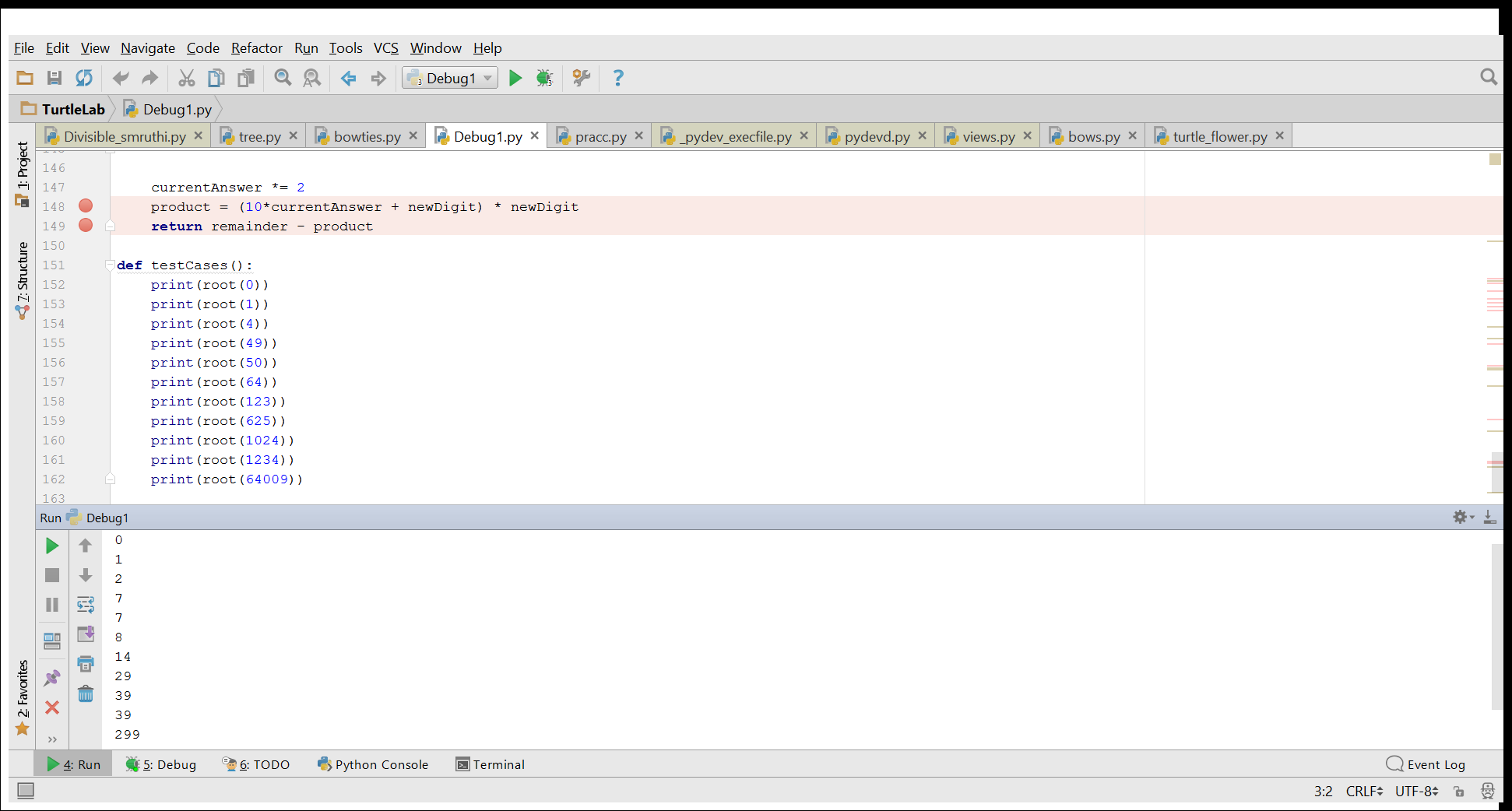
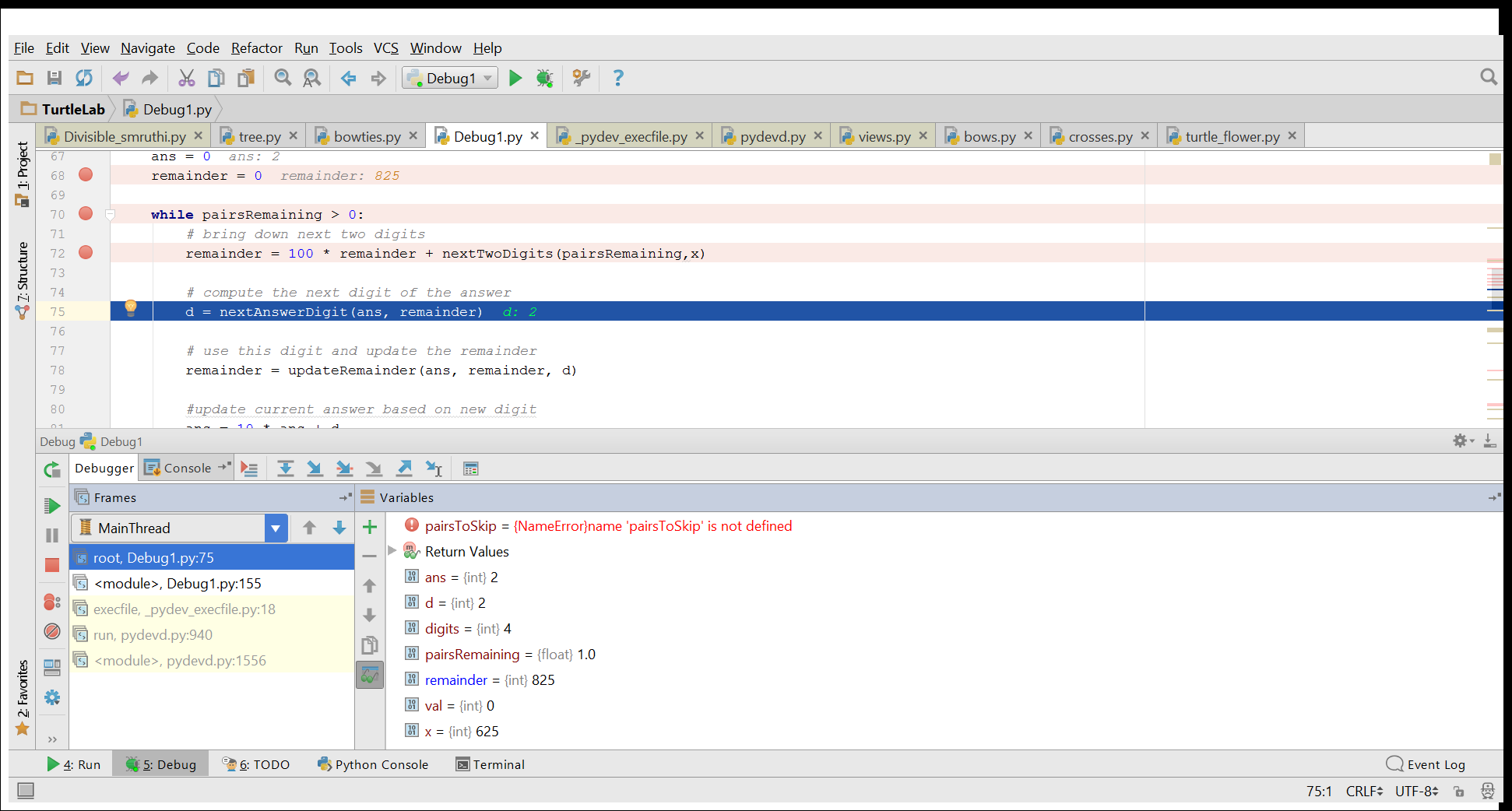
Task 1

