An insider view
with alumni stories

David Kim

Choosing A full-time Coding Bootcamp

Choosing a full-time Coding Bootcamp

with alumni stories

David Kim

This book is for sale at
http://leanpub.com/coding-bootcamps

This version was published on 2013-08-23

Tweet This Book!

Please help David Kim by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought https://leanpub.com/coding-bootcamps/ the guide to choosing a full-time #codingbootcamp

The suggested hashtag for this book is #CodingBootcamp.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#CodingBootcamp

Contents

Acknowledgement

Thanks to all of the hard-working people involved in furthering technical education in the world today.

And of course, thanks mom & dad!

Introduction

When you come to a fork in the road, take it. - Yogi Berra

Preface

You are about to start on a fun journey. Do not sit back with this information; instead, actively engage it. Use it as a starting point for further inquiries. Keep learning to code independently of your application process. Keep learning broadly and explore your interests. Certainly, other questions should come to your mind. Feel free to talk to friends or reach out to people at Coding Bootcamps to further your research.

If you haven't already asked, *ask yourself why you want to attend a coding bootcamp*. Are you a career switcher? Are you seeking to add a skill to your existing portfolio? Do you want a convenient, and guided way to learn a new programming language? Or are you simply enticed by the dream of becoming your own technical co-founder, and building the next famous Silicon Valley company?

A Toolkit for Your Decision

I wrote this brief guide as a tool to help you decide whether to attend an intense, full-time coding bootcamp. You may hear others describe it as a hacker school or programming school.

It is a big leap to quit your day job, and enroll in a full-time Coding Program. Should you take the plunge? The decision to attend can be financially expense, and personally disruptive. And there are other related questions on your mind: Does it make sense move to a new City? Will I have a job after I graduate? Who will my classmates be? Do I really want to work as a software developer? How good a developer will I actually become after a full-time program? Couldn't I just learn to program on my own?

You are reading this guide, because you want to be better informed about your decision. You do not want to spend next several months of your life and upwards of $11,000 of investment only to be unemployed, or worse, to regret your decision and still be uncertain about what you want out of life and career. That's why I wrote this guide. I had the privilege of working closely with coding bootcamp staff and with high-tech managers. This guide aggregates insider insights and experiences to share with you.

This guide is also a set of stories, because listening to stories of others can help us navigate our own. And learning to tell stories will be an important component of making your decision as well as succeeding in a job search. I hope that the bits and pieces of stories and advice found in this guide will be useful to you, and save you time in your exploration.

Goals

Remember. Only you can decide what you want out of life! Our role simply is to furnish you with constructive thoughts and useful information to help you in your journey. To that end, our goals are:

1. Provide general overview of Coding Bootcamps and their context
2. Identify and aggregate pertinent information to save you time
3. Introduce stories of other people who took the plunge
4. Curate learning materials to help further your learning process

1 Getting In

Know thyself - Ancient Greek maxim

1.1 Coding Exercises

Exercise 1: Write a function to compute factorial of a given number, *n*, denoted *n!*.

A factorial is a product of all positive integers from 1 to a given number, *n*. For example, 5 factorial (written as 5!) is 5x4x3x2x1 = 120. If you were sitting next to someone who knew very little about programming, how would you begin to outline what you are trying to do? How would you outline the structure of your code? Let's try writing a pseudocode[1]. It is what it sounds like - an outline of a code useful for thinking out loud and communicating with your pair visually. It won't actually "run."

```
factorial_method(argument)
  answer = argument x (argument - 1) x (argument\
 - 2) x ... x 2 x 1
  return answer
end method
```

The trick is to try to build on small successes and make them bigger. First, make sure you can save the argument into a variable and return it. Does it work? Now, build on a layer of complexity by making it compute argument x (argument - 1). Is the returned answer what you expected it to be? And so on.

Exercise 2: Compute 10!

[1]https://en.wikipedia.org/wiki/Pseudocode

Many programmers first write a recursive[2] solution to this coding challenge. Now, use the function you wrote to compute 100!

If you wrote a recursive solution to compute 5! or 10! did it work for 100! or 1000? So, now what?

If a recursive solution doesn't quite work, because of the computing time, how would you restructure your code? Is there a way to compute the multiplication non-recursively from 1 to n? And how does that improve performance? Could you refactor your code further to take advantage of a technique called **memoization**[3]?

