HOW TO GET A JOB IN PROGRAMMING WITHOUT A DEGREE

I lost a year of time trying to learn my first programming language. It wasn't something that was initially apparent at the time either. Most people suggest that one wastes the most time when it comes to productivity and education through the distractions around you. Whether it be video games, or sometimes even the people around you. I'm here to tell you that that couldn't be further from the truth (and please, do talk to your friends and family!)

The biggest waste of time comes from when you think you're being productive, but you're simply not. It comes from that feeling of accomplishment that drives you forward into nothing but phantom work. Undirected work can be twice as bad as directed leisure: none of the pleasure, and none of the results.

You can't afford to lose a year of learning if you're looking to break into a programming job without a degree. When you sacrifice social proof, the stability, and prestige of a college degree, you will often have to work twice as hard just to make it. This guide doesn't make any promises beyond saying that it will take a lot of sweat and determination to get to where you need to be and that comes from a place of genuine honesty. It comes from a fellow learner who struggled, just as you probably do now, at finding out how to apply your skills to a future career or passion.

I didn't choose the right programming language to learn for a year. In fact, I often switched from one framework to another, from Ruby to JavaScript, from Codecademy to Khanacademy in the futile search for beneficial knowledge. Without a plan or an idea of what I was doing -- all these efforts quickly fell to uselessness. I spent a year learning programming languages and like phantoms, they vanished into obsolescence. I was no closer to building something meaningful after a year spent learning code than I was when I began. I was no closer to a career where I could use programming skills than a year before.

I wrote this guide because I didn't want the same thing to happen to you, and my struggle be in vain. After that year of wasted struggle, I started looking for a new, structured approach to tackle my learning. Though I'm still in the midst of refining it, the knowledge I've acquired fits my interests and has been useful to me in both my work and my passions.

At the end of this guide, you'll know how to program in a meaningful way.

I want to tell you that there can be a happy ending. I've also moved on to help people from non-technical backgrounds get jobs in data science and UX design with Springboard, a cutting-edge 21st-century school for digital skills. I write content and apply my Python skills to problems every single day. The same can happen to you. If nothing else, my story proves that it is possible.

I wrote this guide so that you knew exactly which programming language you should learn and how you should learn it so that you can direct your energy towards learning that matters. For most people, that'll be attempting a voyage to a lucrative, yet fulfilling career. For some people that'll mean creating their own company with the technical skills they've mastered. Whatever it might be for you, I hope this guide can help you accomplish the goals you've set.

I hope it's helpful! Reach out to me at thinkthethoughtbasin@gmail.com if you have any questions. I wish you the very best on this journey of learning.

Frequently asked questions/concerns

Before we get started, I want to address a few questions or concerns that come up, especially as people are looking to start self-learning programming. Consider this a section to address frequently asked questions before we explore programming in-depth.

Do I need to know advanced math?

No. For certain fields of software engineering and certain concepts in specialized fields, you'll want to have a good grasp of linear algebra, calculus and other fields of math -- but there are many technical fields that don't require the use of math. Web developers, for example, often only require a simple grasp of basic arithmetic in order to build full websites.

Is it impossible to learn computer science without a degree?

Far from it. Many top technologists don't have a formal degree in computer science. In fact, in the past, most people self-taught themselves new and emerging technologies, and it is much the same with new and emerging fields. Oftentimes, academia struggles to keep up with cutting-edge practices, so people looking to learn new fields band together and create resources for themselves. Much like those proactive learners, you can do the same.

Getting a degree in computer science can force you to acquire the knowledge you might not have otherwise gotten through a self-learning process. You might learn about machine code and how compilers work -- something that can be important down the line when you're dealing with complexity that requires knowledge of the fundamental principles of programming.

If you want to be a complete programmer, you might want to go out of your way to look for things that those formally trained in computer science pick up that autodidacts usually aren't exposed to. This Quora thread will help in that respect. While there is value in getting a formal education in computer science, you could certainly become a programmer and get technical jobs without it -- what matters most isn't a degree but the amount of passion you dedicate to getting better at programming.

Do employers discriminate against computer programmers without computer science degrees?

Certainly, if you look through different technical job descriptions, you might notice that most require a minimum of a bachelor's degree in computer science or a related technical field. However, that doesn't mean that it's impossible to get a job in programming with an unrelated degree. The secret is to a) find ways to demonstrate that you have technical skills, either through contribution to open-source projects or through the creation of your own projects and initiatives and b) find the right employedavid rs willing to take a chance on somebody who is self-taught. There are certain employers that have programs for talent -- if you're an underrepresented minority in technology, for example, it is possible to be highlighted in this way.

You'll also want to look for earlier-stage companies, smaller startups in need of people who are passionate about their mission and which may be more permissive about learning on the job.

So while not having a degree can sometimes be a negative factor, you can counteract that by showing tangible examples of your technical skills and by choosing the right companies to work with.

What is the best programming language to start learning first?

The best programming language to learn first - the question that often pops up for a lot of people looking to take their first steps into programming.

The answer is really dependent on what you're looking to do with your programming skills: different programming languages have different purposes, and depending on what you're looking to do, the answer can vary widely.

Most universities now teach Python in undergraduate computer science degrees, mostly due to its versatility, clear and simple syntax, and its ability to encapsulate all sorts of programming paradigms, from object-oriented to functional.

The real answer to this question, however, is that the best programming language to start learning first are the underlying logical principles behind the code and the different syntactic forms that logic can take to be expressed. Once you master the governing principles of programming, you can decide exactly what language to use for different purposes. Instead of learning French, learn the underlying principles behind the grammar of Latin languages -- then decide if you're going to speak French, Spanish and Italian -- or some combination thereof!

Now that we've addressed some of the common questions and concerns out there, let's dive into how you can learn to program. **Those of you who have a grasp of the basics and don't need a refresher should feel free to skip ahead to other sections.**

How to learn to program effectively

One of the most popular online courses in the world isn't about programming. It's not about some hot technical skill. It's about the meta-pursuit of knowledge -- how to learn about learning. In this TedX video, the course creator, Barbara Oakley (now a professor of engineering) delves into the most effective learning topics of all time: just exactly how to learn. As an autodidact herself who realized late in life that engineering is what she wanted to do, it's an especially poignant and relevant story.

From this and books on learning (I highly recommend books such as Made to Stick and Moonwalking with Einstein if you want to understand the principles behind learning and memory), I've isolated the following principles for you to learn programming (and with a little bit of generalization, anything at all) effectively:

1- Place yourself in situations where you're forced to practice

You may have heard the saying that "practice makes perfect" -- but that's only half the battle. Sure, practice makes things better -- but how do you place yourself in a position where practice not only becomes mandatory but something you're compelled to do? How do you force yourself to always learn and improve at the fastest pace possible?

The answer for many people is consistency laden with incentives. The perfect mix involves either programming as a professional pursuit or a series of projects that you are strongly compelled to build. In order to practice your programming skills, you'll want to be employed or working on a significant side project or your own company so that your programming skills are adequate for your day-to-day success.

Design yourself a context where you have to practice programming every day. If a programming career is what you want to do, don't be half-hearted about it.

2- Tackle things that are challenging, but not impossible

One of the first things you'll learn about learning is that you'll want to set yourself challenges that are just hard enough to motivate you, but not so hard so as to seem impossible. Make sure that you keep the right balance on your learning so that you're always moving forward with purpose.

3- Learning takes time. Learn to accept that and schedule your growth

There are a lot of different courses out there that promise that you can "learn to program in X amount of days or weeks". The truth is, everybody has their own learning pace and learning itself is accretive: it tends to be an accumulation of practice and slowly absorbing material throughout a time period.

You might have experience cramming last minute for an exam. You can be sure that this process is detrimental to learning effectively. Only by learning as time slowly passes by, day-by-day, can you truly build mastery of a skill like programming.

4- Get all the way to first principles

Elon Musk, the entrepreneur who through his companies SpaceX, Tesla, and SolarCity has sought to conquer private space exploration, electrical cars, and renewable energy was asked how he learned so rapidly through so many different domains once. His answer?

He stuck to learning the fundamental principles behind each domain, representing these essential concepts as the roots of a large tree of knowledge. By hanging new nodes of knowledge onto a sturdy mastery of those fundamentals, and by transferring knowledge through pattern-matching from other domains, Elon Musk was able to quickly master different concepts and build world-changing companies in each domain.

Drill your way down to the fundamental principles of any field of knowledge until your mastery is so solid you can incorporate new and more specific knowledge into it.

5- Have a "growth" mindset -- and believe in yourself

There are many resources out there on the difference between a "fixed" and "growth" mentality. It was Mindset, the seminal book on the psychology of learning from Stanford University psychologist Carol Dweck that first broke ground on this concept.

In essence, a fixed mindset regards the challenges of learning as something to be avoided. A fixed mentality stresses that a human being is a static canvas that cannot be changed, only challenged. Instead of looking at the initial stage of learning as a great opportunity to be reinvented, people with a fixed mindset only see the potential of shame.

People with a growth mentality know that learning involves a lot of difficulties. They know that they'll be embarrassed as they look to master new things -- but they don't care. They know that the costs of learning are well worth it, and they know that eventually, with the will to continued practice, their initial stumbles will become the first steps to mastery.

Embrace the growth mentality and the constant call to self-improvement that it implies. Be fearless in your drive to learn new things, and shameless for the costs it might entail.

First principles of programming

Let's put those lessons on how to learn to use and start with the first principles of programming!

The thing that most people don't get about programming is that it isn't the individual syntax differences between language that pose the largest difficulties. It's really the underlying principles behind code that are the most difficult to grasp. Once you're immersed in the logic behind the code, you can quickly pick up several languages at once -- if you've studied the commonalities behind programming languages and learned how each one expresses that underlying logic, you can quickly become an adaptable and rapidly learning "renaissance"

man -- or woman -- the ideal state if you want to **find a programming job or start your own company.**

PEER BENEATH THE MATRIX

Here are the principles that will get you to look beyond the skin-deep syntax differences, and recognize the underlying strands of logic that lurk within.

1- **Programming is about manipulating data**

At its core, programming is about moving data and playing with it. When you send login credentials to a web server, or when you get your profile picture loaded, that's code sending data back and forth. If you can recognize that moving data is the foundation of programming, you can understand the basement from where you will build your house.

2- **Programming is like writing -- you want to be as clear as possible**

When you're writing code, you may think of it as an individual activity. Nothing could be further from the truth, despite every stereotype Hollywood has thrown at the programming archetype. When you're programming, you'll want to imagine that you're writing for an audience. Programming is a collaborative activity, one that often involves working with close teammates or other collaborators.

3- Know the different types of programming

Do you think of programming as one generic blob or can you differentiate it into specialized parts? Saying that you want to be a programmer is a bit like saying you want to be an engineer -- the devil can be in the details.

We'll go into more details about the broad technical paths you can take as a programmer in different paths later on, but for now, we can distinguish some broad differences.

The largest difference in types of programming can be thought of as the difference between "front-end" and "back-end" programming. Front-end involves manipulating what a user sees directly: think of the interface you see when you log in to any web interface. The back-end is all of the magic that happens sight unseen -- the way servers process your password and grant you access to all of your data.

KNOW THE DIFFERENT PARADIGMS

There are also different types of programming paradigms: different ways to express logic, and different functionalities for each programming language that at an aggregate level, can be summed up into categorical differences.

Here is an overview of the major paradigms in programming:

- **Declarative.** Declarative programming is very simple and plain. It expresses the logic of a particular computation without specifying its flow. The easiest way to think about it is a programming language that ***declares*** **what task** is being done rather than how it should be done. Examples of this include programming languages like SQL, whose syntax is focused on explicitly specifying exactly what you want as opposed to specifying how it's done (ex: the SELECT command which selects data). The underlying steps behind the SELECT query do not have to be explicitly defined for the machine to act upon its underlying logic.
- **Imperative**. Imperative programming focuses on how a task is being done rather than what is being done, unlike the declarative model. Much like how in the imperative mode of language where commands are given, the imperative programming paradigm describes to machines how they should carry out a task. Imperative languages include languages such as Java, JavaScript, and Ruby, though all of them also have object-oriented logic as well -- and most of them are multi-paradigm languages that are compatible with a variety of programming paradigms.
- **Functional**. Functional programming is based on mathematical functions. While here too, commands are meant to specify how routines are carried out rather than what routines are carried out, unlike in the imperative paradigm, the state of a current program cannot be affected incidentally: what this means in practice is that you can have functions without return calls, since the program state will remain constant. Functional programming is emphasized in academia with languages such as Lisp and Clojure prominently supporting functional programming as a paradigm.
- **Object-oriented.** The dominant programming paradigm since the 1980s, object-oriented programming involves building objects with data attributes and programming subroutines known as methods which can then, in turn, be invoked or modified. Languages such as Java, Python, C, C++, PHP, and Ruby are all principally object-oriented. Critically, unlike imperative or functional programming, the concept of inheritance and code reusability are firmly entrenched in programming objects which can persist either as classes (the definition of how a set of objects is defined, and what data they can carry) or objects themselves (which often correspond to real-world objects and a collection of attributes associated with them).

There are also programming paradigms, mostly a bit outdated or a bit theoretical, that revolve around procedural programming, logic programming, and symbolic programming. Research those if you have the time, and compare and contrast!

4- Programming is about forced simplicity

```
Simple is better than complex.
```

```
Complex is better than complicated.
```

```
Flat is better than nested.

Sparse is better than dense.
```

-The Zen of Python

One of the founding principles of effective programming is a sort of forced simplicity that becomes natural with time and iteration. Perhaps perfectly summarized in the Zen of Python, simple is better than complex -- and complex is better than complicated. Much like good writing, which sometimes requires focusing on the right ideas and cutting out as many unnecessary words as possible, good programming means keeping the underlying logic expressed simply so that it can be readable to others and perhaps most importantly, your future self!

5- Small efficiencies lead to large gains

When you're dealing with a machine that can perform complex operations in a matter of seconds or sometimes microseconds, it can be hard to comprehend from a human perspective exactly how to manage the efficiency of such a system. Humans are notoriously bad at exponential thinking and programming depends upon that very strand of thought. A few microseconds of difference in one operation can mean a difference of thousands of hours when extended to a chain of operations that extends past the trillions.

There are three concepts here that can help you manage that complexity.

Time complexity

Become familiar with the concept of time complexity in programming and specifically big O notation. Put simply, Big O notation maps out a pattern of how an algorithm will respond to a given set of inputs.

O(1) algorithms react the same regardless of what input size they're fed. You could input one value or a trillion: -- it doesn't matter, the algorithm will process at the same time. A common example of this is the *return or print* algorithm in most programming languages.

O(N) algorithms react linearly to the inputs they're given. A million data points? Expect it to run a million times slower than with just one.

O(N^2) algorithms react exponentially to the inputs they're given. Every input gets squared as it's processed. Think of an algorithm that has to look over data twice on each iteration. With a dataset that's twice as large as another, you'll quickly see an exponential burst of time when it comes to how slowly that algorithm is going to deal with differently sized datasets.

This goes on across a variety of configurations: this tutorial has a fuller list and a more comprehensive explanation. The takeaway is that you should always strive to use algorithms that scale as linearly as possible, otherwise large datasets will become unmanageable.

Modularity

One of the greatest things about programming is the ability to extend logic that has been saved before. You don't have to reinvent the wheel each time: if you've built a component that works, you can call upon it at any time.

This simple concept of having pieces of code saved that can be called upon like a mix of components is called modularity, and it is an essential time-saving device for individual projects. Modularity also allows you to save time in collaboration. You can work with different people on a complex problem and work on a single piece then have your module interact with those built by others to form a larger solution.

