**Python Lab 5**

**What is debugging?**

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs." — *Maurice Wilkes discovers debugging*, 1949

By now if you have been messing around with the programs you have probably found that sometimes the program does something you didn't want it to do. This is fairly common. Debugging is the process of figuring out what the computer is doing and then getting it to do what you want it to do. This can be tricky. I once spent nearly a week tracking down and fixing a bug that was caused by someone putting an x where a y should have been.

This lab will be more abstract than previous labs.

**What should the program do?**

The first thing to do (this sounds obvious) is to figure out what the program should be doing if it is running correctly. Come up with some test cases and see what happens. For example, let's say I have a program to compute the perimeter of a rectangle (the sum of the length of all the edges). I have the following test cases:

|  |  |  |
| --- | --- | --- |
| **height** | **width** | **perimeter** |
| 3 | 4 | 14 |
| 2 | 3 | 10 |
| 4 | 4 | 16 |
| 2 | 2 | 8 |
| 5 | 1 | 12 |

I now run my program on all of the test cases and see if the program does what I expect it to do. If it doesn't then I need to find out what the computer is doing.

More commonly some of the test cases will work and some will not. If that is the case you should try and figure out what the working ones have in common. For example here is the output for a perimeter program (you get to see the code in a minute):

Height: **3**

Width: **4**

perimeter = 15

Height: **2**

Width: **3**

perimeter = 11

Height: **4**

Width: **4**

perimeter = 16

Height: **2**

Width: **2**

perimeter = 8

Height: **5**

Width: **1**

perimeter = 8

Notice that it didn't work for the first two inputs, it worked for the next two and it didn't work on the last one. Try and figure out what is in common with the working ones. Once you have some idea what the problem is finding the cause is easier. With your own programs you should try more test cases if you need them.

**What does the program do?**

The next thing to do is to look at the source code. One of the most important things to do while programming is reading source code. The primary way to do this is code walkthroughs.

A code walkthrough starts at the first line, and works its way down until the program is done. while loops and if statements mean that some lines may never be run and some lines are run many times. At each line you figure out what Python has done.

Lets start with the simple perimeter program. Don't type it in, you are going to read it, not run it. The source code is:

height = int(input("Height: "))

width = int(input("Width: "))

**print**("perimeter =", width + height + width + width)

***Question:* What is the first line Python runs?**

*Answer:* The first line is always run first. In this case it is: height = int(input("Height: "))

**What does that line do?**

Prints Height: , waits for the user to type a string in, and then converts the string to an integer variable height.

**What is the next line that runs?**

In general, it is the next line down which is: width = int(input("Width: "))

**What does that line do?**

Prints Width: , waits for the user to type a number in, and puts what the user types in the variable width.

**What is the next line that runs?**

When the next line is not indented more or less than the current line, it is the line right afterwards, so it is: print("perimeter = ", width + height + width + width)(It may also run a function in the current line, but that's a future lab.)

**What does that line do?**

First it prints perimeter = , then it prints the sum of the values contained within the variables, width and height, from width + height + width + width.

**Does width + height + width + width calculate the perimeter properly?**

Let's see, perimeter of a rectangle is the bottom (width) plus the left side (height) plus the top (width) plus the right side (huh?). The last item should be the right side's length, or the height.

**Do you understand why some of the times the perimeter was calculated "correctly"?**

It was calculated correctly when the width and the height were equal.

The next program we will do a code walkthrough for is a program that is supposed to print out 5 dots on the screen. However, this is what the program is outputting:

. . . .

And here is the program:

number = 5

**while** number > 1:

**print**(".",end=" ")

number = number - 1

**print**()

This program will be more complex to walkthrough since it now has indented portions (or control structures). Let us begin.

**What is the first line to be run?**

The first line of the file: number = 5

**What does it do?**

Puts the number 5 in the variable number.

**What is the next line?**

The next line is: while number > 1:

**What does it do?**

Well, while statements in general look at their expression, and if it is true they do the next indented block of code, otherwise they skip the next indented block of code.

**So what does it do right now?**

If number > 1 is true then the next two lines will be run.

**So is number > 1?**

The last value put into number was 5 and 5 > 1 so yes.

**So what is the next line?**

Since the while was true the next line is: print(".",end=" ")

**What does that line do?**

Prints one dot and since the extra argument end=" " exists the next printed text will not be on a different screen line.

**What is the next line?**

number = number - 1 since that is following line and there are no indent changes.

**What does it do?**

It calculates number - 1, which is the current value of number (or 5) subtracts 1 from it, and makes that the new value of number. So basically it changes number's value from 5 to 4.

**What is the next line?**

Well, the indent level decreases so we have to look at what type of control structure it is. It is a while loop, so we have to go back to the while clause which is while number > 1:

**What does it do?**

It looks at the value of number, which is 4, and compares it to 1 and since 4 > 1 the while loop continues.

**What is the next line?**

Since the while loop was true, the next line is: print(".",end=" ")

**What does it do?**

It prints a second dot on the line, ending by a space.

**What is the next line?**

No indent change so it is: number = number - 1

**And what does it do?**

It takes the current value of number (4), subtracts 1 from it, which gives it 3 and then finally makes 3 the new value of number.

**What is the next line?**

Since there is an indent change caused by the end of the while loop, the next line is: while number > 1:

**What does it do?**

It compares the current value of number (3) to 1. 3 > 1 so the while loop continues.

**What is the next line?**

Since the while loop condition was true the next line is: print(".",end=" ")

**And it does what?**

A third dot is printed on the line.

**What is the next line?**

It is: number = number - 1

**What does it do?**

It takes the current value of number (3) subtracts from it 1 and makes the 2 the new value of number.

**What is the next line?**

Back up to the start of the while loop: while number > 1:

**What does it do?**

It compares the current value of number (2) to 1. Since 2 > 1 the while loop continues.

**What is the next line?**

Since the while loop is continuing: print(".",end=" ")

**What does it do?**

It discovers the meaning of life, the universe and everything. I'm joking. (I had to make sure you were awake.) The line prints a fourth dot on the screen.

**What is the next line?**

It's: number = number - 1

**What does it do?**

Takes the current value of number (2) subtracts 1 and makes 1 the new value of number.

**What is the next line?**

Back up to the while loop: while number > 1:

**What does the line do?**

It compares the current value of number (1) to 1. Since 1 > 1 is false (one is not greater than one), the while loop exits.

**What is the next line?**

Since the while loop condition was false the next line is the line after the while loop exits, or: print()

**What does that line do?**

Makes the screen go to the next line.

**Why doesn't the program print 5 dots?**

The loop exits 1 dot too soon.

**How can we fix that?**

Make the loop exit 1 dot later.

**And how do we do that?**

There are several ways. One way would be to change the while loop to: while number > 0: Another way would be to change the conditional to: number >= 1 There are a couple others.

### How do I fix my program?

You need to figure out what the program is doing. You need to figure out what the program should do. Figure out what the difference between the two is. Debugging is a skill that has to be practiced to be learned. If you can't figure it out after an hour, take a break, talk to someone about the problem or contemplate the lint in your navel. Come back in a while and you will probably have new ideas about the problem. Good luck.