My code isn’t working!

Basic debugging for CSI and II

About this presentation

Anyone who has tried programming has tried running code that just didn’t work. It can be incredibly frustrating, but also a learning experience.

This presentation will take a look at some of the skills you should work on developing so you can move past the frustration and get to the learning.

There are three important skills to learn:

- Interpreting error messages
- Running code line by line
- Monitoring variables
- “Running code” on paper

Interpreting error messages (I)

There are so many different kinds of error messages that this presentation is not going to attempt teaching them to you. The best way to learn how to interpret them is to experience them for real, anyway.

Pay attention to error messages! Don’t just go “Something was wrong! I’ll change something and see if it gets.” Sometimes that is what you’ll end up having to do, but there are a lot of very informative error messages that you should learn to understand and use.

Error messages can include information about where the program noticed something was wrong, but that isn’t always the place you need to make a change.

Interpreting error messages (II)

If you can’t figure out what part of your code is causing the error message, step through the code line by line until it happens. Then look at the message again to see if there were clues you could have used to find that point faster.

Trying and failing repeatedly is important when learning basic programming, but take the time to think things through before running your code again after making a change in response to an error. Do your changes actually change the conditions that caused the error? Are they compatible with your goal with the code? Are they likely to cause a different error elsewhere?

This might seem hard at first, but it trains the most useful instincts you need to be an efficient programmer.

Stepping through code line by line

The best way to do this depends on the programming environment and the language.

Python is designed to work running line by line, and you can do it by writing, or pasting, your lines one by one in the console, or by using shortcuts in your development environment.

In Java running the code line by line requires you to use the debugging tools in your development environment.

Learning how to use your development environment is always good!

Monitoring variables

Sometimes stepping through the code isn’t enough to figure out why an error occurs, or maybe your code runs without errors, but doesn’t produce the results you wanted.

At this point you need to find ways to see how your variables are changing value as the code runs. Some development environments will let you monitor some or all variables in your programs, but if you are working in one that doesn’t, one approach is to add print-statements that print out variables at different points in your program.

This is especially useful when trying to figure out what is wrong in a loop that runs too many times for you to step through line by line until your error happens.

“Running” code on paper (I)

When your program gets complicated, running through all of it step by step can be difficult. It can also be difficult to figure out where to start programming a solution to a complicated problem. In both of these cases developing skills at writing abstracted or “pseudo”-code can help.

There are programs that let you do similar things without having to find old fashioned pen and paper, but unless your course includes instruction in a specific tool, you are better off using pen and paper.

In the paper version of your code you can gloss over details and write “this function does such and such and returns the correct value” and not have to worry, for now, about how to implement it, or how to make dummy code to avoid errors.

“Running” code on paper (II)

Paper code can use the actual programming language and functions you will be using, it can use shorthand and “generic” commands (pseudocode), it can use flowcharts, or it can combine some or all of these.

Flowcharts are an especially useful tool often overlooked by instructors and students alike. Here is one basic introduction:

https://www.visual-paradigm.com/tutorials/flowchart-tutorial/

(Link does not constitute recommendation of the software tool, it’s just a nice concise introduction to flowcharts.)