Grasping the nuances of how to make your programs modular will not only save you time, it will make your programs stronger and more readable, and less dependent on large blocks of code that can prove to be riddled with bugs.

Shortcuts

Use shortcuts whenever you can! Programming efficiently involves putting your thoughts and logic into machine form as quickly as possible. This table of keyboard shortcuts will help you speed yourself up. Games such as TypeRacer will help you improve your words per minute count when it comes to typing, allowing you to transfer your thoughts more quickly into code.

Mastery of shortcuts will allow you to build things quicker and see results sooner rather than later.

6- Practice makes perfect

It's often said that there are two ways the human mind views growth: either the fixed mentality or the growth mentality. In the fixed mentality, human growth is a function of destiny: no matter how hard you try, you can never grow beyond what innate factors have prescribed for you. The growth mentality is the total opposite of the fixed mentality: here, your growth is only limited by time and your will to do something great.

Nobody is naturally born a better programmer than somebody else. You have to work hard and put in the hours if you want to improve your programming skills.

This extends to programming interviews, a necessary evil. The bar is much higher in those than the normal conditions you'll have gotten used to: ample time and plenty of resources such as StackOverflow to help you along the way (more on that later). A programming interview is designed to stress test you. You'll have to get used to practicing a form of programming under constraints, both natural and artificial: do timed sets of problem-solving, and rely on nothing more than paper to sketch out a few algorithms.

7- Create programs that are flexible to different needs.

One of the largest difficulties in programming is the need to create things that can flex and which don't break even under what are called "corner cases": different uses of your software that may challenge the extreme limits of the variables you've set. The easiest corner case to think of is an app that has gotten very popular and must support thousands of users at once. Even the most elegantly crafted code will begin to strain as that happens.

8- Build useful things. Have empathy and understand the problems of your users.

Most people forget that technology isn't just a skill to use for fun. Technology is a means to an end. Don't forget that you're not building for the sake of building: you're building for an end user or to solve a meaningful problem that cannot be addressed without technology. Know when to build something with code, and know when you don't have to.

If somebody has already built the wheel, there's no need to reinvent it. Plenty of people have built solutions to address how to store sales contacts -- but are there problems out there that remain unseen? Tons! And each could be the foundation for a successful project.

Build useful things with your programming skills. It's the best way to use your skills for good and to highlight them to others.

Now, what does useful mean? That's the crux of the problem. In a world filled with apps for every possible luxury and consumer need, it can be hard to see exactly what useful means. In this context, **it is important to practice empathy and to diagnose real problems that can be helped with technological solutions**. This, notably, does not have to be confined to problems that can be solved for profit. There is much room for technology for social good initiatives or for technologists to solve social problems. This case study of data science work for international development/basic income charity GiveDirectly provides a great example.

In order to become the most impactful technologist possible, it's important to understand what problems people have -- it's important to know when and how to ask the right questions, how to listen and to uncover the underlying problems everyday people face. With the right amount of empathy and perspective, you can **build maximally useful things that help solve problems for others.**

9- Take advantage of as many resources as possible, and give back when you can

Go onto communities such as Hacker News and Quora**.** Look up questions on StackOverflow -- and even ask a few yourself if you're stuck on anything! Look up awesome Github repositories that contain all of the resources you'd need to learn one particular topic: see, for example, Awesome Python for a list of curated resources in the Python space.

And don't be afraid of failure. You'll be learning along the way, practicing your skills and becoming better with every passing day regardless of the outcome.

Building blocks of programming

One of the first principles of programming is that the languages and syntax you express your logic with don't really matter as much as the underlying logic itself. Here are some fundamental building blocks of logic that you can port to different programming languages. We're going to have some code examples from different languages -- don't panic if you haven't seen them already! Our next section will cover different languages in depth.

This section will use what is called *pseudocode* to express logical concepts. Think of them as a rough set of instructions you want to assign to your computer -- in many ways, that is the essence of programming itself! This pseudocode can then be implemented in various programming languages once you understand the different syntactic variations within those languages.

Data Types

You have to deal with different data types as you interact with different programming languages: fortunately, most programming languages share the same fundamental principles when it comes to how to organize data.

Common data types include:

- **Strings.** Think of a sequence of words, such as this text. The data contained within would commonly be organized as strings in programming languages, with quotation marks wrapped around them. "I am walking around" would be a defined string in a programming language.
- **Integers.** Integers are the default way to programmatically store numbers and numerical values. Importantly, integers can work with mathematical functions and can be aggregated by them, unlike strings -- such that 2 + 2 will return 4 if they are integers, but "2" + "2" will return "22" for most manipulations as strings.
- **Floating-Point Numbers.** A computational necessity to fold what would otherwise be large, unmanageable chains of integers (think of, for example, the number of atoms in the universe, represented numerically) into a pair of significant digits and an exponential factor that carries those significant factors forward to large numerical spaces.
- **Boolean.** Booleans are meant to represent the binary structure of logic, with one value asserting something as True, while another asserting something as False. Almost everything in existence can be classified with on or off, yes or no, present or absent, and other basic forms of booleans.

Syntax

Programming syntax varies from language-to-language. It is largely the way programming languages choose to express their commands or logic. A critical thing to note here aside from the vast differences between different programming languages and their respective

syntaxes is the exactitude programming syntax often demands and how different that might be from human language.

In human language, there are grammars and rules that dictate how you form words together both structurally and tactically. A sentence like “Mark is over there.” follows a logical chain with meanings ascribed to each word. However, there is often enough nuance, and even if you disregard some grammar rules (“Mark are over there”) or spell things wrong, your human reader might catch your error but may still grasp the meaning entirely. In programming languages, that will nearly always not be the case. You will have to structure your functions and queries according to exact specifications and make sure the machine gets exact instructions. It is this change in the level of detail required, above all else, that requires testing and debugging to make sure that your program is doing exactly what you need it to do.

Testing/Debugging

Testing and debugging is one of the most important skills you can learn as a programmer.

Here, you should learn how to do ***unit testing***. Unit testing will verify that once you break down your program into individual components, all the way down to the function level, that things work as you designed them to. Most programming languages will have unit testing frameworks -- as an example, here’s is Python’s library. In practice, most unit tests involve testing assertions of logic against the actual performance of a function: you might have designed a function to, for example, create a summation. In order to unit test it, you might compare known outputs of proper assertions (ex: we know an input of 1 and 1 should give 2) and compare it to the actual output (if we find out that the function returns something other than 2, we might have an issue!).

Another thing to consider is the ***divide-and-conquer*** approach. This is where you try to isolate exactly what is causing a bug by eliminating one cause after another. By eliminating conflating factors, once you get to the root of the problem, you can eliminate it altogether.

Functions

Programming functions encapsulate a set of logic in one modular and reusable block. You can think of a function as a way to group together a set of logical substeps to perform something you desire: as an example, if you wanted to calculate the average of a particular dataset, a function could aggregate the dataset and then perform the separate calculations (summation and division) required to generate an average. A function takes an input, runs it through a calculation, and then returns an output.

Objects

An object is a variable that is named in programming and can, therefore, be referenced. One thing to note is that in object-oriented programming, objects have a very specific connotation in that context.

Objects are interactable groupings of both data (often stored as fields within an object) as well as blocks of code (often stored as methods). When they are inheritable, it is largely through the use of classes: which means when you create an object that is a subclass of another, they can inherit the data and programming blocks that the original class of objects had.

There is also prototype-based languages, where objects are created and then linked together rather than instantiated as an overarching class and then having their traits be inherited.

EXAMPLE

Objects are very important concepts in object-oriented programming languages such as Java and Python, though those tend to be multi-paradigm languages, allowing you the ability to not use objects if you don't need them.

Iterators

Iterators allow you to go over elements within a closed organization of data such as a list. It allows you to reference elements of data within an organizer without necessarily amending them.

There are internal iterators that are functions that apply a certain bit of logic to each element in a container. As an example, you might have a function that adds a certain sum to each element: an internal iterator would apply the logic (say + 1) to every element within a list.

There are many built-in functions that deal with iterators. In Python, for example, the "iter" function will take an iterable object and return an iterator that will go through every element. You can go through every element one-by-one until there are no more within a set.

```
In [10]: x = iter([1,3,5])
         x
         next(x)

Out[10]: 1

In [11]: next(x)

Out[11]: 3

In [12]: next(x)

Out[12]: 5

In [13]: next(x)
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-13-5e4e57af3a97> in <module>()
----> 1 next(x)

StopIteration:
```

Note how when you try to iterate away from the defined number of elements within a set, there is an error that pops up.

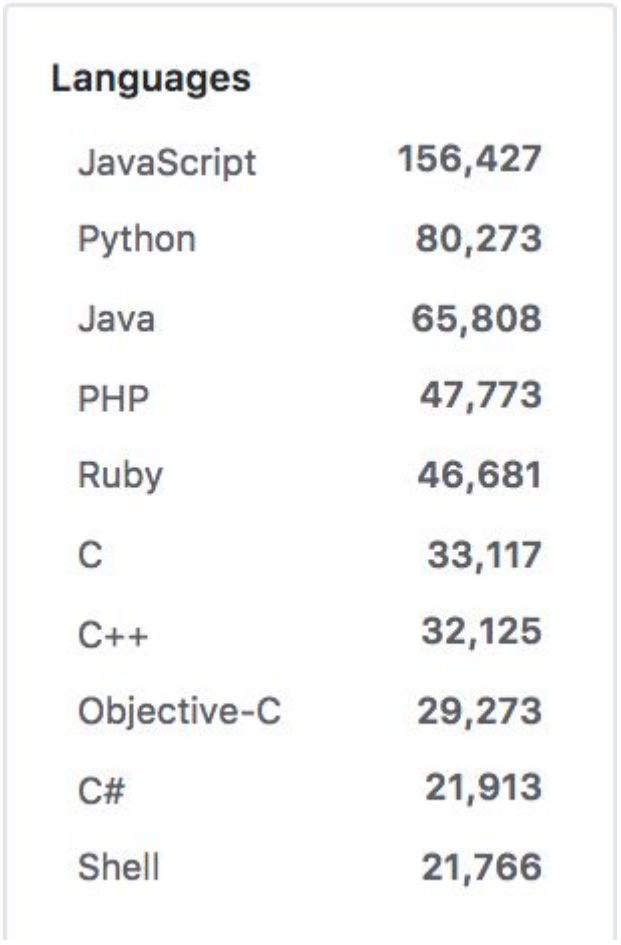
A generator is a subset of the iterator set: every generator is an iterator, but not all iterators are generators. Generators are functions with a yield statement.

Different programming languages

Now that you've learned general programming principles and some of the building blocks of programming, let's get you started on the different programming languages out there.

First, let's do an overview of the different and most popular programming languages out there. It's somewhat difficult to determine an ultimate measure of how popular a programming language is, but we can combine a number of factors.

Github repositories



Github is a living repository of code examples and programming tools that can make your life easier. It's a particularly interesting way to see which programming language is the most popular because it's a living repository: programmers contribute to projects on the platform on a daily and sometimes hourly basis.

By going through Github search and looking for repositories that have been starred more than five times (i.e seen and liked by other programmers), you can get a pretty good rough cut of what the most used and popular programming languages are.

JavaScript dominates the modern web, but you can also see a lot of projects built in Python, Java, PHP, and Ruby. Bringing up the rear are high-performance low-level languages in the C family, with Objective-C focused on iOS mobile applications.

Programming Popularity Index (based on Google)

Worldwide, Jun 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.7 %	-1.3 %
2		Python	15.7 %	+3.5 %
3		PHP	9.3 %	-1.1 %
4		C#	8.3 %	-0.5 %
5		Javascript	7.9 %	+0.5 %
6		C++	6.9 %	-0.2 %
7		C	6.7 %	-0.1 %
8		Objective-C	3.8 %	-0.9 %
9		R	3.6 %	+0.4 %
10		Swift	2.8 %	-0.1 %

This innovative index is based on the amount of Google searches for tutorials in a particular language, which can be a good proxy for the demand for the underlying programming language.

This allows the index to detect trends very rapidly as searches are easier to quantify and aggregate than say, jobs in the field, or the number of programming repositories within each language.

The big difference here is the shift from JavaScript moving down to Java moving up. This may have to do with the fact that Java is often a default language taught to introduce computer science concepts in universities, while JavaScript is a darling of the open source world and innovative open source applications.

Indeed Job Search

If you’re looking for the popularity of programming languages, perhaps no variable might be as compelling to you as the number of jobs available in each language.

The easiest way to look this up would be to take Indeed, a jobs aggregator and return the number of active listings under each programming language. This makes for a solid quantity of jobs -- though we’ll have to dig deeper for the quality of jobs. Thankfully, Indeed has expected salary ranges, and other tools have data that can fill in for the rest.

Example

javascript jobs

Recommended Jobs - 134 new

My recent searches

JavaScript - 1,170 new

data science - 3,668 new

Data Scientist Google - 15 new

Data Scientist - 607 new

» clear searches

Sort by: **relevance** - date

Salary Estimate

$70,000+ (33854)
$85,000+ (27640)
$95,000+ (20885)
$105,000+ (13471)
$115,000+ (7327)

Job Type

Full-time (35428)
Contract (4744)
Part-time (1605)
Internship (561)
Temporary (548)
Commission (45)

Starting with JavaScript, which has about 41,000 jobs in the field. As you can see, the majority of jobs are full-time, and only a few skew to the extreme upper end of the income distribution (with about 17% of all JavaScript jobs registering above $115,000 in compensation).

Sample jobs run the gamut, across different industries and different domains. You could work as a data visualization engineer at a company like Airbnb using frameworks like D3.js, or you could work as a front-end engineer using frameworks such as Angular.js. Combing through different job postings in JavaScript really gives you a good idea of the versatility of the language, and its use across different domains and different industries.

You can use this approach across any number of different programming languages. We'll do that for the most popular programming languages: now that we've helped you aggregate insights from these different programming languages, we'll individually break down each one so you can get more detail.

Python

Description: Python is a general-purpose language that uses the power of its open-source community to extend itself to a variety of uses. From frameworks that can help you build websites (Django) to some of the most exciting artificial intelligence libraries out there (sci-kit-learn)-- Python has a little bit of something for everybody.

Salary: The average salary for those with Python skills varies. Payscale notes that the median salary for those with Python skills as software engineers hovers around $86,000 USD, while those who are data analysts make closer to $68,000 USD. Given how versatile Python is, you can use it for many different purposes: however, the roles that tend to pay more include software engineers, and data scientists. Glassdoor notes that for the general role of Python Developer (which seems to be an aggregation of all these different roles), the average salary hovers around $92,000 USD, with a minimum of $60,000 USD recorded, and a maximum of $137,000 USD.

Uses: Python is one of the most versatile programming languages out there. With the importation of certain libraries, you can use it to either manage static content (Django) or to slice and dice through large datasets (PySpark). Many powerful libraries use Python as a high-level API language to access the functions within. You can, for example, work with big data using the Python interface for either Hadoop or Spark -- or you can work on mathematical computations with your data by using the in-built Numpy library.

Differentiation: What makes Python different from other programming languages starts with its large and expansive open-source community. Because of the way Python works with different modules of code you can import, Python lends itself to large community efforts. Many of the core libraries in Python, including breakaway libraries such as sci-kit learn, are intuitive solutions to several challenging problems, bolstered by several layers of community feedback.

Python's large community is empowered by the very simple syntax of the language, which is intuitive even for non-programmers to pick up. Given the large community that surrounds Python and a large number of technical resources that can help you get immersed in the language, Python stands out from other programming languages on this dimension.

Most Popular Github Projects:

1- Awesome-Python

The Awesome-Python repository, just like all of the other awesome repositories, is a collection of some of the best resources for a particular technical topic. This repository will help you learn as much as you can about Python.