Exercise 3: Take n! and find and print the sum of the digits of the answer. For example, for 5!, the answer is 120, and the sum of its digits is 1+2+0 = 3.

You see how we are building on small pieces of existing knowledge. After you'd given it a good faith effort, if you would like to reference a solution (just one among many possible), then check out this repo of Ruby implementation: Factorial Sum[4]

Did you enjoy trying to figure this problem out? Would you enjoy doing so with someone else next to you? Does the process of optimizing your code and make it compute faster appeal to you?

[2]http://en.wikipedia.org/wiki/Recursion

[3]http://en.wikipedia.org/wiki/Memoization

[4]https://github.com/dklounge/euler_exercises/blob/master/pr20_factorialsum.rb

1.2 New Exercise: Listen to Yourself

You either 1) didn't have enough background to successfuly attempt writing a factorial function and the variations thereof, or 2) enjoyed the exercise and want more.

Before we suggest some resources for those of you on path #1 and on path #2, let's pause. It is important to reflect and pick-up signals from this experience about yourself. In our opinion, there is no shame in being frustrated. There is only shame in ignoring the meaning of that frustration. Try to take it as a constructive signal; there is always a path forward. As you will see below, most programming courses select for those who have demonstrated intent and ability to code.

Path #1: More Prep Materials If you need more basics - try this Ruby in 100 Minutes Tutorial[5] by JumpstartLab[6].

Path #2: More Programming Exercises If you found the Factorials exercise too easy, try your hand at some more challenging algorithms using language of your choice by browsing Project Euler[7] exercises.

As you try these exercises & tutorials, rather than focusing on the content per se, observe your feelings and thoughts during the learning process. How does it feel? Does the prospect of seeing bunch of new keywords annoy you?

[5]http://tutorials.jumpstartlab.com/projects/ruby_in_100_minutes.html
[6]http://jumpstartlab.com
[7]http://projecteuler.net/

frustrate you? scare you? or excite you? As the volume of information and difficulty of concepts rise, how do you react? Do you see yourself taking a break and hesitate to return to the material? Or does the exposure to the basic concepts and keywords set you about searching for more detail on wikipedia or Stack Overflow?

Take a moment to reflect. What do these reactions and feelings say about your interests? About your character? The answers to these questions will not necessarily give you predictive information about whether or not you will do well in a full-time programming course. However, we believe these questions are vital to address for you to be happy and successful in life. Your feelings may be a window as to how much you might or might not enjoy a career in software engineering. (There is a small catch to this logic. People tend to enjoy what they excel in; it is easy and effortless. However, that excellence and ease comes as a result of hard - and often frustrating - work!)

1.3 Admissions: Learning From Those Who Went Before Us

To learn more about getting in, talk to two groups of people:

1. People who got in (we share alumni experience in this guide)
2. People who review your applications and conduct the interviews

In this section, let's hear from those who are on the admissions-end of the programming courses. Stories from the alumni are in another chapter. You may see two broad themes emerge on what it takes to get in: 1) you enjoy coding, and 2) you are a good human being.

We asked Michael Kaiser-Nyman, founder of a programming school called Epicodus[8] (Portland, Oregon formerly in Sacramento, California). Michael is an alumnus of Dev Bootcamp[9] and an all-around great guy. Michael noted the following:

> I can't speak for any other coding school, but the two things I look for in Epicodus applicants. First, they should have tried programming before, so that they know that they like it. Second, they should be a good team player.

[8]http://www.epicodus.com/
[9]http://devbootcamp.com/

Sean Daken, founder of RefactorU[10] (Boulder, Colorado) looks for people with high general intelligence + emotional intelligence + professionalism.

> We want people who are passionate about the path they are pursuing, who have a sense of urgency, and who have maturity in spades. Successful applicants need to be able to communicate very well, be driven and willing to work hard, and be able to learn and apply new concepts quickly. Basically, we're looking at applicants through the lens of a hiring manager - smart, driven, passionate, very strong communicator, etc. We actually score applicants quantitatively and qualitatively based on perceived raw general intelligence, communication skills, professionalism, passion, how much they have invested up to the point of the application in learning on their own, etc. We determine the score based on a combination of their application responses, interviews, and in some cases reference checks.
>
> It helps to have some prior coding experience, but [it is] not absolutely required. It is important that people have tried to learn something related to programming on their own, not so much for what that is in particular but that it shows that they have taken initiative and

[10] http://refactoru.com/

> are serious. The best candidates have all of the qualities I mentioned above AND have tried to learn as much as possible on their own. They know what they want and know that RefactorU is the most efficient path to getting them there.