# Test Cases:



The square root of numbers with more than 2 digits is incorrect

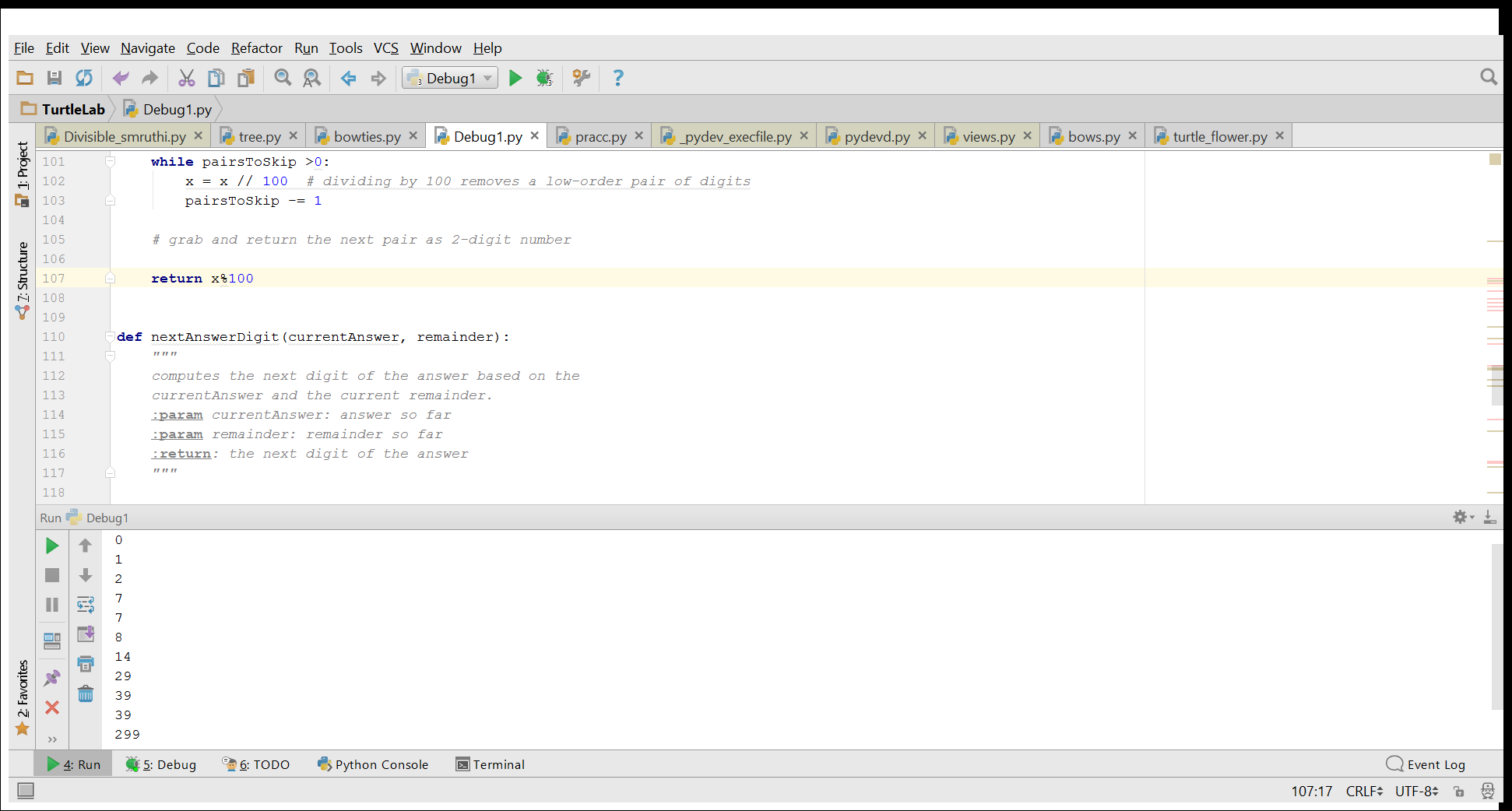
# Bug:



