

---

title: Getting Help  
subtitle: A never-ending google search  
minutes: 10

---

# Programming: A never-ending google search.

## Learning Objectives

- Explain the difference between knowing a programming language knowing how to program.
- Explain how programming languages can differ.
- Give useful debugging tips.
- Offer helpful resource websites.
- Explain how to google an error.

## What it means to "know how to program"

Most programmers can program in more than one language. That's because they know *how to program* generally, as opposed to "knowing" Python, R, Ruby, or whatever.

In other words, programming is an extendible skill. Basic programming concepts -- conditionals, for loops, functions -- can be found in almost any programming language.

Programming languages generally involve the following elements:

- They give **instructions** to a computer, to store, process, and/or retrieve information
- The instructions center around **things** and **actions**, i.e. data and operations performed on data
- **Things** are items or values you refer to: a number, a name
  - You may handle with them one by one, or in lists, arrays, or more complex data structures
  - Later we'll show how you can use variables to store values as you work with them
- **Actions** are operations you tell the computer to perform: create, delete
  - You may enact them one by one, or combine them into longer procedures, often in the form of named functions
  - Later we'll show how you can repeat operations on multiple things using loops, one of the major building blocks of functions and programs

That being said, programming languages differ from one another in the following ways:

1. **Syntax**: whether to add a semicolon at the end of each line, etc.
2. **Usage**: JavaScript is for building websites, R is for statistics, Python is general purpose, etc.
3. **Level**: how close are you to the hardware. 'C' is often considered to be the lowest (or one of the lowest) level languages.

4. **Compiled vs. Interpreted:** compiles code into an executable program before running, vs. executing instructions directly. C/C++ and Java are compiled, R and Python are interpreted.
5. **Object-oriented:** "objects" are data + functions. Some programming languages are object-oriented (e.g. C++, Python), and some aren't (e.g. C).
6. **Many more:** Here's a [list](https://en.wikipedia.org/wiki/List_of_programming_languages_by_type) ([https://en.wikipedia.org/wiki/List\\_of\\_programming\\_languages\\_by\\_type](https://en.wikipedia.org/wiki/List_of_programming_languages_by_type)) of all the types of programming languages out there.

So what should your first programming language be? That is, what programming language should you use to learn *how to program*? At the end of the day, the answer depends on what you want to get out of programming. Many people recommend Python because its fun, easy, and multi-purpose. Here's an [article](http://lifehacker.com/which-programming-language-should-i-learn-first-1477153665) (<http://lifehacker.com/which-programming-language-should-i-learn-first-1477153665>) that can offer more advice.

Regardless of what you choose, you will probably grow 'comfortable' in one language while learning the basic concepts and skills that allow you to 'hack' your way into other languages.

Thus "knowing how to program" means learning how to *think* like a programmer, not necessarily knowing all the language-specific commands off the top of your head. **Don't learn specific programming languages; learn how to program.**

## What programming is like

NEVER HAVE I FELT SO  
CLOSE TO ANOTHER SOUL  
AND YET SO HELPLESSLY ALONE  
AS WHEN I GOOGLE AN ERROR  
AND THERE'S ONE RESULT  
A THREAD BY SOMEONE  
WITH THE SAME PROBLEM  
AND NO ANSWER  
LAST POSTED TO IN 2003



Here's the sad reality: When you're programming, 80% or more of you time will be spent debugging, looking stuff up (like program-specific syntax, documentation for packages, useful functions, etc.), or testing. This may only apply to beginner or intermediate programmers, but I doubt it.

If you're a good programmer, you're a good detective!

## Debugging

1. Read the errors!
2. Read the documentation
3. Make it smaller
4. Figure out what changed
5. Check your syntax
6. Print statements are your friend

## Googling Errors

- google: name-of-program + text in error message
- Remove user- and data-specific information first!
- See if you can find examples that do and don't produce the error

## Helpful Resources

- [Stack Overflow \(http://stackoverflow.com\)](http://stackoverflow.com)
- [Stack Exchange \(http://stackexchange.com\)](http://stackexchange.com)
- the [D-Lab \(http://dlab.berkeley.edu\)](http://dlab.berkeley.edu)!