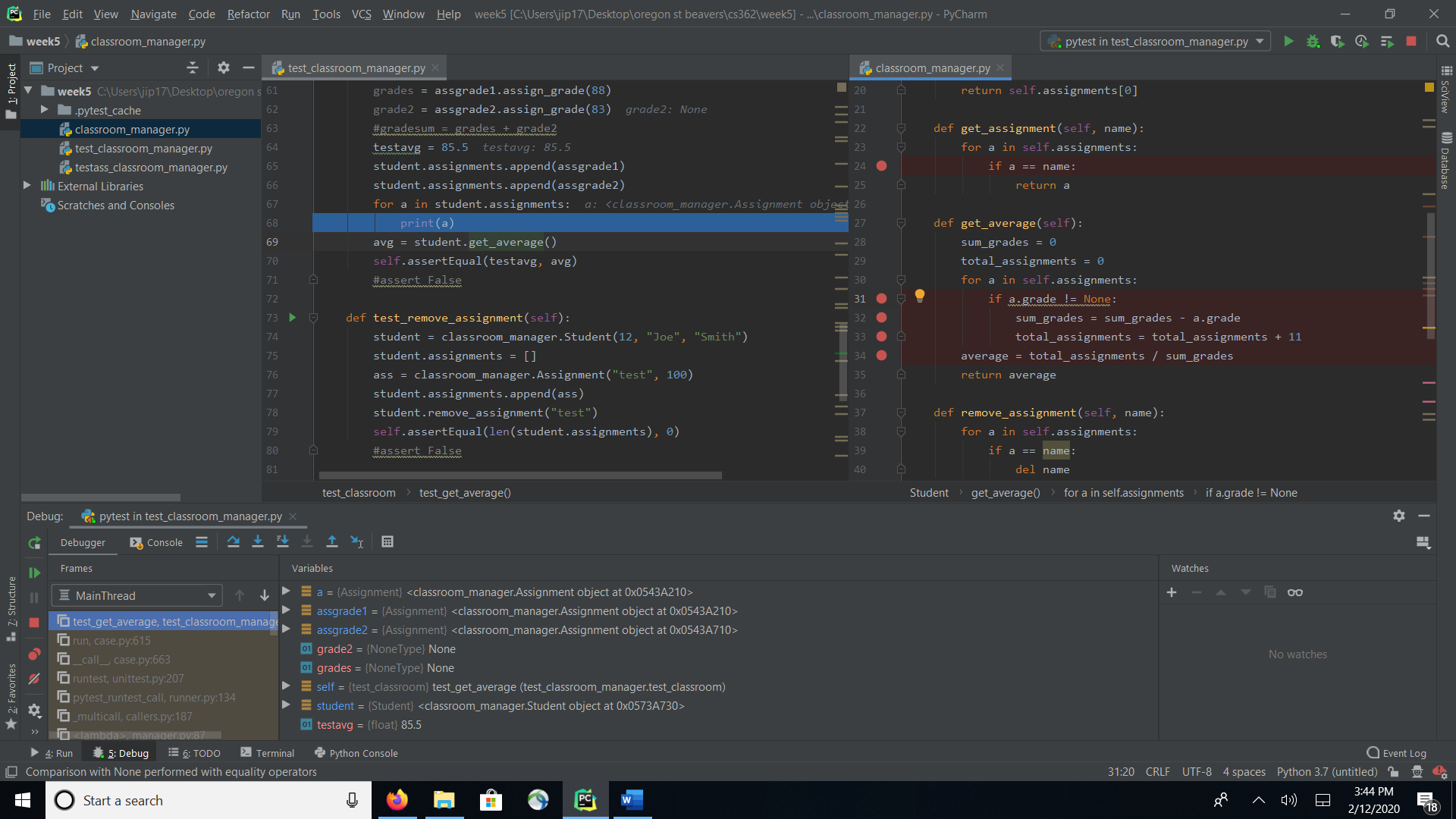
In get\_assignment I found a.name doesn’t exist changed it a == name

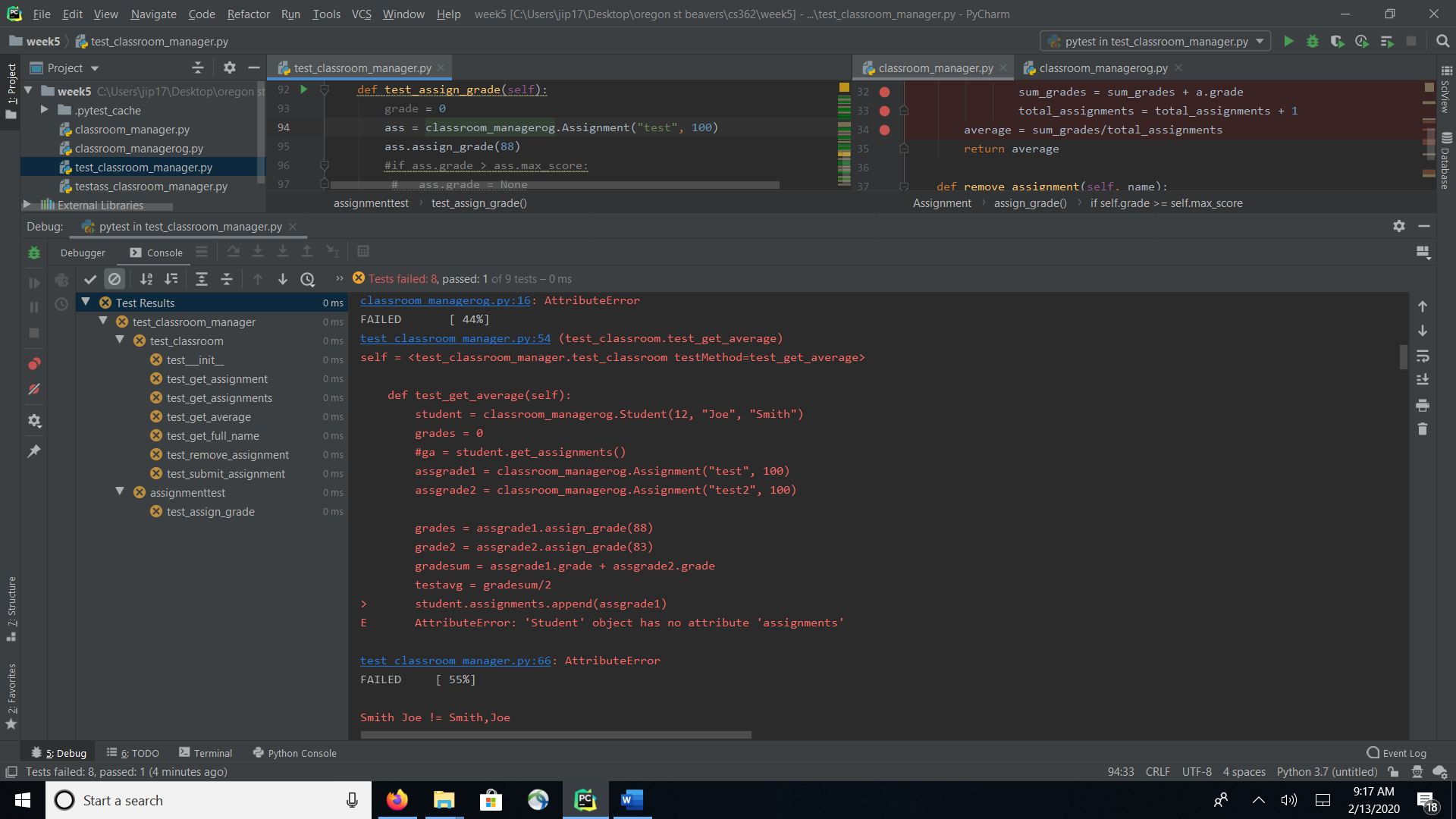