Ex: sqrt(625)

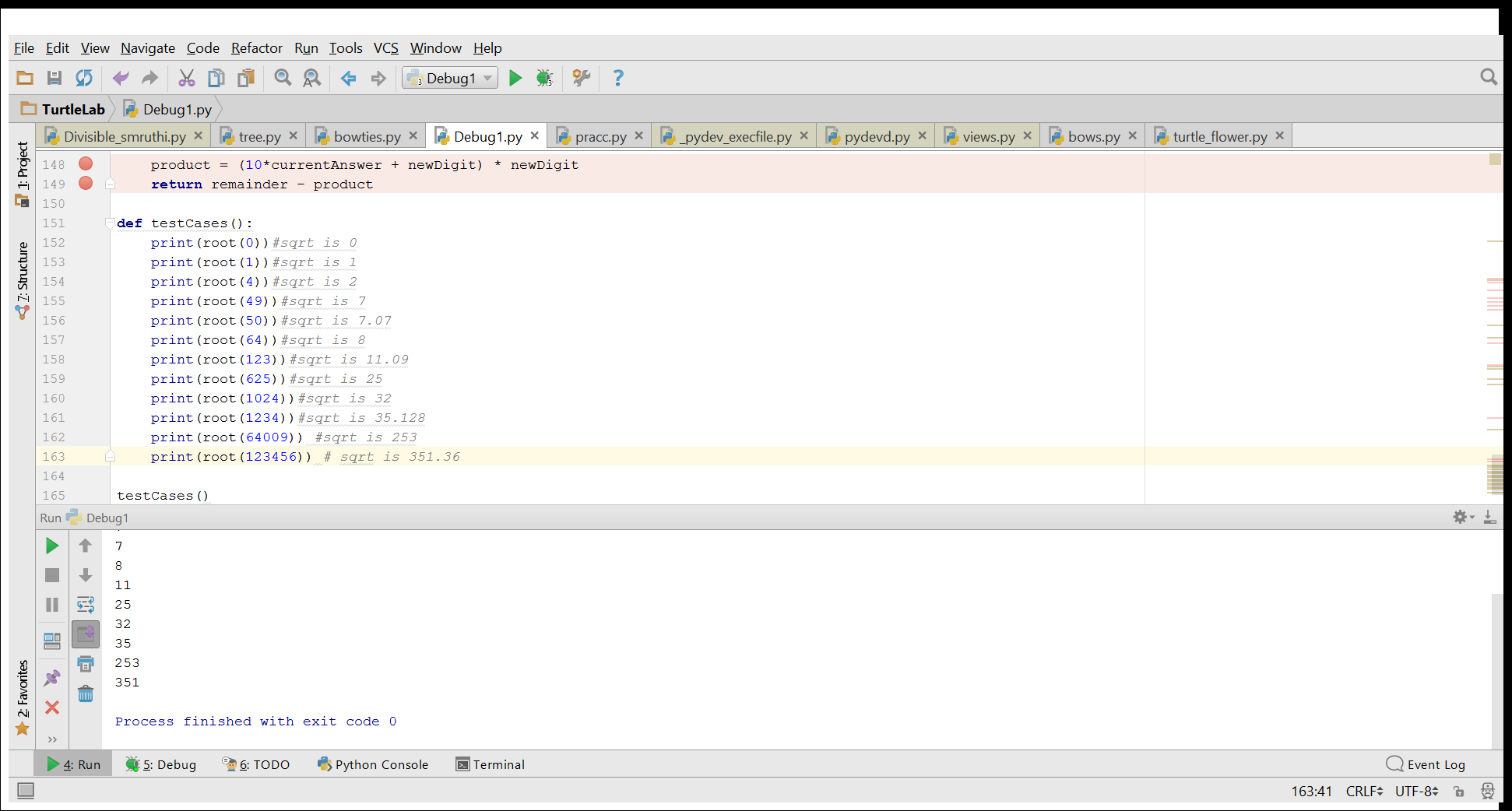
Remainder after 1st iteration is 825 instead of 225. The value of x is not updated to fetch the next pair.

# Corrected code:



The x value is updated to x%100 so that the next pair is picked for computation

# Result:



# Answer the following questions:

1. **Give 2 examples of inputs for which the provided code gives a correct answer despite the fact that it is flawed.**

Root(4)

Root(50)

1. **For each example, explain why the faulty code produced the correct answer, despite the flaw(s).**

The numbers which had only 2 digits did not have to go thru 2nd iteration where the 2nd pair was supposed to be brought down and added to the remainder of the previous division. Hence in case of numbers with 2 digits returned the correct answer despite the flawed logic(semantic error).