Shawn Drost, a co-founder San Francisco-based Hack Reactor[11] lists the following key attributes during the admissions process:

> - **Drive** - We want to work with people that will push themselves and their peers to surprising amounts of success.
> - **Warmth** - The school ends up being like a family, and my cofounders and I go to crazy lengths for our students that don't make any sense from a business perspective. Life is too short to spend time with assholes, even when they're all-stars by other metrics.
> - **Effectiveness** - It turns out that getting things done is a unique skillset that carries over between careers/environments. We like effective people – those skills apply just as well to software engineering.
> - **Intellect** - Raw horsepower also matters.

[11]http://hackreactor.com/

- **Technical Chops** - Many intelligent people will never become software engineers, either because they don't actually have a deep interest in the field or because it just doesn't align with their mindset. We ensure that applicants have learned the first 10% on their own, mostly so that we know for sure that they can clear those hurdles.

From Hacker School's blog about What We look for in Students[12], one can list the following key attributes:

- You enjoy programming
- You want to get significantly better
- You're friendly
- You're self-directed
- You're intellectually curious
- You have a demonstrated capacity for rigor
- You're introspective

Summary

You might be a good candidate for a coding bootcamp, if you are: * coding already, and eager to learn more * friendly person who plays well with others in a team

[12]https://www.hackerschool.com/blog/20-what-we-look-for-in-students

1.4 Geography: Does It Matter?

You might be wondering, does geography matter at all? Jeff Casimir, principal at JumpstartLab[13] shared his views on Quora. Asked what the advantages of attending a programming course in San Francisco Bay Aarea might be:

> I'm obviously biased, but will give you a quick answer anyways:
>
> Not being in the SF ecosystem has plusses and minuses. On the downside, if you're in SF, you're a little minnow in a giant pool. Your cost of living during the program is super high. People probably haven't heard of the program you're in.
>
> On the other hand, they're craving developers. You can't turn around in the coffee shop without bumping into someone hiring. You can easily interview in person rather than scheduling trips or doing it over Skype.
>
> Being anywhere EXCEPT SF has minuses, too. Here in Denver there are dozens of jobs, not hundreds. Being here you're not totally immersed in the tech culture – people are more likely to talk about mountain biking than slinging code.

[13] http://jumpstartlab.com/

> On the upside, there are way more job here than our students will fill up. I expect about 12 of our 23 students to stay in Colorado. You're a small fish in a small pond – people have heard of our program, they've been to our office – essentially our program is the most interesting thing going on in Denver tech. Outside of Colorado, people are coming from SF, Seattle, Portland, NYC, etc to interview and (hopefully) hire our students.
>
> Long story short: I don't think it matters. Pick a program that fits you and has people you like, the job part is easy.

It is an interesting insight. My personal opinion is that location actually matters quite a lot. Certainly, it matters for your personal wishes - you should be where you are happy. Yet, location also has a big professional impact, even in this modern age of GitHub and Google Hangout. Many attend coding bootcamps with the intent of finding a job in software. Geography matters, because location (e.g. San Francisco) directly impacts 1) quantity of jobs, 2) quality of jobs, 3) compensation range, 4) community of like-minded peers, and 5) density of industry experts from whom you can learn more. Clearly, each one of these factors are very important, not to mention combinations of them.

Again, I am quite biased in thinking that San Francisco is one of the best places you could come to attend a full time programming course. That said … move here!

1.5 Class Size & Your Day In Life

Certainly, this is one of the attributes you should learn about the programming course. (See our Appendix of Courses for a starting list.) We won't go deeply into this topic as it is self-explanatory. Hopefully, most of you gave some thought to what the learning environment will feel like in a group. We just want to make sure we debunk the somewhat romanticized myth of a loner programming genius.