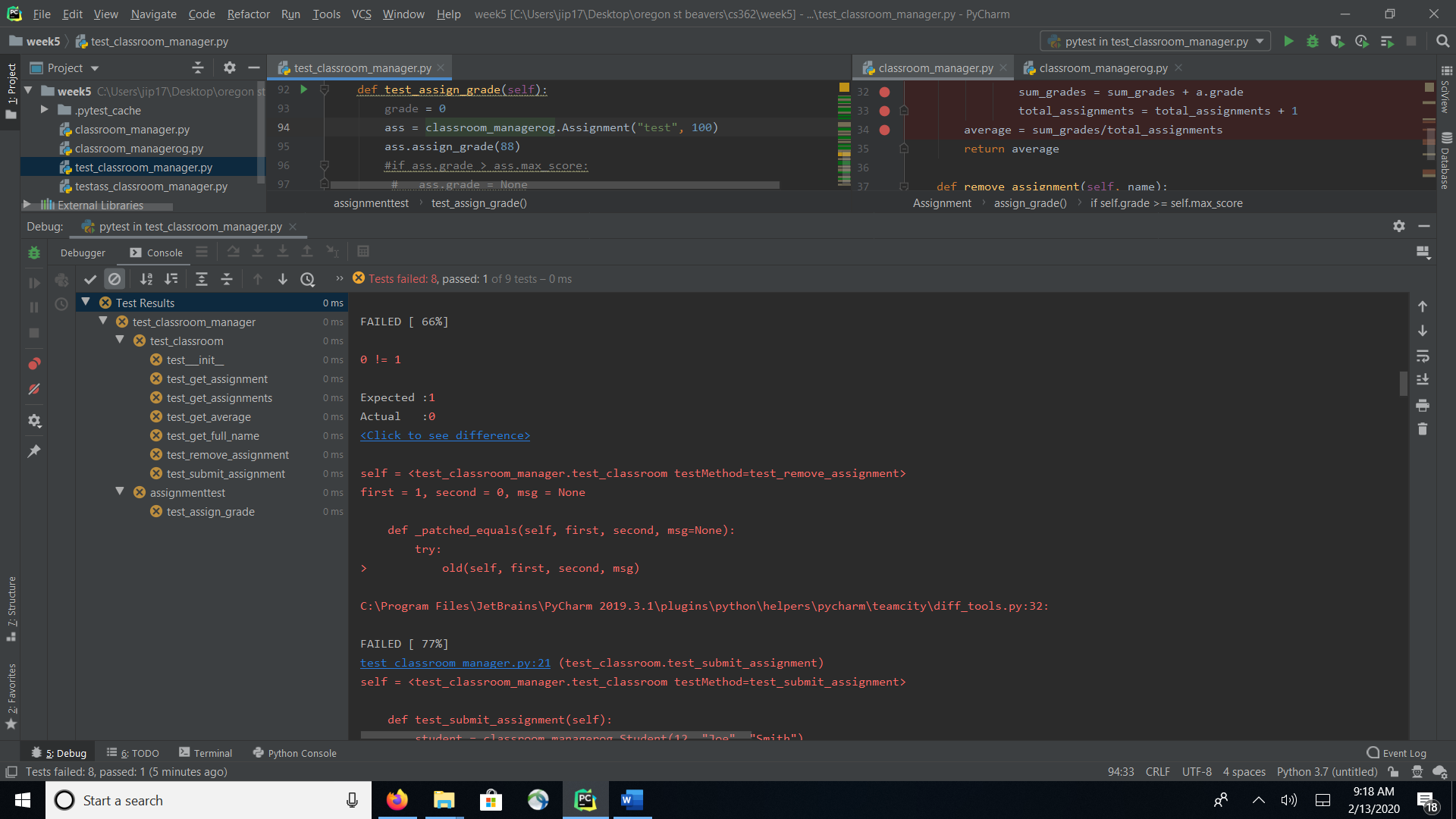




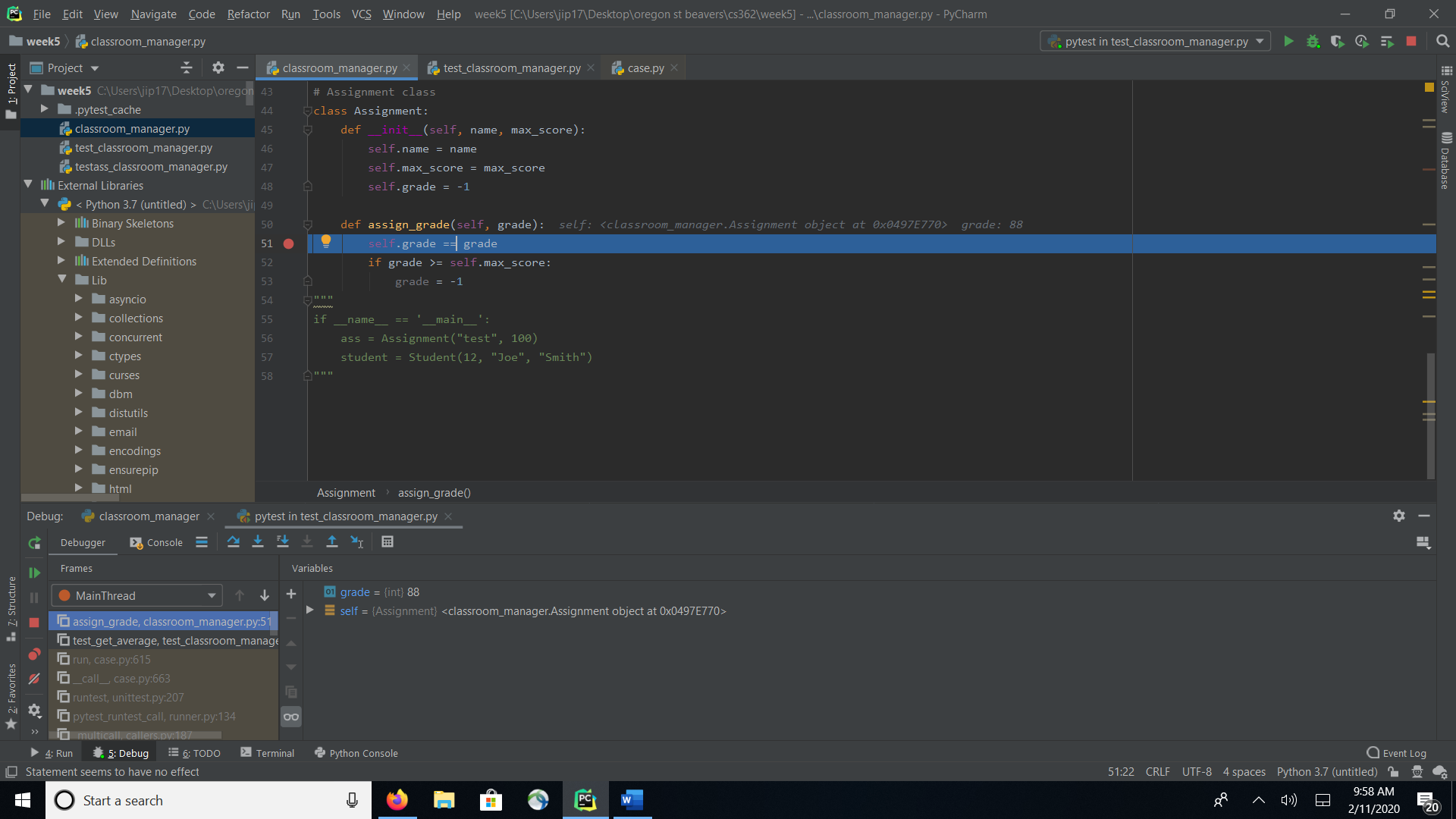
In the get\_average function I found a few bugs, the first one I changed was “!= ” to “is not” (not sure if this was needed). There was a mistake in the for loop that added +11 to each pass to total\_assignments, I changed this to +1. The formula for figuring out average was wrong, and I fixed it by switching total\_assignments/sum\_grades to sum\_grades/total\_assignments. 