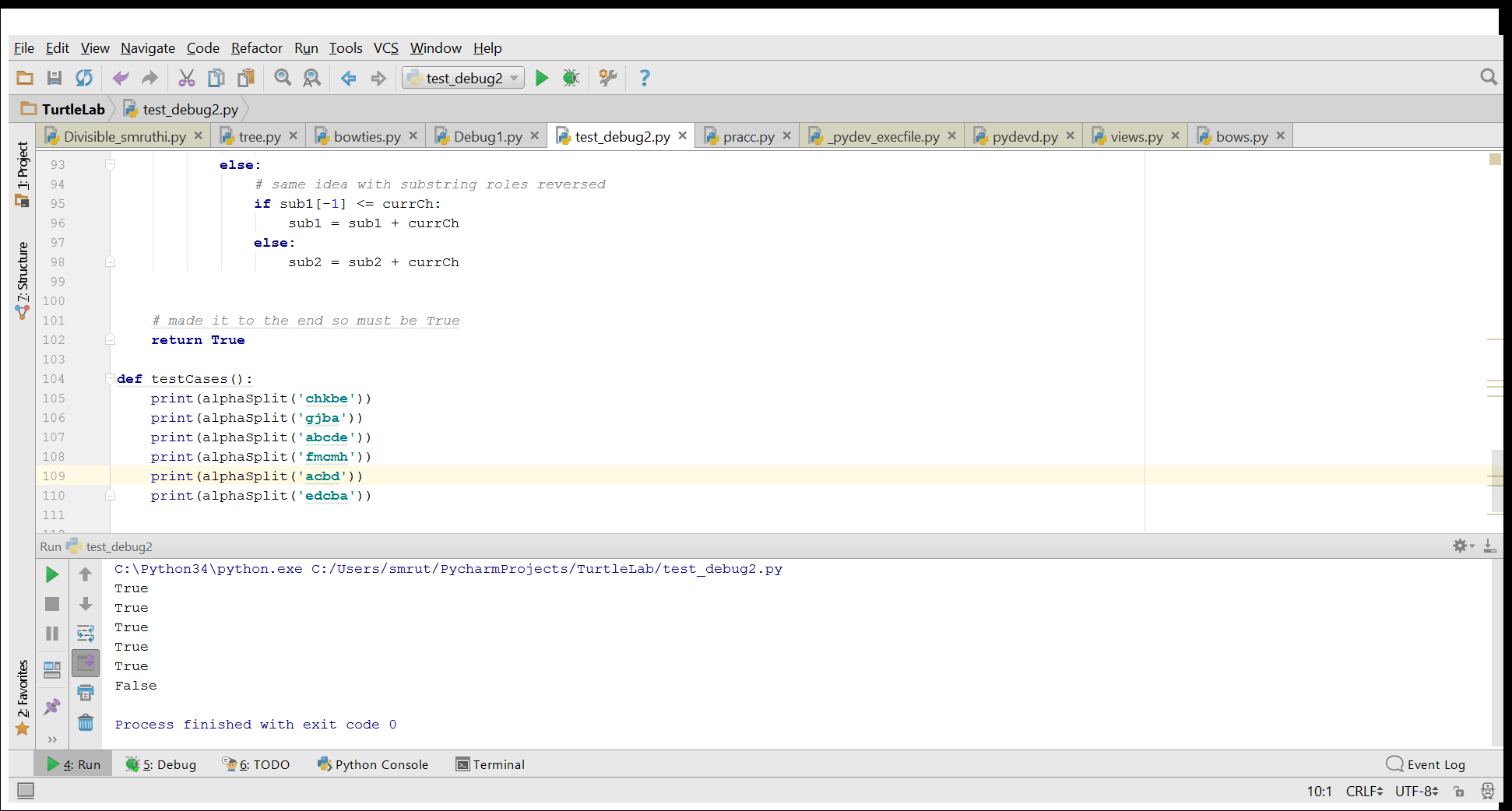
1. **Describe the bug(s) present in the code, and for each bug, indicate what the fix is.**

The bug in this case was a semantic error where the value for x was not updated for remaining pair of values in a number with more than 2 digits. i.e. the remainder was added with initial value of x instead of bring down the next pair of digits of x.

The fix the error, the value of x after each iteration should be updated to x%100 which will return the rest of the digits for computing the squareroot and add the correct value to the remainder.

Task 2

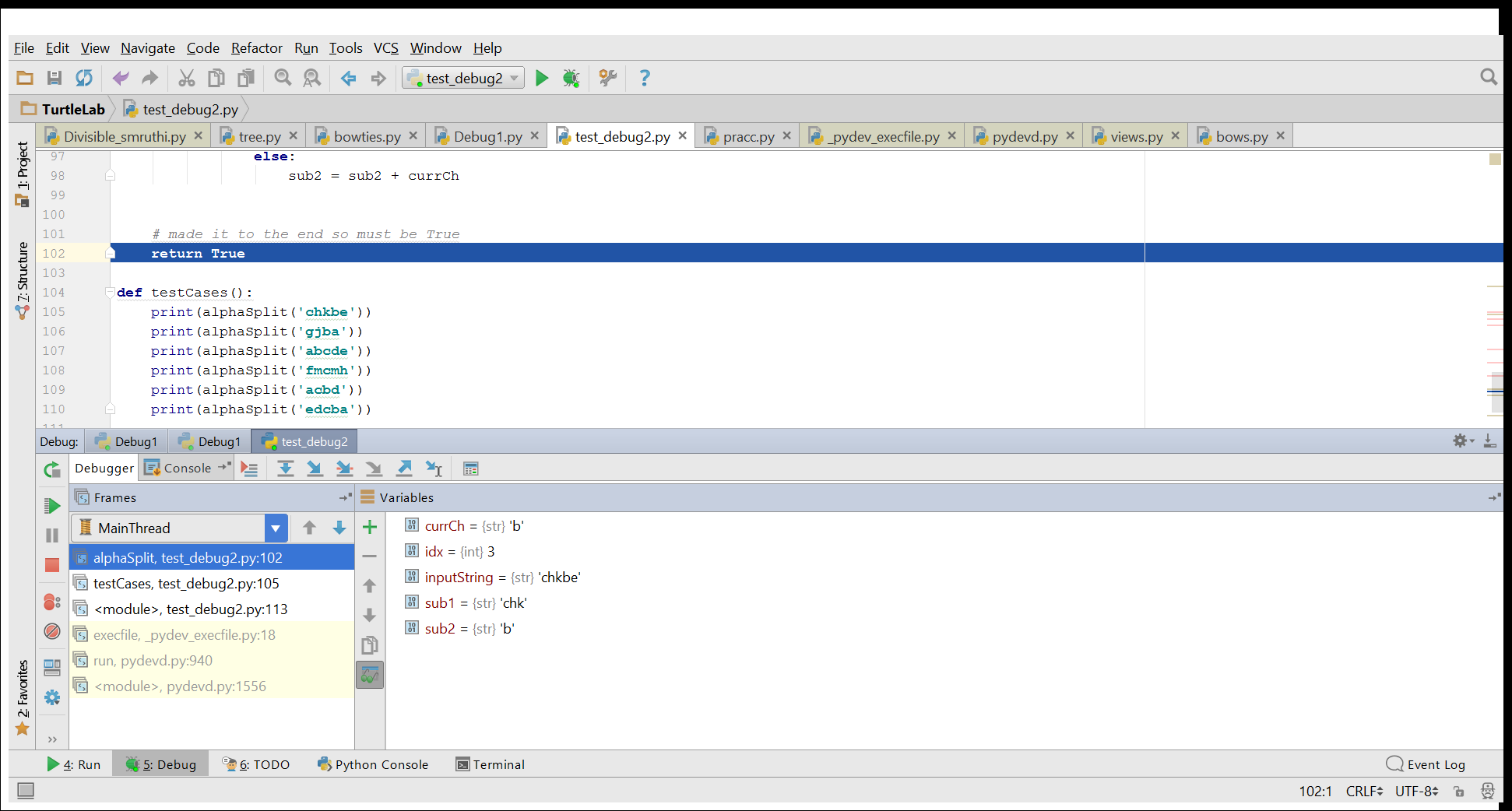
# Test cases:



