# **Debugging Code**

Everyone gets "bugs," or unexpected errors, in their code, and this is a normal and expected part of software development. We all say at one time or another, "Why isn't this computer doing what I want it to do?!"

So an important part of coding is "debugging" your code, to remove these bugs. This can often take a long time, and cause you frustration, but developing effective coding habits and mental calmness will help you address these issues. With determined persistence, you can prevail over these bugs!

Here are some tips on successful debugging that we'll discuss in more detail below:

* Understand common error messages you might receive and what to do about them.
* Search for your error message, using the Web community.
* Use print statements.



### **Understanding Common Error Messages**

There are many different error messages that you can receive in Python, and learning how to interpret what they're telling you can be very helpful. Here are some common ones for starters:

* "****ZeroDivisionError: division by zero.****" This is an error message that you saw earlier in this lesson. What did this error message indicate to us? You can look back in the Quiz: Arithmetic Operators section to review it if needed.
* ****"SyntaxError: unexpected EOF while parsing"**** Take a look at the two lines of code below. Executing these lines produces this syntax error message - do you see why?

greeting = "hello"

print(greeting.upper

* This message is often produced when you have accidentally left out something, like a parenthesis. The message is saying it has unexpectedly reached the end of file ("EOF") and it still didn't find that right parenthesis. This can easily happen with code syntax involving pairs, like beginning and ending quotes also.
* "****TypeError: len() takes exactly one argument (0 given)****" This kind of message could be given for many functions, like len in this case, if I accidentally do not include the required number of arguments when I'm calling a function, as below. This message tells me how many arguments the function requires (one in this case), compared with how many I gave it (0). I meant to use len(chars) to count the number of characters in this long word, but I forgot the argument.

chars = "supercalifragilisticexpialidocious"

len()

There are other kinds of error messages that you'll certainly begin experiencing soon in your Python work. Learning what they mean and how to address them will help you debug your code. You might keep an ongoing page of notes on them.

### **Search for Your Error Message**

Software developers like to share their problems and solutions with each other on the web, so using Google search, or searching in StackOverflow, or searching in Udacity's Knowledge forum are all good ways to get ideas on how to address a particular error message you're getting.

* Copy and paste the error message into your web browser search tab, or in [Knowledge](https://knowledge.udacity.com/" \t "https://classroom.udacity.com/nanodegrees/nd027/parts/212819c2-e958-449f-bc17-3f949b04fe9f/modules/8b964443-f29c-45a7-b2e2-330f960b4432/lessons/462708aa-b9f7-49bc-8a99-9504fcf21658/concepts/_blank), and see what others suggest about what might be causing it.
* You can copy and paste the whole error message, with or without quotes around it.
* Or you can search using just key words from the error message or situation you're facing, along with some other helpful words that describe your context, like Python and Mac.

### **Use Print Statements to Help Debugging**

Adding print statements temporarily into your code can help you see which code lines have been executed before the error occurs, and see the values of any variables that might be important. This approach to debugging can also be helpful even if you're not receiving an error message, but things just aren't working the way you want.

We'll suggest particular occasions to use this approach in upcoming helpful places in this course.