2- httpie

A modern-day command line interface built to help you access http servers across the web in an intuitive fashion. This library has a Python version.

3- thefuck

We all screw up once in a while. This library, built with Python, will automatically and magically correct your latest console commands.

4- Flask

Flask is a mini-framework built to create simple microsites using Python commands. With a few lines of code, you can build small websites that do simple tasks (such as moving a bit of data back and forth between a server and a user). Flask can be a great micro-framework to experiment with as you try out different ideas you have on the web, as it's very rapid and lightweight to put something together with.

5- Django

Django is the go-to web development framework in Python, with enough heft and versatility to manage very large sites with millions of visitors. Django allows non-technical users to edit certain parts of the website if you configure it right -- think of it as a much more versatile Wordpress that trades simplicity for customizability and versatility.

Example Sites: Many websites run or have run on the Django platform, from Pinterest to Instagram. Django has its genesis as a platform for journalists to manage complex content sites -- it continues those roots by supporting the Washington Post, the Guardian, and National Geographic.

Frameworks: There are numerous frameworks in Python -- mostly due to the way that the language imports different modules of code known as libraries. Frameworks such as Django

help you build editable websites. Scikit-learn helps you do different machine learning tasks with ease, allowing you to apply different algorithms to your data. Pandas helps you process and manipulate data -- it, along with NumPy, a library that helps with efficient mathematical processing is often the leading way for data scientists to import, transform and look at their data. Frameworks such as Seaborn and Matplotlib help with data visualization -- the charts and tables you need to display your data in an easily accessible fashion.

Learning Path: The first step to learning Python really involves choosing exactly what purpose you're interested in exploring with it. The learning path for somebody who is looking to do data science with Python is dramatically different from somebody who is looking to do web development -- while the underlying language and syntax may be similar, the libraries and frameworks you would use would be drastically different.

Once you decide what path you want to go on, you'll want to learn the foundations of Python to get started. Python's syntax is very simple and intuitive: it often, by design, reads very similarly to pseudocode you would use to describe new software concepts. Once you get a handle on the built-in commands within Python and how to manipulate and transform different data types, you'll be well on your way. This official documentation from the main Python website on built-in functions will be very helpful in this regard.

Resources: Some of the best Python-related resources involve a curation of different resources. The Python programming language is a large open-source community with plenty of contributors: a large corpus of different resources has been built as a result of almost every nuance of the programming language.

You can start by using a tool such as Awesome Python to find different Python libraries that are best suited for the problem you're trying to solve. The official Python site also hosts a ton of documentation that can make your work in Python easier. This Github repository (also named awesome-python) contains a whole bunch of resources in the space that can be helpful.

JavaScript

Description: JavaScript is one of the most popular languages out there for all sorts of purposes, though it has made its mark as the backbone of the modern Internet. The power of JavaScript mostly rests on the many frameworks that help power different full-stack applications that handle everything from the front-end your users see to the way data is stored in your back-end. The famous MEAN stack (Mongo for data storage, Express.js for routing, Angular.js for front-end manipulation and Node.js for back-end processing), is the new version of the traditional LAMP stack for powering web applications. Newer frameworks like Meteor.js and React.js promise to further solidify JavaScript's capabilities and uses.

Salary: JavaScript salaries vary, with Payscale saying that JavaScript software engineers earn a median of about $80,000 USD, while JavaScript web developers earn closer to $58,000 USD as a median. Senior software engineers who use JavaScript can earn up to a median of $110,000 USD. JavaScript developers earn an average of $72,500 according to

Glassdoor. As you can probably see, it seems to be well worth your while, if you're going to specialize in JavaScript, to be focused more on software development with JavaScript rather than just doing web development.

Uses: JavaScript has a multitude of uses, from its inclusion in front-end frameworks that help render the web the way it is to full-stack frameworks that can handle both the front-end and back-end of any website. It is one of the most versatile programming languages you can find today.

Differentiation: JavaScript differentiates itself by being the de facto go-to language when it comes to web development at scale. Most interactive parts of the web are powered by snippets of JavaScript code that can be called within HTML files. JavaScript powers your login process, cookies that track you, and a bunch of interactive elements within a website, from pop ups to email signup forms.

JavaScript also differentiates itself from other programming languages largely because of the community and the number of different applications and frameworks built on top of it. Because of the breadth of use cases, JavaScript has been used for, several large companies have put their backing behind JavaScript frameworks (for example, Google with Angular.js, and Facebook with React.js). You'll see plenty of JavaScript tutorials as a result and plenty of courses and boot camps are taught in JavaScript.

Most Popular Github Projects:

1- freeCodeCamp

freeCodeCamp is an innovative project that offers to teach you how to build full-stack JavaScript applications with a free, open-source curriculum. By building applications for non-for-profits, you build your JavaScript skills, create social impact, and demonstrate your skills by building usable apps for non-for-profits.

2- Bootstrap

The popular front-end framework for building websites, first released to the public by Twitter, continues to evolve -- use it as a quick way to get something appealing yet powerfully customizable up as a website, and ready to go for your first few users.

3- React.js

React.js is a JavaScript framework built by Facebook that helps effectively build flexible user interfaces across all sorts of different screen sizes. It can be deployed with Node, or the React Native extension allows you to build rich native mobile apps with JavaScript, and JavaScript alone. It is both declarative and component-based.

4- You don't know JS

This series of free books on JavaScript dive deep into different elements of the language from the new ES6 protocol, to the types and grammar associated with JavaScript.

5- Vue.js

Vue.js is a JavaScript framework that's used to build fully interactive web applications. It's built to be an approachable and intuitive framework to those who already know HTML/CSS and JavaScript. Here is a version of Hacker News rendered with Vue.js.

Example sites: Many websites use JavaScript to create and animate interactive elements within their pages, ranging from the IBM Design site to the mini-site called Here is Today. You can see as you browse through these sites that different elements will move and be changed through the flow of time.

There are also a lot of sites built on the different JavaScript frameworks enumerated below. Among the most notable framework, React.js has been used by Facebook to build out Instagram, and it powers the New York Times, Netflix, and Yahoo Mail.

Frameworks: A lot of modern web development frameworks are built on JavaScript. The most prominent is perhaps the aforementioned React.js, however, there are many different frameworks that rely on JavaScript. One of the most well-known is Angular.js.

Learning Path: It is useful to start learning the basics of JavaScript before diving into different frameworks. W3Schools breaks down the different elements within JavaScript, carefully walking you through the basics. Codecademy can help you learn JavaScript interactively. Finally, Learn Javascript the Hard Way will help you dig deeper into fundamental JavaScript concepts if that's what you'd like.

Once you're doing learning JavaScript concepts, you'll be ready to tackle different frameworks that will help you tackle web development tasks. You'll want to learn frameworks such as Node.js, React.js, Angular.js and more that will allow you to build fully interactive web platforms that store and transform user data: think of any number of apps that require you to log in and which assign increasing amounts of data to different personal profiles.

If you want to go from JavaScript basics to getting a fully-functional website up as soon as possible, learning Meteor through the use of tutorials would be a natural second step. Meteor is a full-stack JavaScript platform that simplifies many different web development tasks such as authentication flows. Within a few hours, you can get full-service web platforms set up.

Other frameworks are then worth diving into as well. Most people will use Angular 2, and React.js to build front-end views and use Node.js for the back-end. This tutorial will help you build a Node.js app with Angular.js. This tutorial will get you started on the path of working with Node.js and React.js.

There are also other frameworks worth exploring -- for example, Vue.js, which for some has been easier to pick up and code, especially with recent updates to Angular.js.

This tutorial will allow you to build an interactive quiz game with Vue.js.

Once you're done working with different tutorials, it's time to put your learnings into practice and build different JavaScript applications, releasing them live on the Web for users to experience and to play with.

Resources: There are tons of free resources for those who want to learn JavaScript. This handy listable can help you get started. Here are a list of blogs and different mailing lists in the JavaScript space so you can stay current in the space. If you're more interested in books, here is a list of free JavaScript ebooks that can help you learn JavaScript concepts in a comprehensive manner. The website is categorized so that you can browse JavaScript books for beginners, intermediate developers, and finally, experts.

Java

Description: This object-oriented language was first developed by Sun Microsystems (later acquired by Oracle) about 21 years ago. It's a programming language that is very popular in academia, with many professors and faculties using it as a default instruction language for computer science and engineering majors. Java's syntax is largely borrowed from the C family of programming languages, though it draws its' object-oriented power from a similar framework to that used by Objective-C. Java is also often used to power native mobile apps on Android.

Salary: Payscale has a median salary of about $73,000 USD for Java developers while Glassdoor records an average of around $93,000 USD, making it one of the highest paid programming languages in the United States.

Uses: Java is one of the most-used object-oriented programming languages. It is often used to teach programming concepts in undergraduate level software engineering degrees and courses. It is a higher-level language that has a lot of power and functionality, but is easier to code in than the C and C++ family of programming languages.

Java is used to render interactive applications on the web, desktop GUI apps, and more.

Differentiation: Java is a higher-order object-oriented language that has high security and a rich API. Most people compare and contrast it with the C++ programming family it is derived from. To that effect, it offers more functionality and control of memory allocation than even higher-level languages, but it is more intuitive to use then lower-level languages such as the C family.

It also has a large open-source community, with Github hosting Java repositories en masse -- in fact, at around 300,000 Java-based repositories, Java is one of Github's top programming language by repository count.

Most Popular Github Projects:

1- RXJava

This is a Java library based on providing asynchronous and event-based programs for the Java Virtual Machine.

2- Java Design Patterns

This repository takes a list of design patterns and commonly accepted software principles and applies them in Java.

3- ElasticSearch

ElasticSearch is an open-source and distributed search engine that can be used in API calls to help serve as a search function for different projects.

4- **RetroFit**

This is an HTTP client designed for Java and Android that is supported by the payment processor company Square.

5- OKHttp

Another HTTP client designed by Square for both Android and Java applications.

Example Sites: A lot of the largest websites in the world use Java to a certain extent in their back-ends -- Amazon, Google, and Youtube, for example, all use Java as part of their back-end stack. Initially, serving websites through Java servlets was one of the first ways to accomplish web development at scale.

Frameworks: Java has a bunch of frameworks for a variety of use cases. Hadoop and Kafka, for example, as both big data processing frameworks, are foundationally based on Java. Many of the Apache Foundation's frameworks are based on Java -- important pieces of open-source code that help power a lot of modern software development. Companies like Google also have frameworks for handling front-end JavaScript queries in Java.

Learning Path: Java is often taught in undergraduate and formal university settings as a way to teach object-oriented concepts. You can use something like Codecademy to get started with learning Java with mini-interactive exercises. This Quora thread then has more context on how to extend your knowledge with a variety of different categories of resources. Then, it's suggested that you download the Java SDK and get started building different applications. Then start getting familiar with an IDE (an integrated development environment, a set of tools that will allow you to more easily develop in a particular programming language) for Java such as Eclipse.

Resources: This listable from Simplilearn has more than 40+ resources to learn Java. The learn java subreddit helps community members share different resources and comment on different topics when it comes to learning Java. If you wanted to get the latest resources and news by email, this site offers more than a hundred mailing lists you can join for the latest Java resources.

Ruby

Description: Sometimes considered a toy programming language, Ruby is a simple, clean, principally object-oriented language. Its toy moniker is quite unfair: Ruby is used dynamically and powerfully across the web, especially in conjunction with the Ruby on Rails framework. It's also a multi-paradigm language, able to support functional and imperative programming as well.

Salary: Glassdoor and Payscale both concede that Ruby developers earn on average about $75,000 USD a year, a lower salary than many programming languages. This is perhaps due to the strong use of Ruby for web development, a field of work that traditionally pays less than applications of programming such as software development or data science.

Uses: Ruby was designed to be a simplistic, human-readable programming language that abstracted most of the complexities associated with programming tasks. It is as general-purpose as a language like Python, however, it is mostly associated with web development tasks due to the Ruby on Rails framework.

Differentiation: Ruby differentiates itself through the simplicity of its syntax and its optimization for developer accessibility and happiness. Ruby also has frequent updates and has established a mature ecosystem of developers.

Most Popular Github Projects:

1- Ruby

The Ruby programming language is itself hosted on Github, allowing the community to collaborate and iterate on the programming foundation behind their applications.

2- Slim

Slim is a template language that streamlines programming syntax for HTML and XML to its most simplistic. It allows you to build out well-maintained and well-written HTML and XML pages with a minimum amount of effort.

3- Chronic

Chronic is a time parser that takes natural language queries in time and transfers them over to datetime stamps. For example, the command `Time.now` would return the date and time that was currently there.

4- Treat

Treat is a natural language processing framework but built in Ruby. It allows you to extract text and automatically process it using Ruby.

5- truffleruby

Truffleruby is a high-performance implementation of Ruby. Even though it's missing key drivers that make it impractical to use for web applications, it's a great theoretical framework that is continually improving.

Example Sites: Tons of websites are built on Ruby, especially with the web framework Ruby on Rails. Beautiful photography-based social media 500px is built on Ruby. Popular social media network Ask.fm and project management tool Basecamp are on it as well -- in fact, Basecamp, originally called 37Signals, once developed Rails as an internal tool before releasing it to the public as open source code.

Frameworks: The most popular framework for Ruby is Ruby on Rails -- this allows programmers to set up websites that collect data, and it allows analysts to interact and analyze the data within with ease. With functions that help you set up authentication, localization and more with ease, Rails can help you build dynamic content on the web with very little time.

Learning Path: Ruby in Twenty Minutes helps you get started playing around with the Ruby language and getting accustomed to the syntax. You'll be able to work within your computer's terminal and you'll start getting used to Ruby by putting in simple Ruby commands. You can then use something like Codecademy to extend your learning by practicing Ruby with more real-life scenarios and extending your simple syntax learnings to slightly more complex programming problems. This listable then contains 50 resources on Ruby you can check out and play with.

Once you've gotten Ruby basics down pat, you'll want to learn Ruby on Rails if you want to extend your learning to web development. You can start learning the framework with Codecademy though you'll want to refine your learning with this book and set of interactive exercises: The Ruby on Rails Tutorial.

Resources: This list of 14 important Ruby resources will help you along your learning journey. This Medium post helps structure different Ruby concepts and learning resources together such that you'll be able to seamlessly classify where you are and move towards resources that are at your Ruby skill level.

SQL

Description: SQL is a domain-specific language used for extracting and organizing data in relational databases. Relational databases hold columns of data that are traits of different rows. SQL is a specific declarative programming language: queries use names such as SELECT or LIMIT that encapsulate a set of logic -- you don't have to specify each logical step behind what you're trying to do in order to use SQL: you can merely declare functions that align with what you're seeking to perform (ex: select a portion of the data from a table with the query SELECT) instead of laying out each step.

SQL is a mainstay for most people working with data, as many databases are managed relationally.

Salary: The average pay for most SQL-skilled programmers is about $70,000, though there's a split between data analysts who use SQL (who tend to earn less) and software engineers and programmers that use SQL (who tend to earn more).

Uses: SQL is used to manage databases. Data analysts will use SQL to query tables of data and derive insights from it. Data scientists will use SQL to load data into their models. Data engineers and database administrators will use SQL to ensure that everybody in their company has easy and intuitive access to the data they need.

Differentiation: SQL is written to be a query language -- it's meant to be an intuitive way to access data. What it sacrifices in terms of pure functionality, it gains in terms of making data queries accessible. This is why many projects still use SQL and its associated frameworks as their data storage and querying solution.

Most Popular Github Projects: There is no specific section for SQL on Github, though there is a category devoted to PL/SQL from Oracle.