The string ‘gjba’ returned true when it should have been false

# Debugging:

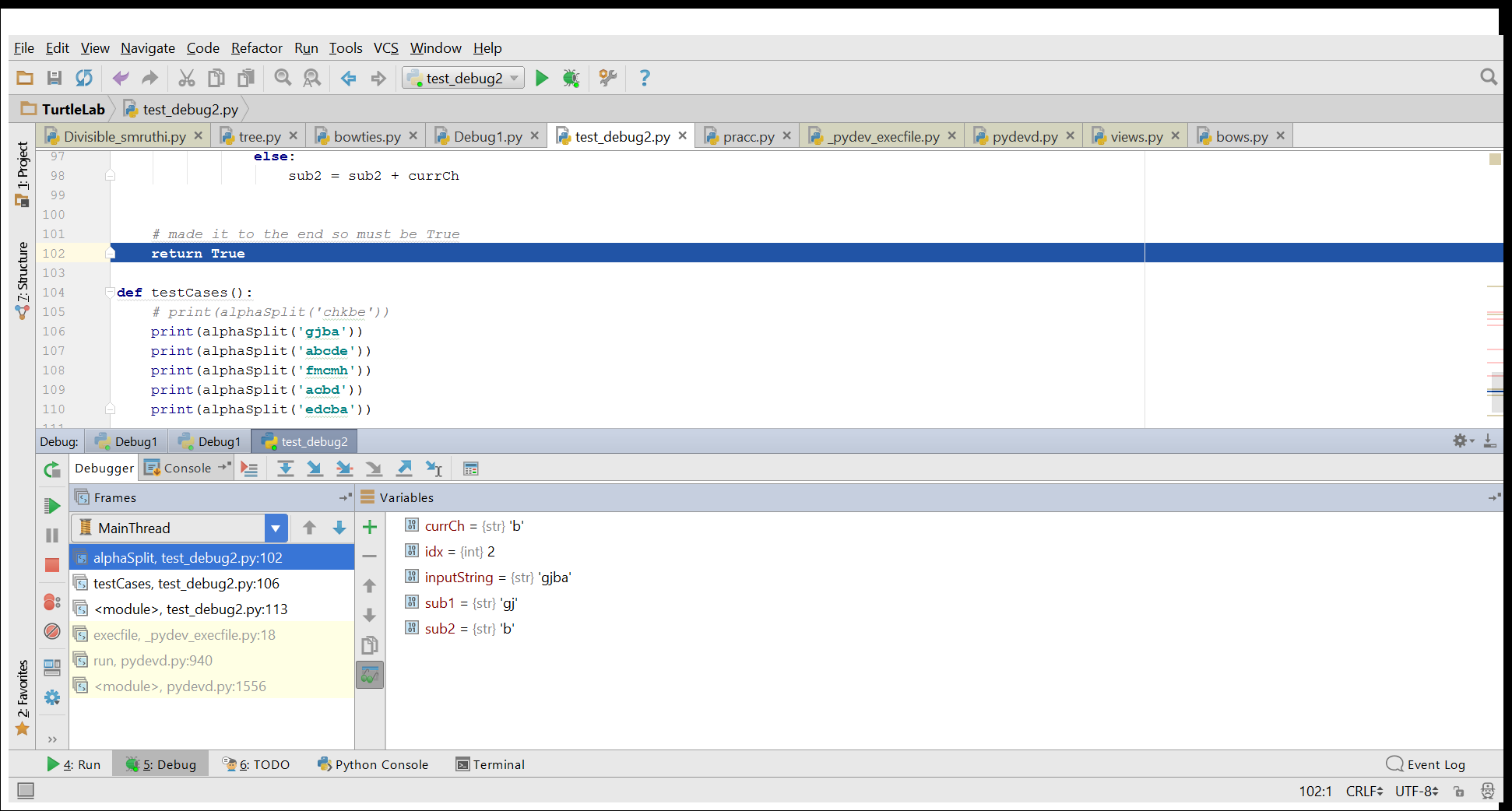
Bug1:



Input:chkbe

Bug: Control moved to end of program when the last letter was still not checked and placed in one of the substrings and returned true.

