6 Tips to Speed Up Debugging (or Avoid It Altogether) in CompSci 101

by Ryan Hardwick

1 Abstraction, Abstraction, Abstraction...

Try to break your problem down into smaller, more manageable chunks, each of which can be solved with a function. Make sure to give your functions meaningful names, so you can call them without having to remember the underlying details.

This may seem like more work in the moment, but it will make debugging go much quicker if and when problems arise.

2 Python Tutor is Your Friend

Although it is not a good idea to copy and paste your whole program into Python Tutor, it can be helpful for visualizing how small parts of your code are executed.

Pro tip: if worried that your bug might be due to aliasing, try enabling the “render all objects on the heap” option.

3 Create Your Own (Smaller) Dataset

If you’re given a problem with a large dataset, take the time to create a smaller dataset (you can even just take a subset of the larger one). Make sure to calculate your program’s intended outputs for the samller dataset by hand.

This will allow you to test your code to ensure that your program is actually working correctly.

4 Be Generous With Print Statements

Like Python Tutor, print statements give insight into what your program was doing at the instant that it was executed. These can be very useful in visualizing outputs to ensure that your code is working as intended at at all points in your program.

5 Algorithmic Problem? Go Back to Pen and Paper

Working instances by hand (and writing down each of your steps) can be a very useful way to gain intuition about how to systematically solve a problem.

Solve the problem on pen and paper with many different inputs before returning to code.

6 Test Code Along the Way

Get smaller parts of your program working first before moving on to other parts. Make sure to test your code with various different inputs to ensure that it is functioning completely as intended.

Pro tip: test your code with edge cases to ensure consistency.