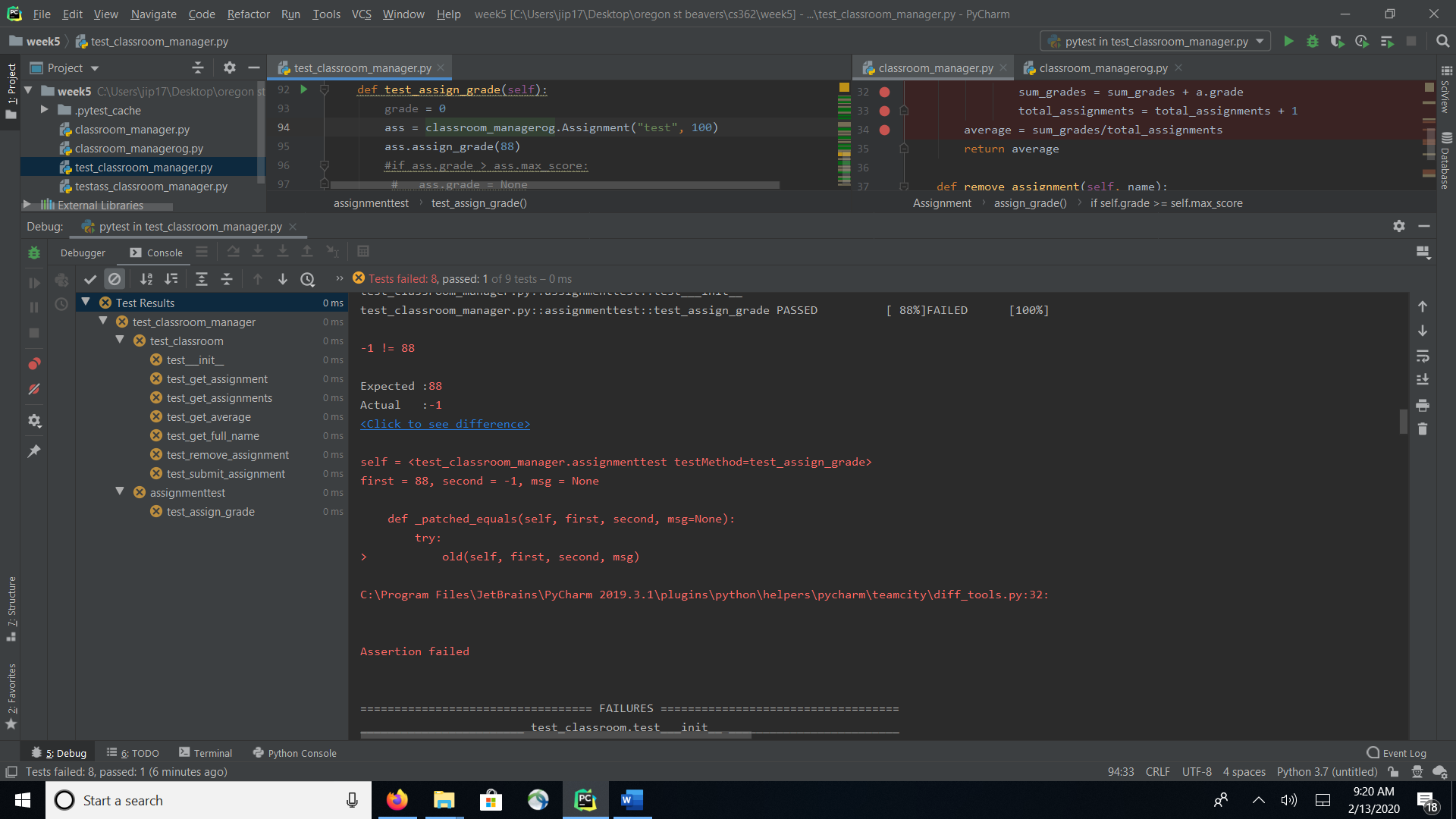


In remove\_assignment I changed del name to self.assignments.remove(name)



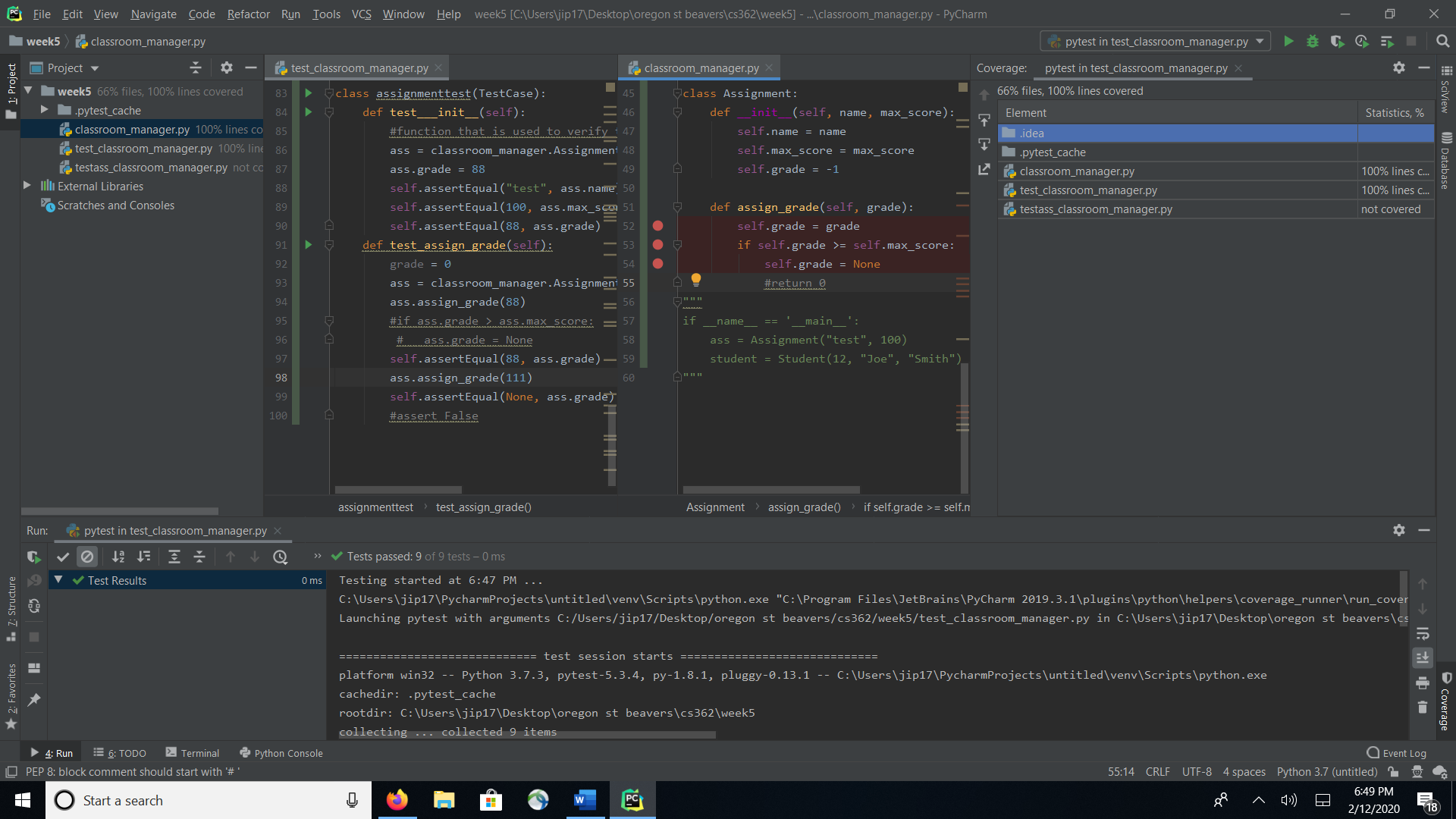
In assign\_grade the bug I ran into was a simple fix the original had == and I changed that to =. I also changed the if statement from if grade to if self.grade and when true instead of grade = -1 I fixed it to self.grade = None.





**Code Coverage**

Based on the tests ran using code coverage, I was able to achieve 100% code coverage



**Section D Git Repository**

