

CSE 231 Lab Exercise #2

(Complete the Pre-Lab on D2L before your lab session)

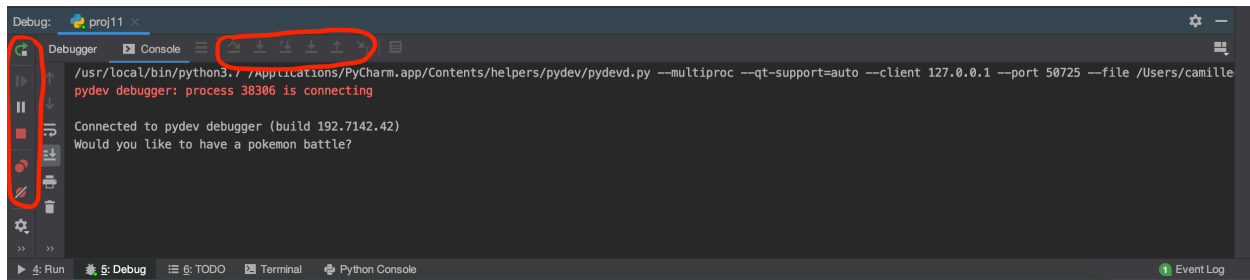
PART 1: Debugging

Assignment Background

This lab exercise provides practice with debugging, meaning the process of identifying and removing errors from your code.

When you are done, demonstrate your results to your TA. On-line (Section 730) students should do it on their own (Do not submit to mimic).

The art of debugging is an integral part of programming. The following exercise is designed to help you understand the basics of the PyCharm debugger. Familiarize yourself with the debug control buttons in PyCharm. They are in the left hand side and the top of the “debug” window that appears when you start debugging your code. Hover your mouse over each one to identify what each button does.



Set breakpoints on lines by clicking to the right of any of the the line numbers. Try it and you will see a red circle; click again to make it disappear. Run the program in debug mode by pressing the Debug button (the green bug on the top bar, or CTRL + D). Execution “freezes” when a breakpoint is encountered. You can then progress the program one statement at a time by pressing the Step Over (CTRL + F8) button (leftmost on the top bar). The next line to be executed will be highlighted in the running file. You can use this along with Step Over to understand which lines of your code are executed. Click on the “Debugger” tap (next to Console) to see the values stored in the variables you created.

Assignment Overview

Open up the file debug2.py in PyCharm. Set a breakpoint at the first executable line of the program (input), and start the debugger by pressing the “Debug” button- not that it may be necessary to first right click on the file and choose “Debug ‘debug2.py’” before this button is available.

Answer the following questions for the input provided below and compare to the results:

Note: Please *do not* use the print() statements during the debug exercise.

Input set 1

Input1: AES

Input2: n

1) As you step through the program, record the sequence of line numbers executed. (e.g. 2, 4, 9)

Input set 2

Input1: aes

Input2: n

2) a. As you step through the program, record the sequence of line numbers executed.

b. When the next line highlighted is line 14, which variables are held in memory and what are their values?

Additional Resources

<https://www.jetbrains.com/help/pycharm/debugging-your-first-python-application.html>

Answers to Input set 1 and Input set 2:

Input Set 1: 8, 10, 12, 14, 15, 18, 21, 23, 10, 25

Input Set 2: a. 8, 10, 12, 14, 16, 18, 21, 23, 10, 25

b. algorithm = aes

go_again = y

Part 2: Programming with Control Structures Assignment Background

Assignment Background

This lab exercise provides practice with selection (**if**) and repetition (**for**, **while**) in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

The if statement is used to choose between alternatives. The for and while statements are used to repeatedly execute blocks of statements.

The for statement is most useful when you know exactly how many times you want to repeatedly do something (for example, you know how many sides of a polygon you want to draw). Here is an example showing how to print something 4 times:

```
n = 4
for i in range(n):
    print("Hi!")
```

Here is the corresponding output:

```
Hi!
Hi!
Hi!
Hi!
```

The while statement is most useful when you don't know in advance how many times the program will be required to repeat a task (for example, prompting the user to enter a value until correct input is provided). Here we will use while to repeat a fixed number of times simply to practice using the statement. To compare and contrast with the previous example, we will print something n times:

```
n = 4
count = 0
while count < n:
    print("Hi!")
    count = count + 1
```

Here is the corresponding output:

```
Hi!
Hi!
Hi!
Hi!
```

In this example, note that you must manage the counter variable (“count”) yourself. If you make a mistake with the counter management, the loop may end up repeating forever. In that case, in the upper right of PyCharm has a Stop button (Red square).

Assignment Overview

Download the file “lab02a.py” and develop a Python program which inputs a series of integers and processes them. The program will:

- a) Continue to process values until the user enters the value 0
- b) Ignore all negative integers
- c) Count the number of odd integers entered
- d) Count the number of even integers entered
- e) Calculate the sum of the odd integers in the series
- f) Calculate the sum of the even integers in the series
- g) Display the sum of odds
- h) Display the sum of evens
- i) Display the count of odds
- j) Display the count of evens
- k) Display the total number of positive integers entered
- l) Optional: print a message whenever a negative integer is entered

Sample output:

```
Input an integer (0 terminates): 1
Input an integer (0 terminates): 3
Input an integer (0 terminates): -2
Input an integer (0 terminates): 2
Input an integer (0 terminates): 6
Input an integer (0 terminates): 5
Input an integer (0 terminates): 0

sum of odds: 9
sum of evens: 8
odd count: 3
even count: 2
total positive int count: 5
```

Commentary

- a) A nice characteristic of this problem is that it can be developed in small pieces. Remember: always try to break a problem into smaller pieces that are easier to solve.
- b) First write a program that prompts for an integer, displays the integer, and stops asking for more integers when 0 is entered. Use while. Under what condition do you continue to loop? Here is a suggested outline:
prompt for an integer # (and convert the string to an int)
while some_Boolean_condition:
 # do something
 prompt for another integer
- c) Once that simple program is tested and working, add in another piece such as (c) to count the number of odd integers entered. Create a variable with an appropriate name, such as odd_count, and assign it an initial value of 0 (before the while loop). When an odd number is entered, add one to odd_count (for example, odd_count += 1). Display the count.
- d) Next: count the number of even integers and display the count.
- e) Next: calculate the sum of the odd integers. The approach is similar to counting: choose a variable name, initialize it to 0; when the integer is odd, add the integer to the variable. Display the sum.
- f) Next: calculate the sum of the even integers and display the sum.

- g) Finally, ignore all negative numbers which the user inputs. One approach: if the integer is positive, do all the counting and summing, else do nothing (that is, you can use an if statement without an else clause).
- h) If you have time, try this: print a message if the integer is negative.

Demonstrate your completed program to your TA. On-line (Section 730) students should submit the completed program (named “lab02a.py”) for grading via the Mimir system.

Note (from the syllabus): Labs are credit/no-credit. To receive credit for a lab

1. You must complete the Pre-Lab, before your lab or the pre-lab deadline, whichever comes first. Pre-labs

are "warm-up" for the labs and are not expected to be perfect -- our expectation is necessarily fuzzy for pre-labs: "you are expected to get most of them correct most of the time."

2. When Mimir tests exist, they test perfection. The Mimir tests allow you to verify that your code is correct.

However, you can get credit for correct Lab code that isn't perfect, i.e. it is possible to get credit for code that fails Mimir tests. (Note that Projects are more strict with respect to Mimir tests.)

3. Adhering to the Coding Standard is expected, but expectations are less strict than for Projects.