devbootcamp

Oh Hello There

Thank you for your interest in Dev Bootcamp (also affectionately known as DBC). To help you decide whether coding is right for you, we offer this mini-course free of charge.

Choose your own adventure

A beginner might work through the mini-course in order, watching some videos more than once and reviewing the sample code carefully. A more experienced student might start on the assignments right away, head right to the Interview Prep, or cruise through the videos at 2x speed.

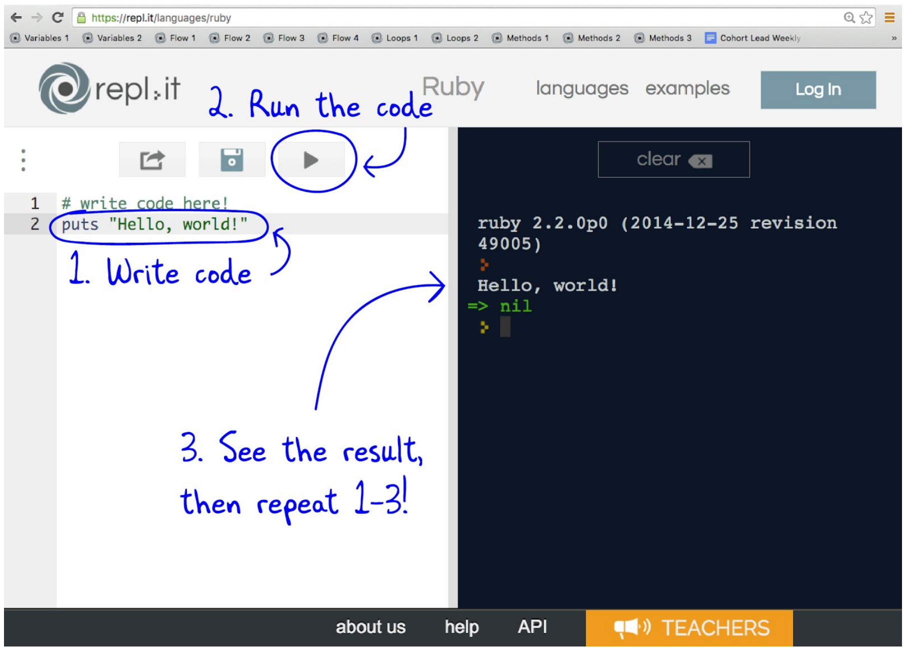
devbootcamp

Getting Started with Ruby

You'll be writing code in Ruby -- a fun, flexible programming language.

In the videos, you'll see the instructor using a tool called a REPL ("repple"). You can follow along using that same tool, which is free to use here. The REPL provides a few ways to write and run Ruby code.

The videos will show you how to use the REPL in more detail, but here's a quick view:



```
# write code here!
puts "Hello, world!"
```

```
ruby 2.2.0p0 (2014-12-25 revision 49005)
Hello, world!
=> nil
```

The assignments in this course are just pre-populated REPLs that you can edit (and, optionally, save). You can start a blank REPL anytime you'd like, and we provide lots of pre-populated sample REPLs throughout the course that you are welcome to edit and experiment with.

Be curious, and have fun! If you make a mess (and we do encourage that sort of thing), you can always start over by returning to this document and loading the original REPL again from here.

1. Information and Memory

Learn

Video (~15 min, can be downloaded for offline viewing)

Experiment

Example code: Variable assignment and debugging basics
Example code: Operations

Try It Yourself

Do your best to complete the challenge below. It's a good idea to run your code each time you add a line or two. (Error messages are normal, but it's helpful to catch them early.)

Assignment 1: Variables are Vary Interesting

Stuck? Looking for some guidance? Email interviewer@devbootcamp.com with your question, and we'll help point you in the right direction.

2. How Programs Make Decisions

Part 1: Basic Decisionmaking

Learn

Part 1 video (~13 min, can be downloaded for offline viewing)

Experiment

Example code: Comparisons
Example code: if/else/elsif

Try It Yourself

Do your best to complete the challenge below. It's a good idea to run your code each time you add a line or two. (Error messages are normal, but it's helpful to catch them early.)
Assignment 2: Global Greetings

Part 2: Repetitive Behavior

Learn

Part 2 video (~8 min, can be downloaded for offline viewing)

Experiment

Example code: until loop
Example code: while loop

Try It Yourself

Do your best to complete the challenge below. It's a good idea to run your code each time you add a line or two. (Error messages are normal, but it's helpful to catch them early.)
Assignment 3: The Mega-Complimentizer

Stuck? Looking for some guidance? Email interviewer@devbootcamp.com with your question, and we'll help point you in the right direction.

3. Reusable Code

Learn

Video (~16 min, can be downloaded for offline viewing)

Experiment

Example code: Designing a method
Example code: Methods with multiple parameters
Example code: Built-in methods

Try It Yourself

Do your best to complete the challenge below. It's a good idea to run your code each time you add a line or two. (Error messages are normal, but it's helpful to catch them early.)
Assignment 4: A Method Ends the Madness

Stuck? Looking for some guidance? Email interviewer@devbootcamp.com with your question, and we'll help point you in the right direction.

Interview Prep

Review

Example code: Even number detector method

Try It Yourself

Mock interview

Stuck? Looking for some guidance? Email interviewer@devbootcamp.com with your question, and we'll help point you in the right direction.

devbootcamp

The Handy-Dandy Freakout Shutter-Downer

All of this is true, so feel free to read any of these when you're ready to sell your computer and join the circus. (Unless you've realized truly *want* to join the circus, in which case, we're very happy for you.)

To be a good programmer, you must first be an awful programmer.

Some students assume that if programming is right for them, it will come easily to them. Programming doesn't come easily to anyone. In fact, good programming looks and feels a whole lot like bad programming.

Learning to program isn't like learning to play a musical instrument or perform a trick on a skateboard; effortless perfection is not the end goal. The most skilled programmers are not flawless; they are resilient. The very best among us still squint at error messages on an hourly basis, and sometimes even a minute-by-minute basis.

Failure is a skill.

Some of the most insightful, intelligent, talented people are downright terrible at failure, having used their sharp wits to neatly sidestep such humiliation for years. But if you want to grow, you have to fail. Learn to succeed at failure. You'll have plenty of opportunity to work on it.

Getting stuck is a wonderful investment.

There will come a time when you've tried everything, and you're out of theories, and your code still doesn't work. You might spend hours or days in this limbo. Think of it less as "getting stuck" and more as "practicing getting unstuck," because that's what you're actually doing. (Feel free to also bitterly refer to it as "building character.")

On the upside, when you finally get unstuck, you will feel a euphoric relief unlike any you have ever known. You'll want to call everyone you know and tell them all about it! Don't do that. It's 3 a.m.

Ignorance is exciting.

What if you woke up tomorrow and there was nothing left to learn? What a depressing idea! There are far worse things in life than discovery. Don’t be in such a hurry to run out of it.

You have a choice. You can view the terrifyingly huge list of everything you don’t know as a burden, or you can view it as a source of emotional wealth you’ll never reach the bottom of.

You’re already somebody’s hero.

If you’ve been programming for three hours, you’re one of the most qualified teachers around, because you know exactly what it’s like. Everything is relative, and it takes no time at all to become someone with something to offer. If you must count the ladder rungs above you, it’s only fair to count the ones below you as well. Celebrate your progress.