Module #6 - Sort of Code with Pseudocode (Required)

In this module, you'll be undertaking an introductory dive into the world of programming logic.

Be warned! This module is one of the most important in the Pre-Work Curriculum, and it is also one of the most challenging. Push yourself to put in the time and effort necessary to complete it fully.

Why is Programming So Hard?

One of the biggest reasons that programming is such a challenging profession is because fundamentally, computers are very, *very* dumb. Much of the complex logic and layered thinking that we humans take for granted would be completely bewildering to even the most complex of computers.

Along this vein of thought, the art of *coding* and *programming* is principally focused on taking complex ideas and on breaking them down into simple instructions that a computer, a machine, or a browser can interpret. Learning to code, thus requires proficiency in the two components necessary to translate ideas into *machine-language*.

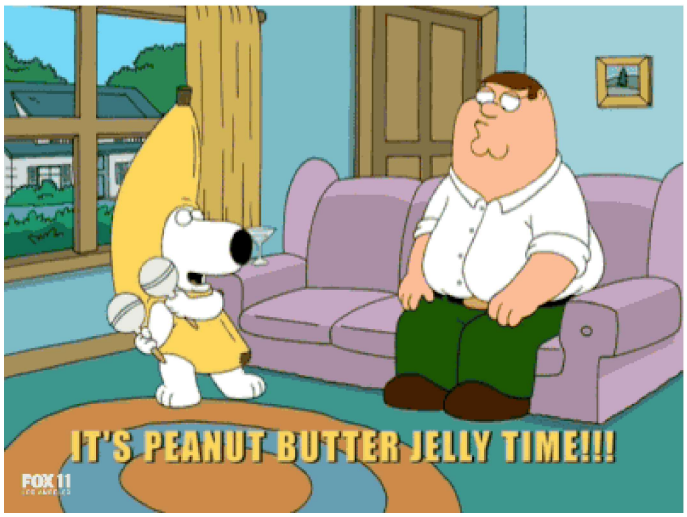
1. The first is that every coder must learn the *syntax* that a computer understands. Syntax is sort of like the grammar and punctuation of machine-speak. It's like knowing that every sentence has a subject and a predicate or like knowing that every sentence begins with a capital letter and ends with a period. In the same way, programming languages come with their own *rule-sets*. These rule-sets may specify a special meaning for brackets or for symbols—or for a particular method for utilizing code across different files. Throughout the course, you'll be learning a *ton* of syntax. It will seem tricky at first, but over time, it will feel like second nature. In truth, syntax will quickly become the least of your worries.

2. Instead, your real challenge will be associated with the second proficiency: converting your ideas into *computerthink*. You see, unlike people, computers are limited in their ability to work with abstract, with vague, or with general ideas. Instead, complex problems need to be broken down into small, discrete blocks of logic.

Think Like a Computer

Take for instance the following challenge:

How do you make a peanut butter and jelly sandwich?



Perhaps you might make a list that looks like the following:

1. *Get bread, peanut butter, and jelly from fridge.*
2. *Get utensils and a plate.*
3. *Slather peanut butter and jelly.*
4. *Combine smeared breads.*

This is all fine and dandy... but really, there's more to it than this.

Take Item #3, for instance: *Slather peanut butter and jelly*. A human might understand this instruction, but a computer or a machine would also need to be told the following:

- *How many times do you slather?*
- *What direction do you slather?*
- *What amount do you slather?*
- *What utensil do you use to slather?*
- *Which do you slather first?*

As you will find out, learning to code often requires one to think far more slowly than we are often accustomed. Instead of rushing through minor details, your computer will force you to slow down and to reason out your thoughts in a sometimes-nauseating number of steps.

But What About the Future?

Unfortunately, until a breakthrough in artificial intelligence emerges, you and I are stuck with our dumb computers. This is good news for us as developers, however, because it guarantees that the skills we possess will be in demand for a good time to come.

In the remainder of this module, you will be shown a video exposing you to the logical building blocks used in almost every programming language (definitely watch this!). You will then take this newfound understanding to create a simple game using the visual programming language Scratch. Don't let the cartoony interface and silly cat logo deceive you. You can build some powerful things with Scratch, and this activity will throw you for a doozy.

Video (Required)

- Introduction to Programming Logic

Assignment (Required):

- Scratch That!

Additional Reading:

- How to Tackle Something You Have No Idea How to Do

Supplemental Resources:

- Getting Started with Scratch
- Scratch Help Guide

Copyright