Debugging Tips and Tricks

The Zen of Debugging

Fix one problem at a time

Read your error messages

Making the Most of Error Messages

Error messages are often cryptic and filled with technical jargon. However, we can still get value out of them.

- Error messages occur where the interpreter ran into a problem, *not necessarily where the error is*.
- Error messages usually show the line number of the code where the interpreter ran into a problem. We can use this to focus on the right portion of our code.
- Don’t worry if there are multiple errors, or long error messages. One small error can cause multiple error messages. Remember: fix one problem at a time.
- Find the earliest error message that points to your code. We can’t solve bugs in other people’s modules or libraries, we can only fix our own code!
- You don’t need to read everything in the error message. Error messages are more like signposts giving an indication of where errors may lie, they’re not a map that you must follow.

Check your assumptions

Check Your Assumptions

Bugs often occur when we have a faulty assumption about how our code works.

- Beware the word “should”. If a block of code *should* be doing something, or a variable *should* have a certain value, don’t just rely on what *should* be happening. Check that it actually is.
- Use print statements. Print out variables and check that their value is what you expect.
- Check variable types. For example, if a variable is supposed to have an integer value, it will usually cause errors if the variable is being stored as a string.
- Test your functions. Run your functions on simple inputs and check that the functions return the value you’re expecting.
- Explain your code to someone else (or to a rubber duck). Often, explaining how your code works to someone else will expose an assumption you didn’t realize you were making.

Common Errors

Common Syntax Errors

Syntax errors are very common, and are caused by missing or misplaced characters.

- Missing a single character, like a comma in a list of arguments.
- Mismatched brackets. For example, opening with a parenthesis, but closing with a square bracket.
- Extra characters. This often happens with nested brackets, and you type one too many closing brackets.
- Missing quotes. If you forget to put your strings in quotes, the interpreter will try to read the text as variables and keywords.
- “Smart Quotes”. Be careful when copying code from different editors, sometimes quotation marks get replaced with different characters to show opening and closing quotes.
- Equality vs. Assignment. This happens when you mix up the double-equal (equality) and single equal (assignment) operators, or when you use an assignment operator inside of a dictionary instead of a colon.

EOF Errors

EOF stands for **E**nd **O**f **F**ile. EOF errors are a specific type of syntax error. There are two types of EOF errors: EOF expected, and unexpected EOF

- EOF expected means that the interpreter thought all the code was finished, but then found more code to run.
- Unexpected EOF means that the interpreter reached the end of the file while expecting more code to run.
- EOF errors are often caused by misplaced or disorganized code blocks, especially in languages that require brackets to open and close code blocks (like JavaScript).

Other Common Errors

- Bad index values. When accessing values from a list, passing the wrong index will either access the wrong information (causing problems later in your code), or even cause an error by trying to access values that don’t exist.
- Undefined values. This often happens when a variable name is mistyped, but it can also happen when a variable is created, but never given a value.
- Overflow. Numeric variables have limits to what values they can store. If you try to assign a value that’s too big (either directly, or through a calculation), that’s called an overflow error.
- Scope errors. If a variable is defined in a function or loop, but you try to access it outside of that function or loop, your code will cause an error, either saying that the variable doesn’t exist, or the variable will have an incorrect value.

Debugging Techniques

Commenting Out Code

Commenting Out Code

Commenting code can be used for more than just adding helpful descriptions. Since comments are ignored by the interpreter, lines of code can be “commented out” and temporarily ignored.

- If you can’t find where an error is occuring, you can comment out a whole code block, and then uncomment one line of code at a time, to make sure that each line is working as expected.
- Commenting out code also works well for testing functions. You can comment out the entire code block inside the function and return a test value.
- In extreme cases, you can comment out the entire program, and then test small code blocks one by one.

Google the Error Message

Google and StackOverflow

Whatever error you've run into, you're probably not the first person to have it.

- Using the entire error message as a search term in Google will show you sites where other people talk about running into the same problem.
- Usually, the first few links point to StackOverflow, a site where coders can ask questions, and experts provide answers. Most of the time, you will find the answer you need on StackOverflow. (Don't forget to scroll down past the question! Usually, there will be an answer with a check mark next to it. That's a verified answer, and the best place to start).
- Other links may be to discussions about the error on forums, or rarely to blog posts describing the error (and hopefully the solution).

Rewrite Your Code

Rewrite Your Code

In extreme cases, you may want to rewrite your code from scratch.

- Do not copy and paste! You may inadvertently copy the code causing the error. Type every line out by hand.
- Focus on formatting. If your original code got a little sloppy looking, it might be hiding the error from you. When rewriting your code, make sure everything is formatted nicely, including indentation and code block organization.
-