Output:true

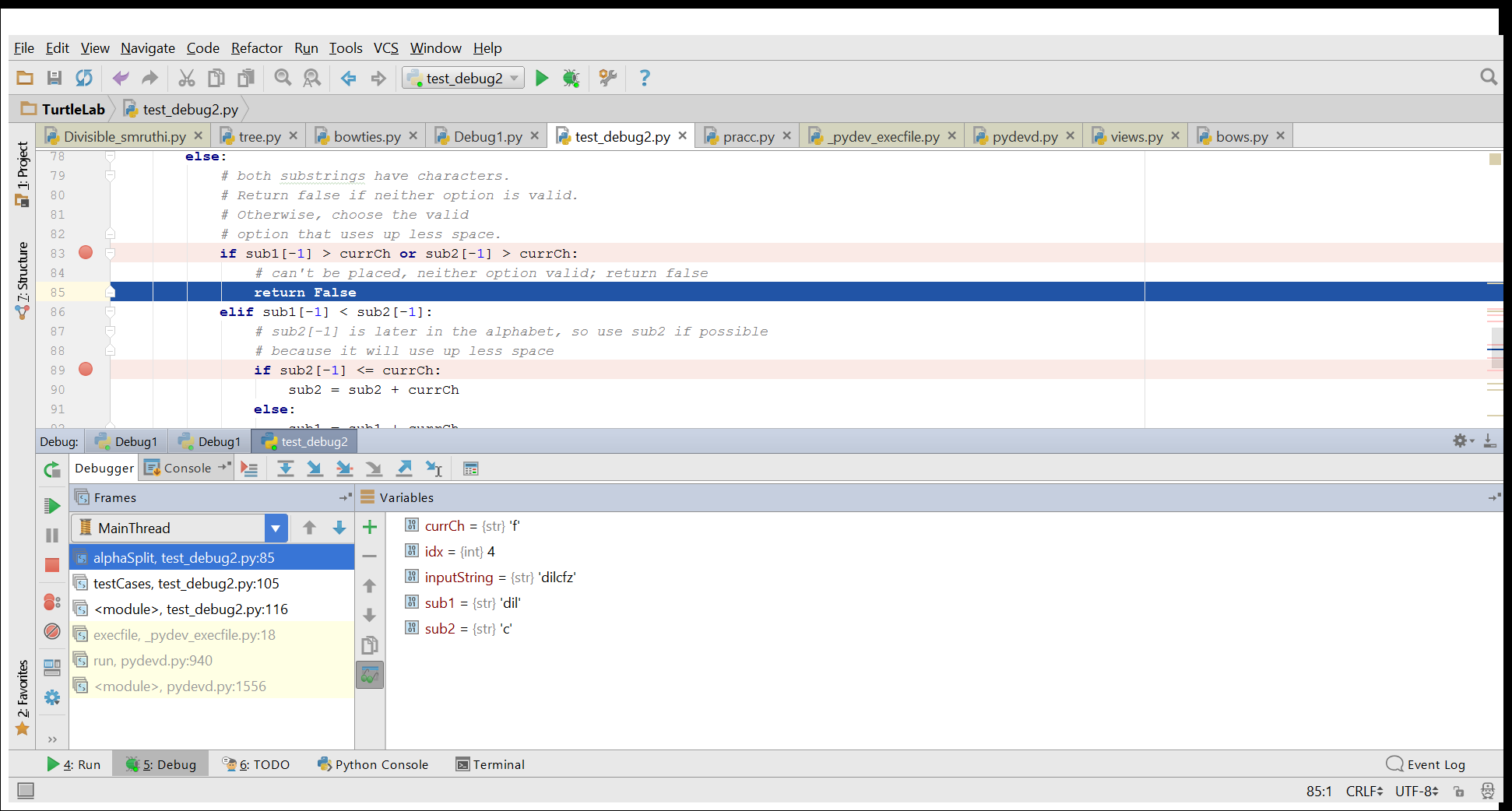


Input:gjba

Bug: last letter was not validated and control Returned true even when the letter ‘a’ cannot be placed in either substrings