A Ruby Rookies Meetup

The truth is, that unless your goal is to work as a solo programmer-contractor or as a solo entrepreneur, you will be working in teams. And you will have a boss - whether that boss is a manager or a customer. That point is probably well understood. So, if you've never done it before, how can you simulate programming in a group? One easy way would be to join a local coding meetup. The quality of this technique depends on where you live, but the point is that you are making an effort to simulate what it might be like to join a 3-month long programming course, bootcamp-style, and work day and night with your peers, discussing lines of code, listen to lectures, walk out to lunch together, come back and do more coding. Does that sound fun to you or not? It is very important to imagine yourself in this setting. It is not to say that if the idea doesn't appeal to you, you should quit this journey. Rather, the intent here is to help you become more self-aware and to suggest a technique you can apply to simulate the experience (find a friend to pair, join a meetup group, start a study group, etc.).

Building on that note, what would your day-in-life would look like at a programming school?

The following is a link to day in the life from a well-known San Francisco programming course called Dev Bootcamp: A Day in the Life[14].

As a side note about solo programmers, they do exist. Talk to enough companies, and you'll find that there is bound to be a "mad scientist" type of roles!

[14]http://devbootcamp.com/2013/02/11/a-day-in-the-life-at-dev-bootcamp/

1.6 About Changing Trends

Nearly nothing in life is constant, and this is especially so in high tech. This also is true of programming courses; they are constantly changing. Everything about them! The curriculum is evolving, the applicant talent pool and admissions criteria are rising. Even the employer needs for technical talent grows and wanes. One of us was an early applicant to a San Francisco-based programming course (was admitted, and decided not to attend).

There are some takeaways from the changing trends:

1. If you have resolved your doubts about attending, there is value to applying early. Either the competition will be thinner now (compared to future), or you will have a chance to measure yourself sooner and update your skills or career expectations accordingly.
2. If your ultimate dream is not to work as a software engineer, then you may want to step back for a "gut check." Is this what you really want to do, or are you chasing a trend? Remember, what is cool, or in demand today may not be tomorrow.

2 About the author

2.1 Follow David @findinbay

David Kim

David Kim organizes Ruby Rookies[1] and Product Management Fast-Track[2] in San Francisco.

David previously was the head of business development at Hack Reactor[3]. David interacts extensively with top-tier

[1]http://rubyrookies.github.io/
[2]http://www.meetup.com/productlovers/
[3]http://hackreactor.com/

companies in the Bay Area. David is a life hacker who solves problems and enables people in the community.

David shares his experiences and musings on Medium[4] and on blogger[5]. You can follow him on Twitter: @findinbay[6] or Github: dklounge[7]. Ask him a question on Quora[8], or join him on a long run with Anyone Can Run[9] San Francisco.

[4]https://medium.com/@findinbay
[5]http://findinbay.blogspot.com
[6]http://twitter.com/@findinbay
[7]http://github.com/dklounge
[8]http://www.quora.com/David-C-Kim
[9]http://www.meetup.com/Anyone-can-run-SF/

Appendix: Prep Resources

Further Reading

Useful Quora Threads

- Follow Programming Bootcamps topic on Quora[10]
- Quora entry: admissions critera[11].
- Quora entry: best US programming courses[12]
- Quora entry: Who hires junior developers from programming courses?[13]

Student and Alumni Blog

If you have good ones to add to this list or would like your story featured, please write me!

- Cirles and Dots[14] - Michelle Glauer, a Hackbright alum (San Francisco)
- Natasah the Robot[15] - Natasha Murashev, a Dev Bootcamp alum (San Francisco)

[10]http://www.quora.com/Programming-Bootcamps

[11]http://www.quora.com/Programming-Bootcamps/What-are-the-chief-admission-criteria-for-those-admitted-to-programming-schools

[12]http://www.quora.com/Computer-Programming/What-are-the-best-programming-bootcamps-courses-available-in-the-United-States

[13]http://www.quora.com/Programming-Bootcamps/Who-hires-junior-developers-from-programming-bootcamps-like-Hack-Reactor-or-Dev-Bootcamp-with-less-than-a-year-experience

[14]http://michelleglauser.blogspot.com/

[15]http://natashatherobot.com/

- Logic Mason[16] - Mark Wilbur, a Hack Reactor alum (San Francisco)

Practice, Practice, Practice

So, there's really no substitute for writing lines of code. I always tell friends "keep pushing that code." If you don't know `git push` yet, you will. Here are some of my favorite exercises for beginners to begin writing and pushing actual lines of code to GitHub (as opposed to reading or coding in browser-only).