Example Sites: Most websites will use SQL as a back-end data storage and data processing solution. Examples of this include Facebook and Yahoo that use MySQL as a backend.

Frameworks: There are different versions and frameworks for SQL, with the most prominent being MySQL, an open-source solution that helps facilitate SQL's role in managing back-end data for web applications.

Learning Path: In order to learn SQL, you'll want to first start with SQLZoo, an interactive interface that will allow you to practice different queries all the way from basic SELECT functionality to advanced subqueries. Mode Analytics then offers a way for you to exercise your SQL skills in order to solve different business case situations. By going through both resources, you'll learn the technical underpinnings of SQL methods, then you'll start understanding the mentality of the data analyst who uses SQL to extract and process data.

Resources: This listable will provide you with 18+ resources to help accelerate your SQL learning. Should you ever need a refresh on SQL concepts and some practice with queries, you can always use W3School's section on SQL.

PHP

Description: PHP, developed in 1994 by Rasmus Lerdorf, is primarily used to define server-side logic (the back-end) of online websites. It commonly forms a part of the LAMP stack that powered most of the Internet (LAMP standing for Linux as the operating system, Apache as the web server, MySQL as the database solution, and PHP as the programming language meant to process server-side logic). While PHP has been eclipsed to a certain intent by new web frameworks (notably JavaScript and Ruby ones), it still retains its appeal among some -- PHP still powers notable frameworks such as Wordpress. PHP does however have some notable detractors.

Salary: The salary for a PHP developer tends to average around $65,000 according to Payscale, and the national average according to Glassdoor is around $85,000. With most of the PHP jobs clustering around web development, it should be no surprise to see slightly less pay for PHP workers than for programmers who specialize in other, more versatile programming languages.

Uses: PHP was oftentimes used as the default back-end programming language for most web applications in the 90s. Nowadays, JavaScript has dominated this niche, however, PHP is still a viable back-end web application technology: as an example, sites run on the Wordpress framework will use PHP for back-end processing.

Differentiation: PHP has a lot of users but also quite a few detractors. Most people seem to settle on the fact that since Apache and mod_php features are so popular, most people will use PHP as a starter language of sorts, and as a server-side extension of HTML templating. However, with the development of Ruby on Rails, server-side JavaScript frameworks and now Golang and more, the utility and differentiation of PHP has diminished, especially for people who argue that PHP's syntax is not so developer-friendly.

However, the context of PHP has to be considered: at the time of its heyday, most web applications would have to use burdensome Java servlets to render web content. At the time, paired with its ease of use when it came to open source database solution MySQL and web server Apache made it the key web development framework to embrace. A lot of the pre-2005 web development ecosystem evolved around PHP, making such ecosystems as Wordpress largely built on PHP. The differentiation that comes from that mostly stems from PHP being a proven solution with a viable community and plenty of introductory tutorials and frameworks built around it.

Most Popular Github Projects:

1- Laravel

Laravel bills itself as a framework for web artisans. It promises to deliver web applications easily with seamless, expressive, and beautiful syntax.

2- Awesome-PHP

Just like the rest of the Awesome repositories, this repository is curated a selection of the best PHP resources available on the web.

3- Symfony

Symfony is a framework of reusable PHP components that can help make web development in PHP easier. It helps you deal with common issues such as routing and authentication and makes them seamless to roll out with preset configurations.

4- CodeIgniter

CodeIgniter is another lightweight PHP framework for web development, built in the same vein as Symfony and Laravel.

5-HHVM

HHVM is a virtual machine that can execute in Hack and PHP that was developed and supported by Facebook.

Example Sites: Facebook and Youtube were sites that were originally built on PHP. Sites built before 2005 will largely be built on PHP.

Frameworks: A lot of PHP frameworks focus on making web development tasks easier, from the aforementioned Laravel, Symfony, and CodeIgniter. This is in line with PHP's general rise as a web development focused programming language. A more comprehensive look at PHP frameworks and a comparison between their different features is placed in this article.

Learning Path: To learn PHP, you can use Codecademy to get started. W3Schools will help you with different PHP components and functions if you need to individually refresh yourself on PHP practice. Then, you'll want to be able to practice with something like XAMPP and build simple web applications to get you going and learning. PHP is something best learned with practice and by building different things you can iterate on. As PHP is mostly used for web development, you can quickly iterate on different versions of a website and see how PHP renders the logic you want to create.

Resources: A list of common PHP tutorials and resources are available here. The PHP Resource Index here contains almost 4000+ PHP resources for a variety of different use cases. PHP Weekly is a newsletter that will help you keep in touch with cutting-edge resources in the PHP space by sending you a weekly email digest.

R:

Description: R is an open-source programming language most often used by statisticians, data scientists, and academics who will use it to explore large data sets and distill insights from it. It offers multiple libraries that are useful for data processing tasks. Developed by John Chambers and other colleagues at Bell Laboratories, it is a refined version of its precursor language S.

It has strong libraries for data visualization, time series analysis and a variety of statistical analysis tasks.

It's free software that runs on a variety of operating systems, from UNIX to Windows to OSX. It runs on the open source license of the GNU general public license and is thus free to use and adapt.

Salary: Median salaries for R users tend to vary, with the main split being the difference between data analysts who use R to query existing data pipelines and data scientists who build those data pipelines and train different models programmatically on top of larger data sets. The difference can be stark, with around a $90,000 median salary for data scientists who use R, vs about a $60,000 median salary for data analysts who use R.

Uses: R is often used to analyze datasets, especially in an academic context. Most frameworks that have evolved around R focus on different methods of data processing. The ggplot family of libraries has been widely recognized as some of the top programming modules for data visualization.

Differentiation: R is often compared to Python when it comes to data analysis and data science tasks. Its strong data visualization and manipulation libraries along with its data analysis-focused community help make it a strong contender for any data transformation, analysis, and visualization tasks.

Most Popular Github Projects:

1- Mal

Mal is a Clojure inspired lisp interpreter which can be implemented in the R programming language. With 4,500 stars, Mal requires one of the lowest amount of stars to qualify for the top repository of a programming language. It speaks to the fact that most of the open-source work done on the R programming language resides outside of Github.

2- Prophet

Prophet is a library that is able to do rich time series analysis by adjusting forecasts to account for seasonal and non-linear trends. It was created by Facebook and forms a part of the strong corpus of data analysis frameworks and libraries that exist for the R programming language.

3- ggplot2

Ggplot2 is a data visualization library for R that is based on the Grammar of Graphics. It is a library often used by data analysts and data scientists to display their results in charts, heatmaps, and more.

4- H2o-3

H2o-3 is the open source machine learning library for the R programming language, similar to scikit-learn for Python. It allows people using the R programming language to run deep learning and other machine learning techniques on their data, an essential utility in an era where data is not nearly as useful without machine learning techniques.

5- Shiny

Shiny is an easy web application framework for R that allows you to build interactive websites in a few lines of code without any JavaScript. It uses an intuitive UI (user interface) component based on Bootstrap. Shiny can take all of the guesswork out of building something for the web with the R programming language.

Example Sites: There are not many websites built with R, which is used more for data analysis tasks and projects that are internal to one computer. However, you can build things with R Markdown and build different webpages. You might also use a web development framework such as Shiny if you wanted to create simple interactive web applications around your data.

Frameworks: The awesome repository comes up again with a great list of different R packages and frameworks you can use. A few that are worth mentioning are packages such as dplyr that help you assemble data in an intuitive tabular fashion, ggplot2 to help with data visualization and plotly to help with interactive web displays of R analysis. R libraries and frameworks are some of the most robust for doing ad hoc data analysis and displaying the results in a variety of formats.

Learning Path: This article helps frame the resources you need to learn R, and how you should learn it, starting from syntax and going to specific packages. It makes for a great introduction to the field, even if you're an absolute beginner. If you want to apply R to data science projects and different data analysis tasks, Datacamp will help you learn the skills and mentality you need to do just that -- you'll learn everything from machine learning practices with R to how to do proper data visualization of the results.

Resources: R-bloggers is a large community of R practitioners and writers who aim to share knowledge about R with each other. This list of 60+ resources on R can be used in case you ever get lost trying to learn R.

Go (Golang)

Description: Go is an open-source programming language developed at Google in 2007. Often referred to as Golang, it bears a lot of similarities to the C class of programming languages. It has been developed to be more readable and concise than C, however.

Salary: Golang has quite a high pay range, Golang software engineers earning around $95,000 USD according to Indeed. In fact, senior software engineers can expect to earn even more with Golang, with an average salary of $115,000 USD.

Uses: Golang is mostly used for multi-threaded applications, a fact of life that is becoming more and more relevant as more multi-core processor hardware is rolled out in computers due to the recent limitations of Moore's Law increases. Because Golang is compiled and not interpreted just like the C++ family, performance is increased. For this reason, Go is considered the server language of the future.

Differentiation: Golang's capability with concurrency on multiple threads, its direct compilation on processor hardware and it is super easy to maintain, as inheritance is minimized. This entails an extra cost in terms of writing out different explicit snippets of code, however in the long run, it keeps Golang code clean and easy-to-read for any programmer. Plus, it doesn't hurt that Golang is backed by Google.

Most Popular Github Projects:

1- Awesome-Go

Part of the Awesome family of repositories, this repository collects and organizes all of the different frameworks and packages you would need to accomplish different tasks in Golang.

2- Lime

Lime is an open-sourced version of Sublime Text, the intuitive text editor that most programmers use to edit their code. It is primarily built in Golang, and it's meant to offer an open-source alternative to Sublime Text where the code is available to the public.

3- Traefik

Traefik allows you to easily manage different microservices you deploy by managing configurations automatically and dynamically. You can host your microservices on programs like Amazon Web Service ECS or Docker and fire and forget it: Traefik will take care of everything for you when it comes to load balancing and routing.

4- TIDB

Tidb is a distributed datastore for transactional analysis that allows you to use MySQL to probe the data.

5- kit

Go Kit is a common tool and architecture for building microservices in Golang.

Example Sites: A lot of different companies use Golang as the back-end for their web applications. This is a comprehensive list -- among the companies and sites to highlight are Atlassian (the creators of Bitbucket and Jira), popular storytelling and self-publishing application Wattpad, and Google will often use Golang for a variety of applications.

Frameworks: There are many different frameworks used in Golang, as you can see in the Awesome-Go list. Each one can be used for a different purpose. If you're looking for a Golang web application framework that connects with HTTP requests, you'll want to check out something like Chi. If you want to make sure you stay productive with Golang programming, you can use frameworks such as Revel to help you stay on track with hot code reloading. Something like Buffalo goes beyond just a framework, proposing to create a set of standards and an ecosystem dedicated to helping Go web apps get up as quickly as possible.

Learning Path: In order to get you started with Golang, you'll want to do interactive tutorials starting with the dreaded 'Hello World' exercise. This site helps you put together the code you'd need to execute commonly desired features. Then you can help execute these basic functions in this interactive tutorial. Then as this thread alludes to, you're best off trying to learn by building different web applications with Golang.

Resources: The official Golang page has a set of handy resources, with different mailing lists, chat groups, and forums that you can participate in. This is a brief list of some books that one can use as well to get acquainted with Golang.

C++, C, C#

Description: The C family of programming often revolves around imperative and general programming principles. It uses a straightforward compiler that allows it to work with lower levels of programming (closer to operating memory) and is thus oftentimes more powerful and faster than higher-level languages. As an example of this, Python, which is built partially on C, often runs a hybrid of the language Cython in order to access faster and more performant functions.

Salary: The starting salary of a C-programming family using programmer is about $100,000 USD, one of the highest salaries out there.

Uses: The C++ family of programming languages is used for computing tasks that require more lower-level control and power.

Differentiation: The C++ family is known to be a highly performant programming language but one with a lot of nuances. You'll have to build in memory management and a bet of lower-level programming tasks that are usually taken care of by higher-level programming

languages. You can create apps that are very scalable, fast, and highly performant using C++'s ability to control the allocation of resources on a very low level.

However, C++ is both syntactically expansive and also quite complex to read and understand. It can take a lot of time and money to maintain apps written in C++ at scale. If you're building something that requires a lot of computational resources such as a video game or an animated movie, you might want to consider C++ as a way to build out something that will work for you.

Most Popular Github Projects:

1- Neovim

Neovim is built on an open-source foundation to refactor Vim (the code editor), allowing it to have access to modern GUIs, API access from all sorts of programming languages, and features such as asynchronous job control. It's predominantly written in C.

2- **Git**

Git is the language Github and other repositories use to update and maintain code repositories. This is a publish-only version of the language that doesn't accept any pull requests or anything that would modify the underlying code, but it's written in C and shell script, and it can be a useful exercise to poke around the repository and see how it's built.

3- **The Silver Searcher**

The Silver Searcher is a code searcher written in C that can help you find different snippets of code within your programming files. It is about 30x faster than equivalent code search engines such as ACK.

4- **Ruby**

Ruby is the interpreted scripting language that we covered above -- its usage with Ruby on Rails has made it one of the most popular programming languages out there, especially when it comes to web development tasks. A lot of its components are written in C.

5- **Vim**

Vim is a popular text editor often used to edit different programs. It is based on the UNIX text editor Vi and it is largely built on the C programming language.

Example Sites: You won't often find websites being built with the C family of languages as the programming language has an expansive syntax and is hard to maintain in terms of both time and money when it comes to large-scale apps that need to be rapidly iterated on. However, a ton of applications on the web are built or enabled with the C family of

programming languages, from every site that runs MySQL for data storage and processing needs to the popular web browser Mozilla Firefox.

Frameworks: The awesome repository for the C and C plus family of programming languages contains a list of frameworks you can use. You can use TensorFlow in the C++ programming language if you want to get working on artificial intelligence and deep learning problems. Dlib will help you implement high performance machine learning algorithms and different solutions to complex computing problems. QT contains a variety of user interface building tools. Finally, Boost contains a variety of generic frameworks for the C++ community, solving all kinds of common problems from date-time implementation to common standards for geometry.

In general, you'll see C-family libraries geared towards taking advantage of C's high-performance capability to solve complex problems that require intense computation -- computation other programming languages might not be able to bring to bear. As an example of this, you have frameworks that deal with rich audio such as FMOD, and game engines such as Allegro.

Learning Path: The C family of programming languages isn't very easy to pick up: it's not an intuitive programming language that you should take on as a challenge unless you're really looking for one. This Quora thread offers a whole bunch of suggestions from books to interactive courses. There are often comprehensive resources similar to textbooks offered for free such as this comprehensive PDF guide. You can then use platforms like HackerRank to solve different challenges in the C family and help apply your newly learned skills to real challenges. Then, you should look to building different applications and creating useful software.

Resources: Toptal offers the ultimate list of C and C++ resources with different books and online resources. Cplusplus.com offers tutorials and blogs, as well as a comprehensive forum with other C adherents. Stack Overflow has C programming questions and discussions around specific programming situations.

Swift, Objective-C

Description: Both Swift and Objective-C are programming languages used to power the mobile applications on the iOS ecosystem. If you own an Apple product and you have access to the App Store, you've benefited from apps written in both languages.

Objective-C is the older of the two languages and has been superseded by Swift. However, Swift has been created to be reverse-compatible with most Objective-C apps. Even though it is largely supported by Apple, Swift is an open source programming language, and one of the fastest-growing programming communities.

Salary: iOS developers tend to earn about $80,000 USD a year in median salary. Most iOS job requirements these days will ask for experience in Swift, though there are still some niches for people developing in Objective-C.

Uses: Swift and Objective-C are mostly used to create apps in the iOS ecosystem. Most of the mobile apps on the Apple ecosystem are built on either one of these two programming languages, with newer applications being more likely to be built on Swift. Both languages allow for seamless integration with Apple products, with powerful native integration with parts such as the camera.