Output: true

# Bug 2:



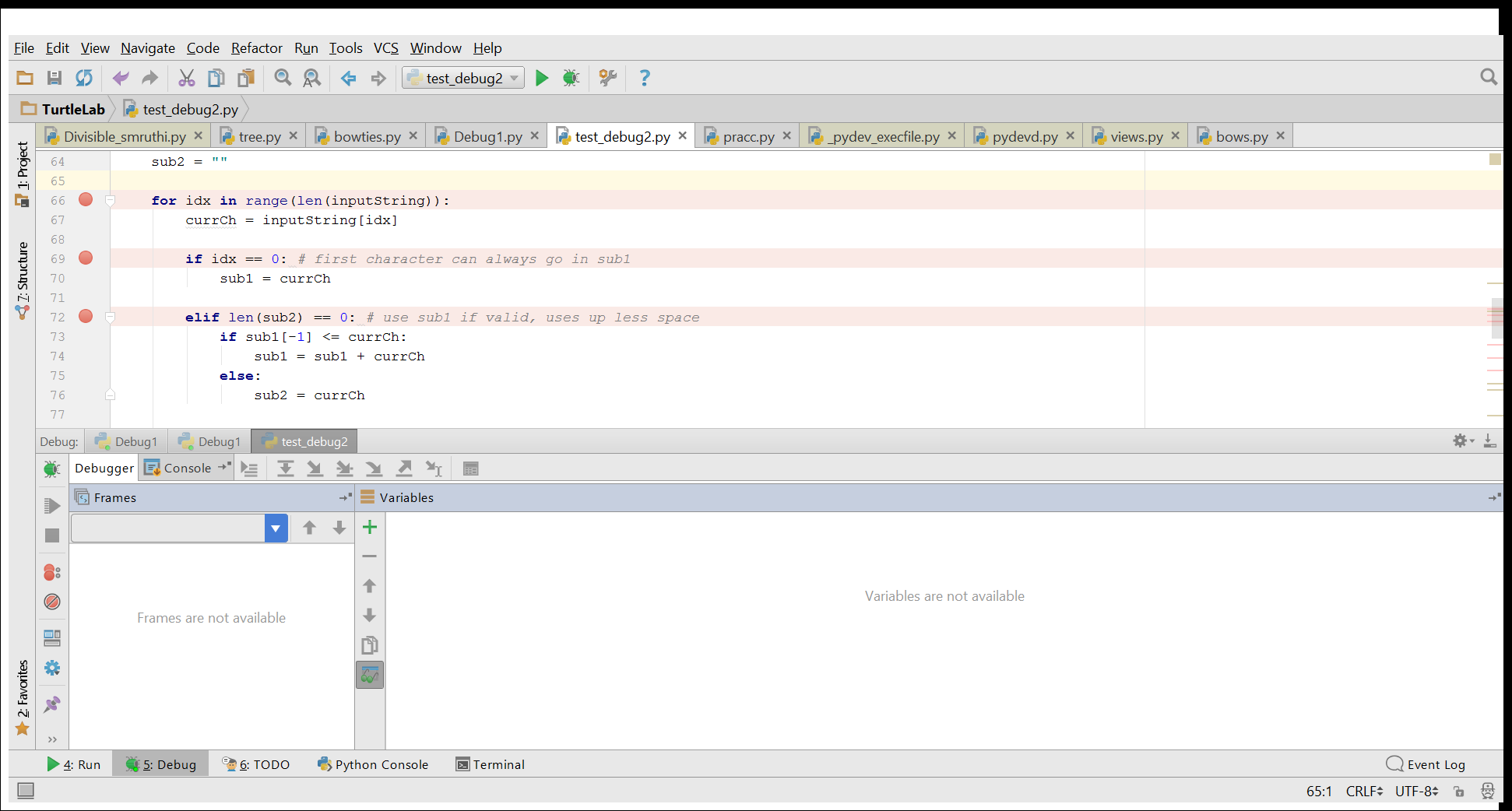
Input:’dilcfz’

Bug:Returned False even when ‘f’ > ‘c’ and it can be placed in sub2

Output:false

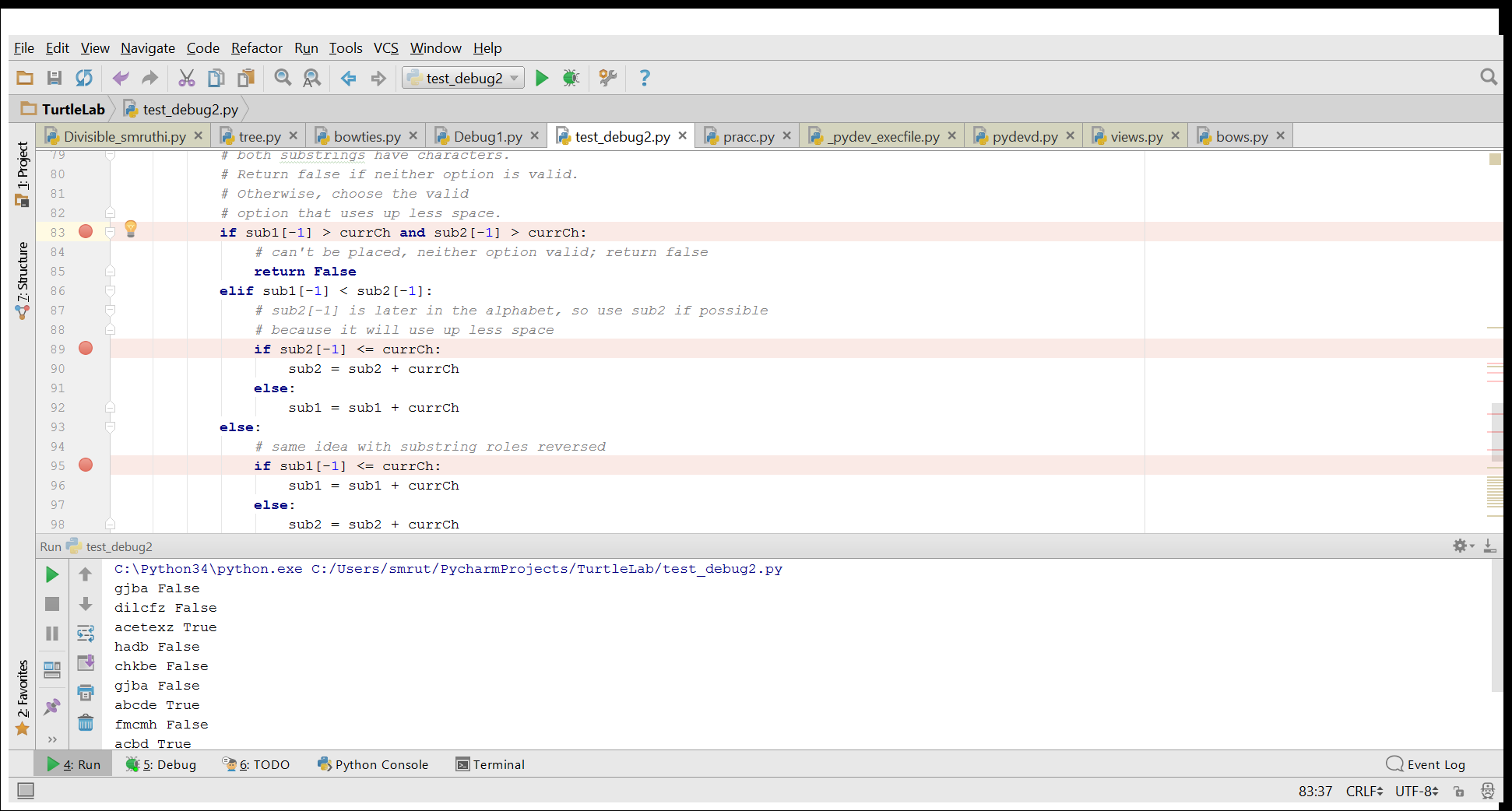
# Corrected code:

