Convert this file to a .pdf before you submit. Files that are not readable will not be gradable and may not be re-submitted after the deadline.

Student group: (fill in names here)

Debugging

Line(s) containing error(s): ___9___
Explain error(s) and how you corrected it in 1-2 sentences:

The line split the data based on tabs and then tried to turn the entire list into an integer. This was corrected by splitting the data by new lines and using list comprehension to turn each individual item into a float instead of an integer.

Line(s) containing error(s): ___13___
Explain error(s) and how you corrected it in 1-2 sentences:

The line tried to set sum_sq equal to item minus the mean squared according to the order of operations. This was fixed by adding parenthesis so the difference between the item and mean was calculated first, and by changing the equals sign to += to calculate the sum.

Line(s) containing error(s): ___16___
Explain error(s) and how you corrected it in 1-2 sentences:

If n is true the line tried to divide sum_sq by the length of the data -1. When n is true mean_sq should be equal to sum_sq divided by the length of data, so the correction was removing the -1.

Line(s) containing error(s): ___18___
Explain error(s) and how you corrected it in 1-2 sentences:

When n is false line tried to divide sum_sq by the length of the data. When n is false it should divide by the length of data -1, so a -1 was added along with parenthesis to keep the order of the operation the same.

Copy and paste the two lines above for each additional line you need to correct. The number of blanks here is not indicative of the number of errors. You may not need all 3 provided, or you may need more.

Exception Handling

Name of Exception Handled:
ValueError
Lines that could raise the exception:
Line 9
Explanation of how exception could occur – user inputs and brief explanation in 1-3 sentences:

This exception can occur when the user inputs a file where there is a string that can't be converted into a integer. This will happen if there is a letter or word in the file and can be fixed by the user.

Name of Exception Handled:
IOError
Lines that could raise the exception:
Line 7
Explanation of how exception could occur – user inputs and brief explanation in 1-3 sentences:
This exception can occur if the user inputs a non-existent file name, or the file is located in a different directory of the computer, so it isn't accessible. When python tries to open the file, it will create the error and the rest of the program won't be able to run.

Name of Exception Handled:
ZeroDivisionError
Lines that could raise the exception:
Line 16 and 18
Explanation of how exception could occur – user inputs and brief explanation in 1-3 sentences:
This exception will occur if the user tries to use a file with a single value inside. If n = False, then the program will try to divide by the length of the data -1, which will be 1-1=0. This makes sense because you cannot take the standard deviation of a single number.

Test Code

Name of corresponding data file:
test_data_negative.txt
Justification- What case and part of the code this input is designed to handle, why it was chosen, and what test failure tells you:

This data file helped me to correct lines 13, 16, and 18 because they were easily calculatable and it was much easier to see where the errors were. This is a case where the user puts in perfect integer inputs and any test failure told me that there was a programmer error with the code instead of a user error.

Name of corresponding data file:
test_data_empty.txt
Justification- What case and part of the code this input is designed to handle, why it was chosen, and what test failure tells you:

This data file was meant to test the ValueError exception that can occur at line 9. It was chosen because if there is an empty file the for loop will not be able to iterate and it can't produce integers for the rest of the code. The test failure told me that the rest of the code is unable to run if the file is empty.

Name of corresponding data file:
test_data_numbers_with_letters.txt
Justification- What case and part of the code this input is designed to handle, why it was chosen, and what test failure tells you:

This data file tests the ValueError exception that occurs if there is a letter in the file. It was chosen because is the user were to make a mistake and have a word or letter inside of the file then the function wouldn't be able to execute line 9, because you can't change a letter into an integer. This test failure told me that the first letter will stop the iteration of the for loop and the function wouldn't be able to iterate over the rest of the numbers in the file.

Name of corresponding data file:
test_data_decimals.txt
Justification- What case and part of the code this input is designed to handle, why it was chosen, and what test failure tells you:

This data was used to test the for loop in line 9. It is designed to handle numbers that would have been cut off if they were turned into integers instead of floats. The test failure told me that I had to further change the code to allow for decimals.

Name of corresponding data file:
test_data_one_number.txt
Justification- What case and part of the code this input is designed to handle, why it was chosen, and what test failure tells you:

This data file tests the ZeroDivisionError exception which occurs when n=False. If the length of the data is one and the function divides by the len(data)-1 the error will occur. The test failure told me that you are not supposed to find the standard deviation of a single number.

You are limited to 5 test cases.

Reflection

I did not work in a group for this project and I also did not use any outside resources. I have included one extra exception handled for the extra credit. This was the ZeroDivisionError exception that I found would occur after I had put in a single number in the data file. I thought that this project was a good introduction to using a function tester and it was interesting to see how our code was being tested the entire year. I had some trouble with the exception handling at first because the code was generating errors since certain variables weren't being defined. I later realized that this could be fixed by laying the try except blocks and including else blocks that would only run if the try code passed. I also missed a few errors in the function that I only caught after I used the function tester. Overall, this was a very useful project that showed me how to use inheritance and the unittest object.

Sources used

Fill in any sources used here.