Differentiation: While frameworks such as React Native have popped up to deal with the programming end of iOS mobile apps, the Swift and Objective-C family have powered most high-performing Apple apps and continue to do so. If you want to build something in the mobile space, you're likely going to use Swift or Objective-C at some point.

Most Popular Github Projects:

Example Sites: Most of the examples of Swift programming are mobile applications built for people who are in the Apple ecosystem of products. For example, the official Firefox and Wordpress apps for iOS are built on Swift. Designer News and the native app for iOS (iPhone, iPads and other Apple products) are built on Swift. This list on Medium contains a selection of the different applications that are built on Swift.

Frameworks: Swift has a number of different frameworks and components that can supplement existing libraries. The core libraries contain different frameworks that can be used for things like unit tests.

Learning Path: The path to learning Objective-C and Swift is very experiential and involves getting to the level where you can build mobile applications. You'll likely want to learn Swift, as Objective-C is being slowly phased out of the mobile industry, though if you really insist, Tutorialspoint has a section-by-section walkthrough on Objective-C.

Moving onto Swift though, Apple has a lot of different documentation resources for Swift that are worth pursuing. Learnswift.tips then has a comprehensive walkthrough that will take you from beginner to more advanced stages, helping you learn everything from the user interface basics in Swift to how you can use Objective-C to interpret your new Swift functions.

You can split between Objective-C and Swift and go back and forth from Objective-C to Swift by migrating your code.

Resources: Devactic has a list of 10+ resources to help learn Swift. This Quora thread has 100+ answers about the path you could take to learn Swift. In case you wanted to pick up Objective-C, this thread on Stack Overflow will help you locate the very best resources for you so you can pursue learning in that direction.

What you can do with programming skills -- and how to direct your learning

One of the largest problems I see in how programming languages are taught is that there's often not a lot of discussion about what you want to do with code. Oftentimes, learners are

told that learning code is a virtue in of itself, which is simply not true. Yes, you can consider it mental exercise -- though I'd point out that you could get about the same from a crossword puzzle. At the end of the day though, this spirit encourages using technology for technology's sake -- diluting the promise of programming that is impactful and makes people's lives easier.

That's the first thing I want to tackle. I don't want to spend the time telling you which programming language is easiest or hardest, or which one pays the most. I want to tell you what programming languages are good for which purposes and sketch out a plan for you to get started doing something awesome. Then I'll detail each language briefly and how it can help you with your goals. Then I'll get into the real meat of the fundamental programming principles you have to understand to build the right foundation and the steps others took to get a programming job without a degree. Finally, I'll leave you with a set of resources, and some people to contact when you've finished your journey of learning. You should skip ahead to the most relevant sections.

Use technology, not for technology's sake, but to accomplish your goals. Use this book accordingly. We'll start with what you want to accomplish -- and if you're not quite sure what that is, that's fine! At the very least, we'll now enumerate the possible.

You can do a whole lot with programming in general

You can many things you'd like with programming and software. Let's enumerate a few examples here:

1. You could build your own virtual storefront and sell different goods.
2. You could build your own blog and start publishing different articles and essays.
3. You could build an application that automated daily tasks for you such as getting a particular snippet of information from different sources. Perhaps you want to put together financial results from different stock markets with the news of the day. You can do all of that by programming software that grabs and displays that information.
4. You could build artificial neural networks that can beat humans at simple and complex games.
5. You can build mobile apps that capture data points and help display or visualize them, or any number of applications, from communications to game apps.
6. You can solve complex scientific and mathematical problems with the power of computation and automation. Scientists will use computer programs to do everything from drug discovery to finding the secrets of the universe.
7. You can build distributed hardware and payments systems that help transact data and value across different nodes. An example of this could be cryptocurrency, and another could be using the Internet of Things to measure different data points such as temperature within different stores.

As you can see, there are tons of things you can do with programming -- we enumerated a few use cases, but the list was by no means comprehensive. You could get started with any number of use cases, from using it to create cutting-edge artificial intelligence applications to

building your own blog. I've outlined a few of those options and given you key resources to pursue if you want to follow that path. For two of the options (deep learning or artificial intelligence in Python and building dynamic web and mobile resources with cutting-edge JavaScript frameworks), I've helped enumerate what resources you can specifically consult.

You want to build dynamic websites that interact with user data

Many people start their adventure in programming by wondering how dynamic websites are built. After all, surfing the web is often an interactive experience filled with mysteries to the non-technical. Web development seems to be a pillar of the digital economy: with so many new ideas flourishing into life, there needs to be somebody who can help support the sharing of those ideas to thousands, perhaps millions of people.

The able web developer plays that role, creating an interactive database that pushes and gets data from people around the web. This, in many ways, is the foundation of 21st-century technical skills -- there are few people who work in technical fields that don't have at least a passing knowledge of HTML/CSS at the very least. Web development boasts an average salary of $58,000 USD in the United States,though companies such as Microsoft and Amazon pay a lot more (almost double, around $100,000). Its importance, however, doesn't lie in the salary you can make on pursuing the skillset alone, but in how web development can help augment anything you do with technology by amplifying the message and impact you want to send.

What you could do with these capabilities

A data science project can be hosted online, and interacted with by thousands if not more, adding to any data you might have. Understanding how to pass information from websites such as Facebook and Twitter can help you create sophisticated data analyses. The fundamentals of web development apply to mobile development. Chatbots need to pass data back and forth in order to carry on with conversations and so do video games.
With web development skills, you can build nearly anything you want.

How to get started

In order to understand how to do web development, you have to start at the very beginning with HTML/CSS. In that respect, Codecademy has a good introductory tutorial and so does W3schools.

My favorite metaphor for how HTML/CSS work is that the HTML is like a housing structure: it's how you set the rooms of a house, how they're placed with one another, and how the rooms relate with one another. CSS is the painting scheme on top of it. CSS dictates that you want the walls all painted white or a certain wall to be given a certain background. It (usually) doesn't affect where the walls are placed. That's HTML's job.

With the basics of HTML/CSS on lockdown, you'll have learned elements of what's known as front-end web development. You can now control what the user sees when they visit a webpage -- though you still haven't learned how to get the website to interact with the user!

At this point, though your CSS skills might not be so advanced, you can start to appreciate sandboxes such as codepen.io. You might be able to use those snippets to enliven your own websites! You can also start using frameworks such as Bootstrap to deal with a lot of tricky problems starting up -- anything from the graphical elements you want to display (such as buttons) to responsiveness: what your website will look like on different screen sizes and a variety of different devices. You'll be able to quickly copy from different examples and create websites that, within a few minutes, will not look out of place on the first page of Google.

At this point, you can start learning the basics of JavaScript as well. By then using libraries such as jQuery you'll be able to make websites interactive with clicks and certain user actions. Once you're at this stage, a coding sandbox such as JSFiddle will help you practice your skills with JavaScript in the mix.

Once you've mastered the basics of front-end development, you can now look at back-end development, where you are able to interact with a server you control and manipulate the information it shows and collects from different users.

You'll want to get acquainted with a back-end framework that can help you abstract away most of the work. Some popular ones include JavaScript mainstays meteor.js and node.js and the various frameworks that work with Node to control different views, from react.js to angular.js. There is also Ruby on Rails and Python web development platform Django.

With all of this knowledge, you're well on your way towards building something on the web -- and continuing to iterate on it.

Career prospects

Skills in web development can help you create your own company, work in web development jobs or in product management. Web developers earn an average of $66,238 -- that goes up to about $87,965 if we're talking about the Bay Area though, where a large number of startups and tech companies are based.

Eventually, you can use these skills to climb to the ranks of top product or technical officer in a larger company, or be at the helm of a successful company or agency that uses your web development skills.

Summary

Most people think of web development when they think of programming -- it's often the first career track many pursue. With a bit of learning and a lot of practice, you can quickly gain a set of skills in this area: becoming a web developer and a programmer.

You want to play with data, machine learning, and artificial intelligence

There's been a recent craze with artificial intelligence, machine learning, and data science. Experts say that there should be at least one million people who need to be trained in artificial intelligence in the United States.

Data science, artificial intelligence and machine learning interface with one another. Data science is a combination of mathematical, programming and communication skills and traits that help people and teams unleash insights from within the huge amounts of data now generated every day by the Internet or proprietary data sources.

Machine learning is a segment of artificial intelligence approaches, where a computer is trained on datasets and is able to perform at human-like levels or much beyond at cognitive tasks such as classification and pattern matching.

Artificial intelligence is a broad umbrella of different approaches to allow machines to replicate humans on some level. While the Terminator type of artificial intelligence could best be described as "strong" artificial intelligence (a machine that could think and act independently of humans at a level that came close or exceeded human consciousness), the artificial intelligence that is in use and has had practical results is much simpler. Deep learning approaches now dominate the more practical side of artificial intelligence, where computer scientists are trying to accomplish cognitive tasks at human-like levels or above with computer programmed neural networks that are simplified versions of how the brain is theorized to operate, with virtual feedback cycles that are supposed to be a crude simulation of how our brain's neurons work.

The common path between these different interests can help lead you down a variety of interesting forks with very few variations in what you'd have to learn. Gaining the capability to slice through data and do something actionable with it may just become an essential 21st-century skill.

What you could do with these capabilities

While it seems as if artificial intelligence, at times, has become a buzzword that threatens to overwhelm society (nevermind prospective learners), the truth of the matter is that artificial intelligence has many practical engineering purposes. Whenever Amazon or Netflix is able to serve you recommendations tailored to your preferences, you'll have benefitted from data science and artificial intelligence.

Data science doesn't just extend to increasing profits. It can also be a significant part of creating social impact. Data scientists used machine vision to categorize satellite photos of African villages -- and in doing so, were able to isolate which neighborhoods were poorer due to the presence of thatch roofs. This allowed charities such as GiveDirectly the ability to

direct their donations to the neighborhoods where they would have the most impact without spending money sending teams to manually find poorer neighborhoods.

In popular culture, artificial intelligence is vaunted as both an existential threat and a fascinating device with which we examine our own humanity. The reality, as of now, is certainly more mundane and practical. While there is lots of work to do when it comes to the science of artificial intelligence and the definition of consciousness, the engineering component of artificial intelligence has blossomed thanks to the great amount of data humankind is creating each day on virtual networks. You could use that engineering power to achieve astonishing, automated performance on a bunch of difficult tasks that usually require human intervention, from classifying different images to establishing correlations between different patterns.

Yelp used data science and machine learning to classify different images of restaurants and create a quality score for them. Airbnb used bookings data to see if net promoter score (a proxy for how satisfied a customer was with the service and product Airbnb offers) was an accurate predictor for rebooking. You can use data science and machine learning to automate a class of actions that can help you unleash insights in large sets of data -- whether that is financial, health or any other kind of data.

How to get started

In order for you to get proficient with data, you'll most likely have to learn either **R or Python**, the most popular data science programming languages. You'd also be inclined to start playing around with datasets such as free public ones. Springboard has a good list, while Quandl offers a search engine that deals with large datasets, especially economic and financial data.

Data science has a lot of different components to it. You'll also want to get acquainted with different statistical methods and algorithmic approaches to data and the basics of machine learning and deep learning if you want to get into the nitty-gritty of artificial intelligence.

There are a lot of different ways to get started with exploratory data analysis. It could start as easily as playing with datasets in Excel, then proceed to use SQL to extract out data -- Mode Analytics has a great SQL school and sandbox for just this purpose. Once you get the hang of exploratory data analysis in SQL and Excel, you'll want to start learning the statistical methods and algorithms behind data science. Khan Academy has a good primer on inferential and descriptive statistics in case you need a refresher on the subject or in case you haven't learned it at all. For the algorithms behind data science, this guide offers a brief overview of each category of algorithm.

The best way to get acquainted with them all, however, is to think of questions you might have that can be answered with data -- and then proceed to tackle them, using resources like Stack Overflow's Data Science focused exchange to advance.

Thankfully, learning either R or Python can be the conduit to more advanced machine learning work where algorithms can be called rather than made from scratch. Python, in particular, has what is known as the data science stack: NumPy to help with the manipulation of numbers, Pandas to help with the table-like data management of large amounts of data, NLTK to help with any natural language processing (a technique that translates streams of text into something coherent and manipulable for computers to process) and Matplotlib to help with any data visualizations you might want to generate, from histograms to pie charts. You can also use libraries such as Seaborn for fancier visualizations such as box plots that may accurately convey the insights within your data.

Most importantly though, Python boasts the scikit-learn library, which is a black box solution that allows you to implement algorithms on the go. Scikit-learn has everything from the algorithms themselves to modules of code that allow you to evaluate how effective and accurate those algorithms are -- everything is off-the-shelf. You'll be able to do machine learning Ikea-style without losing much if any efficacy from something tailor-made.

Python also allows you access to specialized deep learning tools like Google's Tensorflow and Theano, as well as open source neural network software such as Keras built on top of these frameworks. This allows you to easily use deep learning techniques and neural networks meant to be a learning algorithm that will actually correct and optimize itself towards a goal -- whether that's properly classifying road signs or storefronts, or translating old manuscripts into computerized text. Deep learning is the foundation behind the artificial intelligence craze, and the path to wondrous technological innovations, from self-driving cars to parcel-delivering drones.

You could also tackle big data, terabytes or beyond of data with billions of rows of transactional data with specialized tools such as Hadoop and Spark. The best part? All of these tools are open source, and free forever!

Ideas of different problems you can tackle which will require data science are the classification and labelling of images en masse (for example, you could classify which images in a restaurant are of food, and of what kind), pattern-matching trends that could be extrapolated to the future (predicting, for example, the rise and fall of certain stocks) and the segmentation of different customers into different categories, perhaps based on their likelihood of buying your products. You could perform sentiment analysis of Tweets, create predictive models and do all sorts of data analysis with your new skills and tools.

Career prospects

There are a lot of careers that require knowledge of data, though perhaps none is as celebrated as the role of data scientist, the "sexiest job of the 21st century". With average salaries above $100,000 USD, the field is one where many can make lots of impacts and where anybody who succeeds in the industry can thrive financially.

There are also different data roles you can embrace such as being a data analyst or data engineer where you either query data and create reports for business teams, or you create

entire data pipelines that handle millions of lines of data. The possibilities are endless for somebody with the programming, statistics, and business skills of a data scientist.

Summary

In order to pursue knowledge in machine learning and get a career in data, you'll want to sharpen your statistics skills, your business knowledge, your ability to create data visualizations, your knowledge of different algorithms and approaches to artificial intelligence and come out with the ability to drive insights from the large amount of data generated every single day in the digital age.

You want to build mobile apps and get involved in the Internet of Things with smart hardware

The Internet used to be a static tether where you signed onto a desktop at home, got the information you needed, then left. Now with the explosion of mobile devices and screens that have more power than the mainframes of older models, that reality has starkly changed. The static Internet of the 90s has grown into a dynamic, fully embedded Internet of Things, where intelligence and data animate every layer of the world around us, from billboard signs to smart wearables.

What you could do with these capabilities

It's amazing how far we've come from the computer revolution of the 90s. Nowadays, a mobile phone carries more processing power than ever. In fact, a modern cell phone now has many more times the processing power as the computer that sent Apollo 11 to the moon. It's not just mobiles either. Smart hardware, aided by the creation of Raspberry Pi and Arduino has evolved so that makers can independently build hardware projects such as world clocks and animatronic hands.

All of these layers of hardware have created the largest app marketplaces out there, a few basic examples being the app stores from Apple and Google Play. You don't have to know much or any of the hardware to get started with some basic software development that can vault you into an established career as a mobile developer or at the helm of a freelance career in mobile development.