- ProjectEuler.net[17] - somewhat "mathy," but these will have you getting comfortable with all of the basics of programming, from data types and control flows to data structures.
- CodeEval[18] - similar to ProjectEuler, though a key difference is that many challenges are actually sponsored by hiring companies. Pass these, and you get a chance to hit an **apply** button to real jobs.
- RailsApps Tutorials[19] - this tutorial is written by a great SF teacher, and I highly recommend it! You can follow Daniel on twitter @danielkehoe[20]

The following sections will list additional resources. The truth is that you won't have time to look at them all. It also

[16]http://logicmason.com/
[17]http://projecteuler.net/
[18]https://www.codeeval.com/
[19]http://railsapps.org/
[20]https://twitter.com/danielkehoe

goes to show just how much information is out there in the world to teach you how to code. You have no excuse! :) Survey the resources, pick a few that you think would work for you, and then keep at it!

Keep pushing!

Beginner Coding Tutorials

- rubymonk[21]
- Code School[22]
- Ruby on Rails Tutorial by Michael Hartl[23]
- SQLZoo[24]
- JumpstartLab Tutorials[25]
- Mozilla JavaScript Guide[26]
- Learn Python the Hard Way by Zed Shaw[27]
- Peep Code Screencasts[28]: subscription-based website where you could choose from lots of technologies and frameworks, e.g., RoR, JS, NodeJS.
- ScreenCasts.org[29]: emerging JavaScript screencast hub.
- Backbone Screencasts[30]: all the things BackboneJS on a pay-as-you-go basis.
- NodeTuts[31]: Node.js Free screencast tutorials for NodeJS.

Video Tutorials

[21]http://rubymonk.com
[22]http://codeschool.com/
[23]http://ruby.railstutorial.org/
[24]http://sqlzoo.net/wiki/Main_Page
[25]http://tutorials.jumpstartlab.com/
[26]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
[27]http://learnpythonthehardway.org/
[28]https://peepcode.com/
[29]http://screencasts.org/
[30]http://backbonescreencasts.com/
[31]http://nodetuts.com/

- Railscasts by Ryan Bates[32]
- The Intro to Rails Screencast I Wish I Had[33]
- Udacity CS101 with David Evans[34]

[32]http://railscasts.com/
[33]http://youtu.be/cMcEgOPza8A
[34]https://www.udacity.com/course/cs101

JavaScript Tutorials

- Eloquent JavaScript: A Modern Introduction to Programming[35]
- Developing Backbone.js Applications[36]
- Step by step from jQuery to Backbone[37]
- Douglas Crockford on JavaScript[38]
- Leaning JavaScript Design Patterns[39]

---

[35]http://eloquentjavascript.net/

[36]http://addyosmani.github.io/backbone-fundamentals/

[37]https://github.com/kjbekkelund/writings/blob/master/published/understanding-backbone.md

[38]http://javascript.crockford.com/

[39]http://addyosmani.com/resources/essentialjsdesignpatterns/book/

Coding Blogs & Newsletters & Other Sites You Should Know

- DailyJS[40]
- JavaScript Weekly[41]
- JavaScript.is (Sexy)[42]
- Superhero.js[43] - A collection of best stuff on JS
- DevDocs[44] - All-in-one API documentation
- stackoverflow[45] - Get your coding questions answered
- github[46] - Brose open source projects and start reading other people's code

[40]http://dailyjs.com/
[41]http://javascriptweekly.com/
[42]http://javascriptissexy.com/
[43]http://superherojs.com/
[44]http://devdocs.io/
[45]http://stackoverflow.com/
[46]https://github.com/

Massive Open Online Courses

Use these resources to learn programming and beyond:

- Udacity[47]
- Coursera[48]
- edX[49]
- Open Classroom[50]

These websites offer a wide variety of classes at relatively small cost:

- Udemy[51]: lots of discounts; courses on Lean Startup methodology.
- Online Marketing Institute[52]: all things related to online marketing; subscription-based.
- Khan Academy[53]: short videos primarily on high school subjects.
- iTunes U[54]: mostly videos and textbooks from top universities like Yale.

[47]http://udacity.com/
[48]http://coursera.org/
[49]https://www.edx.org/
[50]http://openclassroom.stanford.edu/MainFolder/HomePage.php
[51]http://udemy.com/
[52]http://onlinemarketinginstitute.org/
[53]http://khanacademy.org/
[54]http://www.apple.com/education/itunes-u/

- Lynda[55]: paid business and software-oriented courses with homework, labs and tests.

[55] http://www.lynda.com/