Correction 1:



Parse through entire string with len(inputstring) rather than len(inputstring)-1

Correction 2:



Changed the conditional statement from

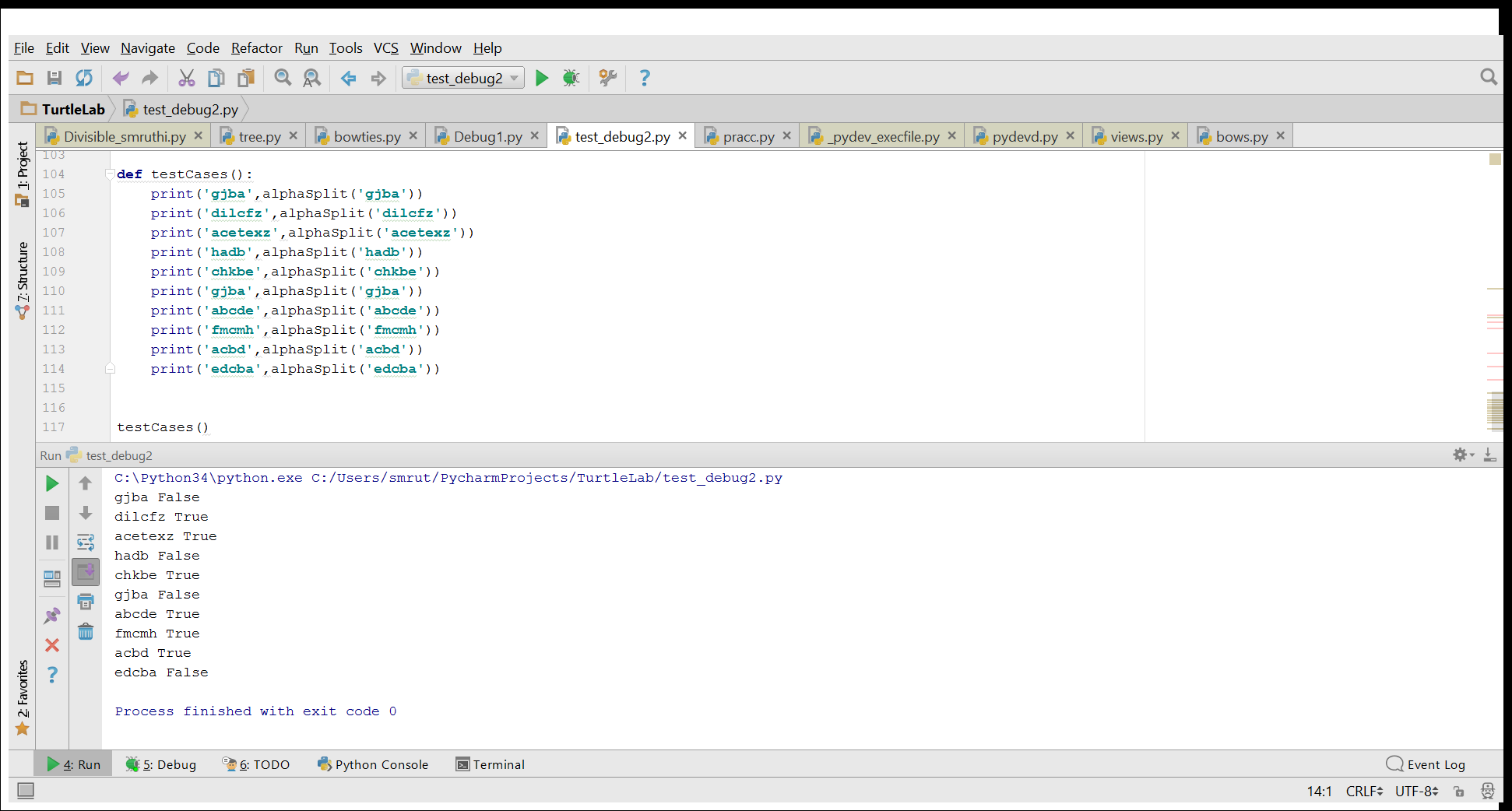
**if** sub1[-1] > currCh **or** sub2[-1] > currCh:

to

**if** sub1[-1] > currCh **and** sub2[-1] > currCh:

# Result:

After correction



1. **Give 2 examples of inputs for which the provided code gives a correct answer despite the fact** **that it is flawed.**

Example 1:edcba

Example 2: dilcf

1. **For each example, explain why the faulty code produced the correct answer, despite the flaw(s).**

In **Example 1**, the characters after ‘e’,’d’ cannot be placed in either sub strings even before reaching the end of the string ,hence the bug does not affect the result.

In **Example 2**, ‘dilcf’ returned true even when the below conditional statement would have returned false

i.e. l>f which is true and the ‘**or’** command would not check for the 2nd condition hence the command would move to **return False**

**if** sub1[-1] > currCh **or** sub2[-1] > currCh:  
 *# can't be placed, neither option valid; return false* **return False**

But,since the last character (f) was truncated due to the below command, the control moved to end of the program and returned true.

**for** idx **in** range(len(inputString)-1):

1. **Describe the bug(s) present in the code, and for each bug, indicate what the fix is.**

Bug 1: the base case of the loop truncates the last character of the string and ends it before the last character is read.It has to be corrected to

**for** idx **in** range(len(inputString)):

Bug 2: The conditional statement which is used to check if the character is greater than last characters of both substrings has an ‘or’ command which will return false even if one of the substrings can accommodate the character. It should be corrected to

**if** sub1[-1] > currCh **and** sub2[-1] > currCh:

where the function returns false only if both the substrings cannot append the character to the end of the string.