How to get started

If you want to get started on the software side, the first thing to note is how split the ecosystem is. You'll want to especially note the difference between the Apple and Android ecosystems, the two largest players in the space. While Android has more currency when it comes to engagement, users, and downloads, Apple regulates its ecosystem much more strictly and is able to market successful apps better -- which leads to a balance where Android users download more, but Apple users pay more. Depending on what matters to

you, you might then decide to look into the different options for mobile development for both ecosystems: Java for Android -- and Swift for Apple.

Career prospects

There are many different opportunities in the mobile development field. You could build apps yourself as an entrepreneur, or you could become part of a larger team.

Summary

You want to play with cryptocurrencies like Bitcoin or deal with matters of Internet privacy and security

Ever since Bitcoin has emerged, it has become a global phenomenon: an almost magic-like apparition of technological money. Yet, despite all the hype, there is still something deeper within the phenomenon of bitcoin that remains unexplored, a link to a fascinating field of technology that is slowly but surely taking over essential parts of the Internet.

Cryptography has been an age-old pursuit. Millennia ago, Roman soldiers used the Caesar cipher, a simple system that moved letters a default setting away from their original placing (such that, for example, a cipher of 3 would mean A becoming D, B becoming E and so forth) so that enemies and the general public couldn't read their correspondence.

Bitcoin's relationships to cryptography is often an under-explored part of the ecosystem. Bitcoin solves an age-old problem in cryptography, namely the Byzantine Generals problem: how do you coordinate trust and optimal solutions with a bunch of agents, some of whom may be untrustworthy or looking to undermine the system? Bitcoin solves that with proof of work and the blockchain, a decentralized ledger of legitimate transactions. This allows for a system that transfers significant economic value among different agents without the possibility of a nefarious agent taking over the entire chain and stealing all of that value.

Cryptography is an age-old art that has now emerged as the basis behind essential parts of the web, from financial transactions, to protecting sensitive data. In a world that is becoming ever more open and connected, the need for security grows ever larger. With connected pacemakers and every element of your life being recorded into large digital databases, the promise and peril of being connected has never been larger. The need for people concerned with cybersecurity, encryption, and cryptography has never been higher.

What you could do with these capabilities

You can work on enterprise security, work as a tester or a quality assurance developer, or go on bug bounty programs that help you earn money for different faults you find in online security. You could be a major part of a cybersecurity startup. There are many career paths in cybersecurity that will help you advance your skills and impact on the industry.

Being proficient at cryptography also means gaining a deeper appreciation of many different mathematical fields from game theory to statistics to dealing with all sorts of differential equations. With those mathematics skills in tow and the ability to read different cryptographic papers, you could transfer the skills you've learned to all sorts of different technical pursuits, should you choose to do so.

How to get started

Cryptography as a whole being a complex field, it's important to gain skills in mathematical logic and understand what you're getting into. Many people in the field will have spent decades of their lives understanding exactly how to be proficient in cryptography. This isn't the easiest technical field to get into without a degree -- so buyer beware. Unless you're utterly fascinated with how security works on the Internet, you may be better off pursuing other paths more sound for beginners.

You shouldn't expect to get a career by focusing your research on theoretical cryptography: that'll only lead to research paths that require advanced degrees. A lot of companies and startups will however, need somebody that can help them implement cryptographic logic.

There are a few resources out there that can help you get started in applied cryptography. This Quora thread delves into just a few of the possible options. With resources such as Coursera courses and this book on Modern Cryptography, you could be on your way to understanding modern cryptography with minimal time invested. If you can understand the basic strands of mathematical thought and logic (summarized here in a series of blog posts) and some of the basics of cryptography algorithms, you should then start being able to read cryptographic papers -- perhaps none more famous than Satoshi's Bitcoin whitepaper.

Once you can read cryptographic papers, it's up to you to build something: a system that can stand the test of multiple hackers. One way to do this more easily from a programming standpoint is to use some of Python's cryptography libraries to play around with basic concepts such as SSL and hashing.

Once you've got the groove of things, you'll want to check out blogs such as Schneier on Security and Krebs on Security to keep on top of ever-evolving security threats and trends. Among all fields, cybersecurity often moves the quickest -- with its association with skyrocketing digital money to protecting some of the most valuable assets on Earth, there can be no hesitation in saying cryptography is an exciting, yet challenging field to be embarking upon.

Career prospects

Cryptography is a booming field with lots of financing going to young up-and-coming companies -- more than $7 billion in total already. With established companies always looking to grow their enterprise security teams (Glassdoor has 15,000 positions alone that call for enterprise security consultants) the field is booming.

Summary

Cryptography is becoming ever more important in a world where more data is being digitized every day. You can become part of this movement even if you don't have a degree -- but don't underrate how difficult it might be.

You want to build virtual reality and video game experiences

What you could do with these capabilities

By now, you've probably heard of how virtual reality is going to change the world. By providing immersive experiences that transcend the screens computer pixels are usually displayed in, virtual reality offers the possibility of creating entirely new virtual worlds - an exciting, and perhaps scary prospect. The field is just beginning to pick up steam, with major players like Oculus being acquired by internet powerhouse Facebook.

How to get started

This overview by Y Combinator, the foremost startup accelerator in the world, helps explain some of the technical complexities of the field of virtual and augmented reality, concepts that are joined at the hip. Virtual reality/augmented reality (known by the acronyms VR/AR) combine a host of hard and interesting engineering and science problems. Elements of human-computer interaction, hardware performance, optics and artificial intelligence come into play. In order to build a completely realistic virtual world, you'll have to spend a lot of time solving different technical issues that arise -- ranging from performance to the realism of virtual avatars.

You'll want to either specialize in one of the domains required for great virtual reality experiences or you'll want to strategize virtual reality experiences within a team of specialists.

You can build virtual reality content with an engine such as Unity or Unreal Engine. Each game engine is well-established technology with large communities and resources to help you learn new techniques. Both technologies are based on either C++ or C#.

You can use a course such as Udacity's Nanodegree in VR development to help you learn how to create VR experiences in an organized fashion. Meanwhile, different community resources can be used for learning for free.

Career prospects

Opportunities to be part of the virtual reality industry abound, whether you're developing the technology required for intelligent virtual agents, or if you work on the hardware behind delivering great VR experiences. In this Quora thread, salary ranges for developers are discussed, though they vary by industry. You can earn about 75k-80k USD if you're building

virtual reality for gaming experiences but if you build things for medical applications, you could earn upwards of 100k USD and more. There are about 1,800 jobs posted on Glassdoor for Virtual Reality Developer.

Summary

Virtual reality is a booming field with tons of potential that interweaves a lot of different technical challenges and fields. As the technology matures, entire industries will be built on top of it. In order for you to get involved, you'll have to play around with the different virtual reality engines that exist right now -- and look to develop your skills in this nascent field.

How to get a job programming (job search strategy)

Now that we've summarized what you can do with your programming skills, the following section is dedicated to helping you get the technical job you need to practice your skills on a daily basis while being paid.

As somebody who has had to perhaps beat a non-traditional path through to programming jobs, you'll need to do things that are slightly different and a little bit more than the norm so that your profile sticks out. Without a formal computer science background, you'll have to show that you can do different technical tasks.

Here are a few things you'll need to do:

1- Build a portfolio of tangible technical tasks and participate in communities

You'll want to demonstrate that you actually can program. There are a few ways to do that: either building your own portfolio site, having an active Github profile or contributing to Stack Overflow.

By being an active participant in different programming communities such as r/learnprogramming or Hacker News, you can also demonstrate that you're an active contributor to ongoing programming discussions -- and you'll be able to start networking with different members of the programming community -- including hiring managers.

2- Do informational interviews and build out a network

Once you have a baseline of connections in different programming communities, you should start mapping out different companies and industries where you want to work. Then, you should start reaching out and finding people who work in those particular companies and industries.

Once you've found people you want to reach out to, do what's called an informational interview -- reach out to them and ask them about what they're working on and what problems they face at work. Here, if you can show that you're a proactive problem-solver and somebody who can respond with empathy and skill to different situations that arise (and

somebody who seems to have done their research and homework on a particular company or industry), then you might not just leave with insight about a field that you're interested in -- you might leave with a potential advocate for your skills in the company, a potential ally in your search for a job.

3- Practice, practice, practice for the coding interview

Once you've got an initial interview lined up, there's no amount of practice that you should limit yourself to. Better safe than sorry. However, one important thing to note is that you should **practice with limitations in mind**. A coding interview is not going to take place with you comfortably able to browse through Stack Overflow and take an hour or so of time to debug -- a coding interview is going to test you by limiting the time and information you have in front of you. Sometimes, coding interviews will actually involve writing down your code on a piece of paper and reciting it over the phone! Be prepared for anything by practicing with those constraints in mind. Force yourself to accomplish programming tasks in a set block of time.

4- Follow up and don't give up

The interview process is always lengthy and it can take a while for companies and hiring managers to get back to you. Never give up -- follow up proactively every once in a while to get a sense of where you are in the process. If a company rejects your candidacy, you can ask them for feedback but be aware that there are legal constraints on what a company can tell you. Once you've gotten your response, accept it with grace and work continuously through the process. Take any feedback you can get and learn from it.

Hacks

I call this section hacks because, while it's an overused word, it's a pretty good description of how you'll have to think and act if you want to get an edge towards building something great or pursuing a career where you don't have what is known as a "traditional" background. These are little tips and tricks I've picked up on how to get into Silicon Valley without the accompanying STEM degrees.

1- Find non-traditional methods of outreach

It can be hard to build a network and reach out to people in the programming community -- in many ways, it's a sort of chicken and egg problem. In order to get hired and noticed, you should have a good network -- but if you come from a non-traditional background when it comes to programming, it can be hard to get that network unless you get hired.

One way to get around that is to network extensively, using solutions such as LinkedIn, AngelList or local meetups to do informational interviews with experts in the field. That's how you can build a network from scratch if you don't have one already.

2- Leverage your unique strengths

If you have a background in a certain domain, leverage that to get an extra edge while looking for programming jobs. If you worked in banking before, talk with financial technology companies. Emphasize your passion for the industry a company is working in, and tailor your profile to a particular industry: it'll help increase your odds.

3- Demonstrate your willingness to work

One advantage you'll have looking for programming jobs is that a lot of people who are established in the industry won't go the extra mile when they're looking for work. It can be easy to differentiate yourself by simply being more willing to demonstrate that you're able to put in the hours. This can be as easy as doing some work for the company before even interviewing with them, perhaps building an app or a landing page filled with solutions to potential problems you've identified.

Perhaps you want to reach out to current users and customers of the company and ask what problems they have and see if you can design something to solve them. Being proactive and showing you can be a problem solver will help you gain that extra edge.

4- Get involved in helping non-for-profits or cities

Want to be able to demonstrate your programming skills? You could volunteer for a local non-for-profit or build open source projects on the open data most cities release. It could be an awesome way to build and highlight your skills.

5- Be involved in the community

Check out your local meetups and see if you can get involved! You might want to even organize your own meetup if you don't see anything around -- this can be a great way to build your personal brand and to get in front of hiring managers.

Go to hackathons and build solutions with different team members, many of whom might be hiring managers or who might be able to recommend you to them.

6- Write about your learnings

As you're learning to program, writing about your challenges and also building valuable resources and tutorials for people who might look to follow your steps will differentiate you as somebody who is obsessed with learning relentlessly and communicating that passion with as many people as possible: a huge differentiator when you're applying for a job or starting your own venture.

7- Make it easy for people to reach you

Whether that's by starting your own personal website or maintaining an active presence on LinkedIn, local community events, programming-oriented communities such as Hacker

News, Stack Overflow or Quora, you'll want your personal profile fleshed out and ready to be shared across these different platforms. Make it easy for hiring managers to find you and to reach out to you.

Job boards/companies willing to hire

Here is a list of different job boards you should go to if you're looking for a programming job and don't have a degree that can be particularly fruitful for your job search.

General

The following job boards often have a selection of general jobs, but they are also useful resources that can be used to find technical jobs -- if you're able to process the information correctly.

LinkedIn

Sometimes it's good to start at the most obvious place: LinkedIn has a large number of technology jobs that you can find quite easily. You can sign up for a free trial of the premium version and quickly look through different jobs.

LinkedIn can also be a great way to research hiring managers and get a sense of what a company is like before you even apply there. You'll be able to see what the organizational hierarchy looks like by scrolling from one profile to another -- and you'll be able to see what skills the company emphasizes, either by looking at the profiles of those who were hired or by using your trial Premium account and looking at job postings or company pages.

You'll want to think about how to optimize your LinkedIn profile so you can get the most out of this career-oriented social network.

Crunchboard

Crunchboard is the job board associated with TechCrunch, a publication that specializes in writing about emerging technologies and new companies. As you can imagine, their job board is filled with a lot of technology and web development positions due to their audience.

Another technique you can use related to this is to look for startups that have just raised a large fundraising round on either TechCrunch or CrunchBase and reach out to hiring managers or executives at those companies: immediately after raising a fundraising round, a company is in aggressive growth mode, and is most likely looking to hire many qualified people to fill different and interesting job roles.

Hacker News

Besides being a great repository of technical articles and a community that curates people who are interested in the cutting edge of technology, Hacker News also serves as a job

portal of sorts for Y Combinator companies -- technology companies that might be as young as a two-person startup and also those who have started full maturing (as an example, Dropbox, Airbnb, and Quora were all at one time or another incubated by Y Combinator). The jobs section of the site features different YC companies and their hiring needs. There are also monthly threads started by a bot called Ask HN: Who is hiring? --where discussion about urgent job opportunities are surfaced that may be hard to find elsewhere. Here's an example of the latest "who's hiring" thread in May 2017.

By commenting on different articles and reaching out to different members in the Hacker News community, many of whom are senior figures in the startup world, you might also find your way to different mentors -- and somebody who can introduce you to the right hiring manager.

AngelList

AngelList is an online repository for different startups. The jobs on offer here tend to be with earlier stage companies working at the edge of technology. One great perk about this is that entrepreneurs may be more willing to accept people from non-traditional backgrounds to work with them -- especially if you're willing to accept and maybe even embrace the risk that comes with working in a startup.

GitHub

GitHub, the living repository of code collaboration, also offers a selection of curated jobs for developers around the world. You can even search by programming language here, ensuring the best match for your skills.

Stack Overflow Jobs

Stack Overflow, the popular Q&A site for programming questions, offers a selection of different programming jobs, many of them posted by hiring managers who are trying to find top talent within the Stack Overflow community.

Glassdoor

Glassdoor is an interesting job board since you'll be able to see what employees think about the company and you can get some transparency on the salary range the company offers as well. All in all, Glassdoor is a great general place to find technology jobs -- but its greatest value probably rests in the additional data on employee satisfaction and approximate salary ranges that can help guide your career decisions.

Mashable

Mashable, the popular content repository based out of New York City, has a job board as well with a lot of different technology companies and job positions.

The Muse

The Muse is a unique jobs resource, with tons of personalized career coaching and resources related to career development. It can be well worth browsing the content on the site itself if you want to learn about salary negotiation, interviews and career progression from a somewhat general perspective. The jobs board section also boasts a selection of technical and developer jobs.

Startupers

Another community oriented towards posting startup jobs, many of them programming-related.

Dice

One of the leading repositories of tech jobs in the world, Dice offers nearly 80,000 jobs in technology for you to consider.

Cybercoders

Run by a placement agency for engineers, Cybercoders offers an easy way to search across 10,000+ different technology jobs across different industries.

Front-End/Design

The following job boards focus on jobs that are oriented towards front-end work and user design. Check these out if you're looking to work on how the user experience of digital products feels to different people.

Smashing Magazine

Smashing Magazine is one of the premier web development and design resources on the web. They offer a selection of jobs tailored to front-end web development.

Codepen Jobs

Codepen is a great interactive sandbox for front-end code, where you can use HTML/CSS/JavaScript to generate awesome interactive graphics -- or where you can copy those snippets of code for use on your own website. The site also offers a job board that tilts towards front-end web development and design jobs, as you might expect.

Web Development

The following job boards will help you hone your skills in web development if that's the technical career path you want to choose.

Sensational Jobs

Sensational Jobs curates a selection of different positions for web professionals of all sorts and stripes.

Wordpress Jobs

The official Wordpress jobs board will help you curate a selection of jobs in web development specifically focused on building things with the Wordpress platform -- a popular, open-source content-management system that serves as the back-end framework for nearly one in six of all websites on the Internet.

WPHired

WPHired is another great job board if you're looking for development jobs oriented around Wordpress.

Data Science

Data science entails a mix of statistics, programming and communication skills that are quite specialized. Oftentimes, data science job postings will be found in these specialized communities that have grown to help support the data science community.

Kaggle Data Science Jobs

Kaggle is an online community centered around machine learning competitions. Here, they've used their reach in the data science community to curate a selection of data science jobs for you.

Data Elixir Job Board

Data Elixir offers a newsletter filled with data science resources, and also curates this job board to help data science jobs seekers.

KDNuggets Jobs

KDNuggets is one of the leading data science content hubs, filled with useful tutorials and resources to help you understand different topics in data science. This static jobs page is updated quite frequently with different job postings in data science.

Mobile Development

The following job boards curate different opportunities for those looking to build mobile apps on a variety of platforms. The most common tend to be iOS or Android-oriented.

Android Jobs

Android Jobs curates a selection of jobs for developers interested in building Android applications. Come here if you want to make your mark in mobile development.

Core Intuition

Core Intuition features a selection of curated Mac Cocoa and iOS development jobs -- if you want to develop apps for Apple products, there are few job boards as well-placed as Core Intuition to help you advance along that career path.

Language-Specific

The following job boards are specific to a type of programming language. It can be a handy place to look if you plan to specialize in one language and grow your career there.

AngularJobs

AngularJobs is a job board curated around the Google-backed front-end JavaScript framework. Come here if you want to work with Angular.js and develop your JavaScript skills.

We Work Meteor

We Work Meteor is a job board focused on meteor.js, a full-stack JavaScript framework that can handle every part of web development. If you're interested in pursuing a career using Meteor as your tool of choice, or if you're interested in developing your JavaScript skills -- coming to this job board wouldn't be a bad choice.

Ruby Now

Ruby Now is a job board focused on curating Ruby on Rails specialists. Given the extensive use of Ruby on Rails for web development, you'll mostly be working with web development positions if you look through this job board -- though there are some more senior oriented positions in back-end development.

Python Jobs (official Python website)

Python.org (the official centerpiece of the Python programming community) hosts a small repository of curated and interesting jobs that involve the use of Python.

Python Jobs

Python Jobs (unaffiliated with the official Python programming community) is a great free resource for looking up Python jobs and web development jobs associated with the Django framework.

R-Users

R-Users is the place to go if you're proficient in R or if you're a statistician looking to get some work developing their programming skills in R.

Remote

One of the luxuries of working in a technology-oriented career is the ability to be able to work remotely from anywhere in the world. The following job boards curate remote opportunities in technology.

We Work Remotely

We Work Remotely curates a selection of jobs that are online and remote, with a section dedicated to just programming jobs.

Remote OK

RemoteOk is another job board that curates different jobs where remote work is available. They have a large selection of technology jobs and they have a neat categorization of the highest paying remote jobs and the technologies involved with it.

AngelList Remote Jobs

AngelList curates a selection of startup jobs where it's acceptable to work remote. Again, as with the rest of AngelList, most of the jobs revolve around earlier stage startups -- so be aware of that as you browse through this selection.

Upwork Jobs

Upwork is a curated marketplace where freelancers can meet potential employers. The entire process of payment, job search, and work management can be completely managed on Upwork. As a result, it can be a great place to find remote work in different technical fields.

Where programming is going

A lot of questions arise here about the direction of programming and what career paths will be available. What skills will be in demand in 10 years? Will there be a dramatic shift in the demand for programming skills if the technology industry undergoes a dramatic restructuring or a recession similar to the software stock bust in the early 2000s?

These are apt questions to ask. In a field where change is the only constant and where older programming frameworks constantly become outdated, it can be a terrifying time for those who are being disruptive and those being disrupted.

There's a lot of talk about programmers themselves being automated and displaced in an age where AI is placing a lot of different jobs at risk. I get approached a lot about questions like this at Springboard, where I worked for an education company that training people towards new career fields. What I tell people is that for now, jobs are certainly safe and prosperous for people who can bring programming skills and intelligence to solve different problems. In the future, however, more rote or routine parts of programming such as web integration of HTML and CSS elements or perhaps certain parts of application building and data analysis which are repeatable or arduous tasks more suited to machine repetition might be at risk.

However, while it seems some more functionary parts of programming might be automated, it's also important to note that as new programming tools and approaches start taking precedence, entire new economies will be built. Could you have imagined a decade ago that social networks and the data they would create would spawn billion-dollar companies and trillion-dollar ecosystems? Or that so many of the world's goods would flow through digital storefronts? As new infrastructure is built, and new innovations are created, new economies will arise. And while I'm not sure exactly what the future holds, I do think that anybody who fiercely commits to solving problems with technical skills and who is tenacious about learning new skills will succeed.

Learning Paths

Sometimes, it's not just the resources that matter, but the way that they're presented.

Here are two curated learning paths for common programming scenarios, fully fleshed out from beginning to end. It'll give you a plan to organize your learning, and attack other topics if you wish.

How to learn data science and deep learning in Python

Data science and deep learning are very popular topics: with the rise of artificial intelligence, programmers have been able to do everything from beating human masters at Go to replicating human-like speech. At the foundation of this fantastic technological advance are programming and statistics principles you can learn.

Here's how to go about doing it:

Python Basics

Before you learn how to run, you have to learn how to walk. Most people who start learning machine learning and deep learning come from a programming background: if you do, you can skip this section. However, if you're new to programming or you're new to Python, you'll want to take a look through this section.

Codecademy for Python

Codecademy is an online platform for learning programming, with free interactive courses that encourage you to fully type out your code to solve simple programming problems.

Introduction to Python for Data Science

This interactive Python tutorial is created by Datacamp, and is more suited to introducing how Python basics work in the context of data science.

11 Great Resources to Learn and Work in Python

This list of resources will point you to great ways to immerse yourself in Python learning. It's a broad list filled with different resources that will help you, no matter your learning style.

Installing Jupyter Notebook

These are instructions for installing Jupyter Notebook, an intuitive interface for Python code. You'll have all of the important Python libraries you need pre-installed and you'll be easily able to export out and show all of your work in an easy-to-visualize fashion. I strongly suggest that you use Jupyter as your default tool for Python, and the rest of this learning path assumes that you are.

Statistics Basics

In order to understand data science and machine learning principles, you not only have to learn the programming behind it -- you'll also have to learn statistics. Here are some resources that can help you gain that fundamental knowledge.

Khan Academy, Math, and Statistics

Khan Academy is the largest source of free online education with an array of free video and online courses. This section on Khan Academy will teach you the basic statistics concepts you need to know to understand machine learning, deep learning and more -- from mode, median, mean to probability concepts.

Probabilistic Programming & Bayesian Methods for Hackers

This book will delve into Bayesian methods and how to program with probabilities. Combined with your budding knowledge of Python, you'll be quickly able to reason with different statistical concepts. It's a book the author gave out for free -- and its deeply interactive nature promises to engage you into these new concepts.

Pandas

The main workhorse of data science in Python is the Pandas data science library, an open-source tool that allows for a tabular organization of large datasets and which contains a whole array of functions and tools that can help you with both data organization,

manipulation, and visualization. In this section, you'll be given the resources needed to learn Pandas.

Cooking with Pandas

Julia Evans, a programmer based in Montreal, has created this simple step-by-step tutorial on how to analyze data in Pandas using noise complaint and bike data. It starts with how to read CSV data into Pandas and goes through how to group data, clean it, and how to parse data.

Official Pandas Cookbook

The official Pandas cookbook involves a number of simple functions that can help you with different datasets and hypothetical transformations you might want to do on your data. Take a look and play with it to extend your knowledge of Pandas.

Data Exploration and Wrangling

Before you can do anything with the data, you'll want to explore it, and do what is called exploratory data analysis (EDA) -- summarize your dataset and get different insights from it so you know where to dig deeper. Fortunately, tools like Pandas are built to give you relevant and surprisingly deep summary insights into your data, allowing you to shape which questions you want to explore next.

By looking through your dataset from afar, you'll already be able to understand what faults the dataset might have that will keep you from completing your analysis: missing values, wrongly formatted data etc. This is where you can start processing and transforming the data into a form that you want to answer your questions. This is called "data wrangling" -- you are cleaning the data and making sure that it is able to answer all of your questions in this step.

Python Exploratory Data Analysis with Pandas

This article from Datacamp goes through all of the nuts and bolts functions you need in order to take a slightly deeper look at your data. It covers topics ranging from summarization of data, to understanding how to select certain rows of data. It also goes into basic data wrangling steps such as filling in null values. There are interactive embedded code workspaces so you can play with the code in the article while you are digesting its concepts.

A Comprehensive Introduction to Data Wrangling

This blog article from Springboard is filled with code examples that describe how you can filter data, detect and drop invalid/null values from your dataset, how to group data such that you can perform aggregated analyses on different groups of data (ex: doing an analysis of survival rate on the Titanic by gender or passenger class) and how to handle time series data in Python. Finally, you'll learn how to export out all of your work in Python so that you and others can play around with it in different file formats such as the Excel-friendly CSV.

Pandas Cheat Sheet

This Pandas cheat sheet, hosted on Github, can be an easy, visual way to remember the Pandas functions most essential to data exploration and wrangling. Keep it as a handy reference as you go out and practice some more.

Data Visualization

Data exploration and data visualization work together hand-in-hand. Learning how to visualize data in different plots can be important is seeing underlying trends.

Beginner's Guide to Matplotlib

This legend of resources on the official matplotlib library (the workhorse library for Python data visualization) will help you understand the theory behind data visualization and how to build basic plots from your data.

Seaborn Python Tutorial

The Seaborn library allows people to create intuitive plots that the standard matplotlib library doesn't cover easily: things like violin plots and box plots. Seaborn comes with very compelling graphics right out of the box.

Introduction to Machine Learning

Machine learning is a set of programming techniques that allow computers to do work that can simulate or augment human cognition without the need to have all parameters or logic explicitly defined.

The following section will delve into how to use machine learning models to create powerful models that can help you do everything from translating human speech to machine code, to beating human grandmasters at complex games such as Go.

It's important before we get started implementing ideas in code that you understand the fundamentals of machine learning. This section will help you understand how to test your machine learning models, and what statistics you should use to measure your performance.

A Visual Introduction to Machine Learning

This handy visualization will allow you to understand what machine learning is and the basic mechanisms behind it through a visual display of how machines can classify whether a home is in New York or in San Francisco.

Train/Test Split and Cross-Validation in Python

This article explains why you need to split your dataset into training and test sets and why you need to perform cross-validation in order to avoid either underfitting or overfitting your data. Does that seem like a lot of jargon to you? The article will define all of these different concepts, and show you how to implement them in code.

Sci-kit Learn

Sci-kit learn is the workhorse of machine learning and deep learning in Python, a library that contains standard functions that help you map machine learning algorithms to datasets. It also has a bunch of functions that will allow you to easily transform your data and split it into training and test sets -- a critical part of machine learning. Finally, the library has many tools that can evaluate the performance of your machine learning models and allow you to choose the best for your data.

A Gentle Introduction to Scikit-Learn

This post introduces a lot of the history and context of the Sci-Kit Learn library and it gives you a list of resources and documentation you can pursue to further your learning and practice with this library.

Scikit-Learn Documentation

The official scikit-learn documentation is filled with resources and quick start guides that will help you get started with Scikit-Learn and which will help you entrench your learning.

Regression

Regression involves a breakdown of how much movement in a trend can be explained by certain variables. You can think about it as plotting a Y or dependent variable versus a slew of X or explanatory variables and determining how much of the movement in Y is dependent on individuals factors of X, and how much is due to statistical noise.

There are two main types of regression that we're going to talk about here: linear regression and logistic regression. Linear regression measures the amount of variability in a dependent factor based on an explanatory factor: you might, for example, find out that poverty levels explain 40% of the variability in the crime rate. Logistic regression mathematically transforms a level of variability into a binary outcome. In that way, you might classify if a name is most likely to be either male or female. **Instead of percentages, logistic regression produces categories.**

You'll want to study both types of regression so you can get the results you need.

Simple and Multiple Linear Regression in Python

This informative Medium piece goes into the theory and statistics behind linear regression, and then describes how to implement it in Sci-Kit Learn.

Building a Logistic Regression in Python, Step-by-Step

This Medium tutorial uses the Sci-Kit Learn tools available to implement a logistic regression model. The amount of detail in each step will help you follow along.

Clustering

Another type of machine learning model is called clustering. This is where datasets are grouped into different categories of data points based on the proximity between one point and other groups of points.

An Introduction to Clustering and different methods of clustering

Analytics Vidhya has presented this comprehensive introduction to clustering methods: it's good to get a handle on this theory before you try implementing it in code.

Customer Segmentation using Python

This article from Yhat demonstrates how to do simple K-means clustering across different wine customers. It'll take your learning in Pandas and Scikit-Learn and combine them into a useful clustering example.

Deep Learning/Neural Networks

Neural networks are an attempt to simulate how the human mind works (on a very simplified level) in computational code. They have been a great advance in artificial intelligence -- and while in some ways they are a black box of complex algorithms working in tandem to learn how data generalizes, their practical applications have exponentially multiplied in the last few years. Deep learning encompasses neural networks as well as other approaches meant to simulate human intelligence.

In a huge breakthrough, Google's AI beats a top player at the game of Go

This short Wired article isn't a technical tutorial: it's the recounting of an epic match between a human grandmaster at Go, a game that was supposed to be so complex for computers to win that technology to do so wasn't supposed to come until around the 2030s. By leveraging the power of neural networks, Google was able to bring AI victory forward some two decades or so. This article should give you a great glimpse at the potential and power of neural networks.

A Beginner's Guide to Neural Networks in Python and SciKit Learn 0.18

This example-laden tutorial uses the neural networks module in the Scikit-Learn library to build a simple neural network that can classify different types of wine. Follow along and play with the code so you can get a feel for how to build neural networks.

Develop Your First Neural Network in Python With Keras Step-By-Step

This tutorial from Machine Learning Mastery uses the Python implementation of the Keras library to build slightly more powerful and intricate neural networks. Keras is a code library built to optimize for speed when it came to experimenting with different deep learning models.

Big Data

Big data involves a lot of volume and velocity of data. It's an amount of data, measured in petabytes, that can't be processed easily with tools like Pandas, which are based on the processing power of one laptop or computer. You'll want to scale out to controlling many processors and servers and passing data through a network to process data at scale. Tools that allow you to map and reduce data between multiple servers and others such as Spark and Hadoop play an important role here. It's time to take the learning you've had before this and apply it to massive data sets!

Get Started With Pyspark and Jupyter Notebook in 3 Minutes

This blog post will help you get set up with PySpark, a Python library that brings the full power of Spark to you in the Jupyter Notebook format you've been used to working in. PySpark can be used to process large datasets that can go all the way to petabytes of data!

PySpark Video Tutorial

This video tutorial will help you get more context about PySpark and will provide sample code for tasks such as doing word counts over a large collection of documents.

Using Jupyter on Apache Spark: Step-by-Step with a Terabyte of Reddit Data

This tutorial from Insight goes a little further than installation instructions and gets you working with Spark on a terabyte (that's 1024 gigabytes!) of Reddit comment data.

Machine Learning Evaluation

Now that you've learned a baseline for all of the theory and code you need to implement machine learning in practice, it's time to learn what metrics and approaches you can use to evaluate your machine learning models.

Metrics to Evaluate Machine Learning Algorithms in Python

In this tutorial, you'll learn about the different metrics used to evaluate the performance of different machine learning approaches. You'll be able to implement them in Scikit-Learn and Jupyter right away!

Model evaluation, model selection, and algorithm selection in machine learning

This long six-part series (check the end of this blog post for more posts after) goes deep into the theory and math behind machine learning evaluation metrics. You'll come out of the whole thing with a deeper knowledge of how to measure machine learning models and compare them against one another.

Suggested daily routine

Learning isn't often a static thing. You need ongoing practice to master a skill. Here's a suggested learning routine you can implement in your day to make sure you practice and expand your knowledge.

Here's my suggested daily routine:

1) Continue working on something in machine learning at all times
2) Go to StackOverflow, ask and answer questions
3) Read the latest machine learning papers, try to understand them
4) Practice your code whenever you can by looking through Github machine learning repositories
5) Do Kaggle competitions so you can extend your learning and practice new machine learning concepts

How to learn web/mobile development in JavaScript

This learning path has been curated to teach you how to do both web and mobile development with JavaScript frameworks. You'll be able to build interactive web applications that will be able to take data back and forth.

Introduction to JavaScript

JavaScript is a high-level programming language that is used for most of modern web development. Through the use of powerful frameworks, you can easily build many things and experiences for your online users. The following section will get you up to speed with this modern and versatile programming language.

Introduction to JavaScript

Codecademy offers an interactive track where you can learn the basics of JavaScript by playing around with code in an interactive module. Take advantage of this to practice and start establishing your JavaScript skills.

Javascript Tutorial by W3Schools

This JavaScript tutorial by W3schools is focused on breaking down things topic by topic. Feel free to come through here after you've finished the Codecademy track and take the time to practice parts of your learning you feel like have slipped.

r/LearnJavaScript

The Reddit community is large and varied, and it includes niches like this JavaScript subreddit community that will help you pick out the best learning resources, and which can be a place for you to ask questions and connect with fellow learners.

Introduction to Web Development Basics

Now, you'll need to apply your JavaScript skills to web development purposes. In order to do that, you have to learn some of the basics of web development.

HTML Basics

This course by Codecademy helps you learn what different HTML elements do and how they affect the way your website renders. HTML is the building block language of web development. In mastering HTML, you will master where and how different web page elements flow into one another.

CSS Tutorial

This website covers an introduction to CSS and then a deeper dive if that's what you'd like. It'll help you understand both broadly and specifically how CSS interacts with HTML -- you can think of HTML as the foundation of a house, and CSS as the layer of logic that helps dictate what each house element looks like and where they belong in the house. Going through this CSS tutorial site will help you understand how to implement CSS and how to use it to augment your HTML.

The MEAN Stack

The MEAN Stack is a popular new approach to end-to-end web development that contrasts with the traditional LAMP stack (Linux as the operating system, Apache as the server processor, MySQL as data storage and PHP as back-end programming language). It entails Mango for data storage, Express.js for routing, Angular.js for front-end processing and Node.js for server-side processing. Different JavaScript frameworks can now act together to do all of the end-to-end lifting for web development.

An introduction to the MEAN stack

This tutorial not only gives you a broad overview of the MEAN stack, what it means and how the pieces interact with one another -- it also dives down a little bit deeper into the specifics of each component.

Mongo/JSON

Mongo is the JSON-based data storage system used in place of MySQL for most Javascript applications. It's commonly called NoSQL technology. In this section, you'll learn more about Mongo and you'll learn more about the JSON data format often used to transfer data back and forth in JavaScript frameworks.

A Non-Programmer's Introduction to JSON

This simple tutorial breaks down the JSON data format in a way that makes it comprehensible even for non-programmers.

MongoDB Tutorial

MongoDB is a database solution that runs exclusively on JSON and collating together different JSON snippets. While many people also use SQL to store and manage their data with JavaScript apps, MongoDB is also quite popular. This tutorial will walk you through step-by-step how to update, delete, and insert data into a MongoDB database.

Angular.js

Angular.js is a JavaScript framework that can be used to program simple interactive web applications. You don't even need a proper back-end to use it, making it an ideal resource to do some quick prototyping, if need be.

Angular.js learning paths

Pluralsight offers a collection of curated resources to teach you Angular.js, with an organized series of mini-courses that tackle Angular concepts from the basics to more advanced concepts.

Angular.js - Egghead

Egghead offers a series of video courses on Angular.js concepts -- you'll be ready to build your own Angular apps after working through the different mini-courses.

Node.js and Express.js

Node.js and Express.js are often used together to create the server-side or back-end logic of JavaScript applications. Node.js is a web server software package that uses JavaScript for its logic. Express.js uses a series of API calls to do routing and direct traffic to the right place. The two used in conjunction can help flesh out the heavy data lifting required to create truly powerful JavaScript web apps.

Ultra fast applications using Node.js

This tutorial will teach you the basics of web applications and how Node.js transmits information back and forth. You'll be able to build your first basic web application using Node after looking through this tutorial.

LearnYouNode

For a more interactive experience, use LearnYouNode and type in commands and snippets of code into your terminal, validate the code according to the guidelines and directions set, and learn Node.js for much win!

Express Tutorial

This tutorial from Mozilla will teach you how Node.js and Express.js work together to form a strong foundation for any web application you build. You'll learn how Express can help facilitate the routing logic of Node modules and how you can build complete web applications using the two.

React.js

React is a powerful JavaScript library maintained by Facebook which can manage and control what user interfaces look like.

Build with React

This simple interactive module will allow you to play with React.js right in your browser, accelerating your learning of the basics through interactivity.

Top 5 Tutorials for Getting Started with React

If the tutorial above wasn't the best for you getting started, there are four more that this article can help you get started with. Try each one of the different resources to entrench your learning in React.

Introduction to Mobile Responsiveness

To understand how to do mobile development properly, you have to understand what differentiates it from web development. The best way to start is to learn mobile responsiveness principles: the sets of rules that can help transfer the logic and design of a web application to something that can be friendly across a whole bunch of different screen sizes.

Responsive Web Layouts for Mobile Screens: Intro, Tips, and Examples

This tutorial for designers will help you understand what mobile responsiveness means and why it is so needed. Finally, you'll get tips and examples to help you make sure your efforts are responsive to mobile needs.

Responsive Web Design Basics

This in-depth tutorial from Google will cover specifically how you should break down your page into responsive elements. It's well-done and comprehensive, and you'll want to take this advice as Google plays a huge role in determining web trends by penalizing or favoring certain behaviors in their search engine algorithm.

React Native

React Native is Facebook's project to extend the React.js framework so that you can build a native app for mobile ecosystems. A native app has more functionalities than HTML5 or hybrid apps that don't fully tap into the power of mobile devices.

React Native

This is the official tutorial and documentation for React Native produced by Facebook. It includes the blog and community sections for this powerful technology.

React Native Introduction

Part of the same website Facebook uses to host documentation on React Native includes this handy tutorial to the basics.

Suggested Daily Routine

Now that you've read through some theory and gotten a sound understanding of how to move into two different programming fields, it's time to talk about practice.

Learning doesn't involve just reading and memorizing different concepts: in order for you to really absorb programming concepts, you have to apply them in practice. That'll look different for you depending on your learning style and what your learning goals are. However, I've found the following daily routine to work quite well (and I've tried to set aside time to make sure I can accomplish this daily routine.)

This is a generalized version of the daily routine from the machine learning path:

1. Create a project where you can test out your technical skills. Continue working on it at all times.
2. Go to StackOverflow, ask and answer questions related to the field.
3. Read technical tutorials, try to understand them.
4. Practice your code whenever you can by looking through relevant Github repositories.

5. If there are competitions and hackathons go and attend them! Meetup will help you find events where you can practice your skills.

Resources

Here are some resources that will be generally helpful for you on your programming journey, no matter what particular path you want to take.

Programming Communities

r/Learnprogramming

A subreddit within the larger Reddit community, this subreddit is dedicated to programming resources and for programming learners. It's a great resource where people will upvote the top resources to learn programming for your consumption.

Stack Overflow

One of the largest Q&A sites in the world, Stack Overflow has its beginnings in the programming community. Here, you can see a variety of programming challenges and supplied answers from experts in different programming communities.

Hacker News

Hacker News is an initiative of the Y Combinator startup incubator. It's a daily curated feed of the most valuable and relevant technology and programming news out there. Community members are responsible for upvoting and downvoting both articles and comments, ensuring that quality submissions come to the forefront.

Quora Programming

Quora is a large Q&A community with many sophisticated answers. With many of the initial users based in Silicon Valley, the site has become a hotspot for reaching out to intelligent and technically skilled folks.

Slashdot

Slashdot is a large programming community filled with IT professionals: it tends to be filled with people who use SourceForge.

Repositories of code

Github

Github is the world's largest living repository of code. The code here is updated by different contributors on an almost-hourly basis, with many of the fundamental building blocks of

different programming languages constantly being hosted and upgraded here. Look through different blocks of code here, contribute some code of your own, or host projects on Github for collaboration. You can also search for the “awesome” repositories to get a list of curated resources on different programming topics.

Bitbucket

Bitbucket is another set of Git repositories, more suited to the needs of distributed teams. You can use it to upload your code and you can take a look at other repositories. The main difference between it and Github is that you can have unlimited private repositories, unlike Github’s pricing when it comes to making repositories private. While this makes Bitbucket much more attractive to private teams, it also means that most of the open-source projects out there are hosted on Github, which is more attractive based on the large community of programmers actively looking over open-source projects.

Wikis

Learnprogramming Wiki

The learnprogramming subreddit community has already been mentioned above as a great resource. This Wiki is a collaborative effort between members of that community to create a living, valuable resource that can help you with the very basics of code, from formatting questions to how to debug.

Wikibooks

Wikibooks is a living library of different user-contributed books. Many of them are on programming topics such as this Wikibook on C++ programming.

Kaggle Wiki

The Kaggle Wiki is a data science focused Wiki filled with different resources in the space. It’s the creation of Kaggle, an online community of data science admirers who come together to compete on the best machine learning models -- you can be certain that the Wiki will contain a lot of resources that will be valuable to your learning journey on programming and data science.

Interview with a coding learner

In this section, I wanted to make all of the theoretical learnings very real and actually bring you the real experience of somebody who was a self-taught coder. David Ernst went through Hack Reactor, an elite programming boot camp and then emerged as the CTO of Numer.ai: he’s now a technical entrepreneur focusing on bringing liquid democracy to the fore with United.Vote. Here are some of his answers to questions I asked him about the journey of self-taught programmers.

1) **How did you learn programming without a degree?**

David was 9 years old when he first started programming -- he was self-taught from a young age with different books and different camps. While he did not have much formal education in university around computer science, he did take a few courses - though he focused largely on math and philosophy. He was mostly self-taught, depending on the Internet as a resource and as a set of communities that could help him maximize his learning.

2) **What were the most helpful resources for you?**

The most helpful were always communities of learners who got together and collaborated with one another both from a resources perspective but also for human support. Nowadays there are two specific resources David recommends: one is CodeCombat, built by a friend -- a game that helps people learn programming by introducing new concepts progressively through a video game-like setup. Another is Project Euler, which is amazing for practicing the theory of programming challenges.

David notes that the really awesome thing about programming is that you get immediate feedback on whether you're right or wrong on something. You can quickly learn by doing by hacking through and trying out different things.

3) What are some of the difficulties inherent in informal education? What are some of the advantages?

The most valuable thing you can get from a formal computer science education is four years of forced practice. In informal education, you have to be highly motivated and disciplined to replicate that kind of experience.

However, the flip side of it is that when you are self-taught, you can choose to hang your programming practice on topics that you are passionate about, making motivated learning easier. This isn't to take anything away from formal education, which can be incredibly valuable as well. These are simply two modes of learning, each one bearing an advantage and a disadvantage.

4) How did you start your career?

In 2013, David took on what he thought was a temporary gig at a startup because the challenges seemed interesting -- from there, he grew attached to the project. Early-stage startups will often take a chance on people who might be a little bit less experienced.

David enjoyed that experience so much, he decided to buckle down and do Hack Reactor, an intensive boot camp. He wanted to take three months off to really sharpen his skills in the area and work through different problems. He was very happy with his choice.

5) Is a boot camp worth it?

David was happy with his choice then, but notes that boot camps are becoming saturated -- even quality ones like Hack Reactor may start to struggle with increasing demand to become a boot camp graduate while maintaining their desired baseline of student quality.

Now that you have a solid foundation in terms of programming skills and know-how, as well as different career resources and some tangible insights into the real career progressions of coding learners, here are two sections on the most common paths one can take to acquire a programming job without a degree -- either a coding interview for a company looking for a programmer or tips on how to make it as a freelancer and technical entrepreneur.

Common interview tips

Here are some interview tips for different programming interviews.

1) Prepare with constraints

Most people prepare for coding interviews by just working through their normal day-to-day flow: however, coding interviews are a different beast to that. Oftentimes, interviewers will ask you to scrawl code on a paper and read off of it, or do the famed whiteboard interview where you have to put everything together without the use of a computer or the ability to really type things through.

You don't want to prepare for coding interviews by doing much of the same. Practice programming under all sorts of constraints, from time limits to scrawling algorithms on paper. That way, when you're doing the interview under constraints and facing time pressure and the pressure of trying to get a job, you'll be on the right path.

2) The process is as important as the answer

When you're working on different questions through the coding interview, keep in mind that the way you structure your answer and how you come to different conclusions matters more than getting the "right" answer. It's not very expensive to correct programming mistakes, but it is expensive to hire people who don't have the right logic and systems of thinking about programming.

3) Practice makes perfect

You'll want to practice answering as many sample questions and completing different practice exercises before you sit down and have the real programming interview. Here are some resources to help you get started.

Resources

Cracking the Coding Interview is a great book for thinking through and prepping for a programming interview. Interview Cake offers free practice interview questions and a weekly email newsletter that offers a practice problem that will keep you sharp. Pramp is an interactive platform as well that will allow you to practice problems with others.

How to be a technical entrepreneur/digital nomad

If you don't want to apply your skills to a settled career with an established company, you can go and become a technical entrepreneur, a freelancer, or somebody who creates their own company. Here are a few resources and tips to get you started.

NomadList

NomadList helps classify cities according to different variables such as Internet speed, safety, nightlife etc. to find you the best cities for you to stay in if you wanted to become a digital nomad. It's a critical first resource to check if you're looking for the best spot to do your best work.

Digital Nomad Visa Tips

This resource will help you understand what visa types and other financial/logistical details you need sorted before you start your digital nomad adventure. Make sure you're on the legal side of things with this handy guide. While it's written in the context of British citizens, it can be useful for anybody else.

How I Built a Startup While Travelling to 20 Countries

This article talks about compelling reasons why travel makes sense for entrepreneurs and breaks down the benefits as well as talking through how to be productive and work while travelling.

Some More Resources

I wrote this article a while back about how to become a freelance data scientist. This Quora thread is a great discussion of how to get started on the freelance route.

Conclusion

I hope this guide has been helpful to you and that you can use it to shape the technical career of your dreams, even if you don't have a computer science degree. If you have any suggestions for the guide or just want to reach out, don't hesitate to find me at thinkthethoughtbasin@gmail.com!