codeop

5 Steps to Starting a Successful Career in Programming

Table of contents

Coding is for everyone

Like the Backstreet Boys told us back in September of ‘97, it doesn’t matter who you are, where you’re from, what you do, or whether you’re just starting today...

You can learn to code and build a successful career in tech.

You’ve probably already done some research on how to start coding. You may even have found yourself joining online coding communities. And by now you’ve probably read countless success stories of coders who’ve made it big.

But let us guess: Those stories are always about those boys who started programming at the age of 12. Or those geniuses who learned everything by themselves. Or those visionary billionaires who hired a team to build a revolutionary app.

As you go through those stories, you see the categories of these successful people—categories that you might not exactly identify with. If this has ever made you feel like coding isn’t for you, we’re going to stop you right there...

The truth is that more than 90% of programmers aren’t like that, and getting into tech isn’t as difficult as it might appear to be.

The fact that you’re reading this already proves that you have the desire to build a career in programming.

Our guide will show you the next steps on how to get there.

Step 1:
Know your why

First things first.

Understand why you want to learn how to code.

Maybe you're just bored and want to try something different. Or maybe you want to be able to advance in your current job. Or maybe you want to have a better quality of life (you've heard that software developer/engineer jobs pay well).

Whatever your reason may be, you should have it clear in your mind.

Learning how to code is not going to be an easy journey, so being firm about your why will help you get going and keep going as you continue along the learning path.

Step 2:
Explore what field you want to go into

Getting a job "in programming" or as "a coder" is actually a vague career description, so let's try and narrow it down.

There are a **lot** of jobs in the field. You can be a:

- Computer programmer
- HTML Developer
- JavaScript Developer
- Software Developer
- Software Engineer
- Web Developer
- Front-End Developer
- Back-End Developer
- Full Stack Developer
- Database Administrator
- Computer Systems Analyst
- Computer Systems Engineer.
- Software Automation Engineer

- JavaScript Developer
- UX/UI Designer
- Data Scientist
- Data Analyst
- Technical Solutions Engineer
- Network Systems Administrator
- QA Team Lead
- Mobile App Developer

It's important that you have a general understanding of the different paths. That way, you can decide which one might be best/more interesting for you.

You can start with a rabbit-hole like Google search, but attending online events about tech, joining coding communities, and taking online quizzes will help a lot.

Let's talk about some of the most common roles so you can see which coding jobs might appeal to you.

Quick note: These job titles might vary as they can be interchanged in the industry from time to time.

Web Developer

If you're a huge fan of instant gratification, you'll probably enjoy this. Web developers listen well to their clients' needs and problem-solve to give them the best website possible for their business. How a website looks and functions are the direct results of a web developer's work.

At the end of a project, you have a working, accessible website to show off your hard work. Web developers do well when they can show a portfolio of their work and have a deep understanding of coding.

We've provided a break down of the 3 different types of web developer roles below.

FRONT END DEVELOPER

A front end developer works on the configuration and design of everything internet users see when they use a website or app. They focus on the interactive elements of web pages in terms of design and functionality.

Front end development is important for the usability and performance of the website. The developer must ensure that the website is accessible, easy to use and has a high-performance on all devices and screen sizes. This tends to be an attractive role for those developers who have a strong interest in graphic design.

Important languages:

- HTML
- CSS
- Javascript

BACK END DEVELOPER

A back end developer focuses on the server-side of web applications, which are the parts of a software or web application that the normal user never sees.

This includes:

- application logic
- databases
- Application Programming Interface (API)
- servers
- and other backend processes.

Backend developers are responsible for building and maintaining the logic and functionality needed to power the user-facing components of a website.

Their backend work enhances the usefulness of the user-facing elements built by front end developers. This enables web apps to perform properly, keeping them fast and efficient.

Important languages:

- JavaScript & Node.js
- Python
- PHP
- Java
- Ruby

FULL STACK DEVELOPER

A full stack developer is a well-rounded professional with hybrid skills in frontend and backend development.

This position combines the responsibilities of both front- and back end web developers, which means they're skilled at using frontend and backend frameworks and languages. Additionally, their expertise extends to the server, hosting, and network management.

Due to the scope of this position being so wide, it's not uncommon for full-stack developers to take on leadership or management roles and oversee large projects.

Mobile App developer

This is an ideal programming career for someone who has a "big picture" mentality.

Mobile App developers are responsible for creating and enhancing applications for cell phones, tablets, and other mobile devices.

If you love collaborating with others and making ideas come to life, then this is your game.

Important languages:

- Java
- Kotlin
- Objective C
- Javascript

Data Scientist /Analyst

If you're working as a data analyst or scientist , you'll find yourself standing at the intersection of information technology, statistics, and business.

The primary goal of this role is to increase efficiency and improve performance by discovering patterns in data.

Data analysts take control of these fields in order to help businesses and organisations succeed. To become a DA is to become a control freak (in a good way).

Important languages:

- Python
- SQL
- R

Database Administrator

Be the big boss as a database administrator (DBA).

A DBA manages, backs up, and ensures the availability of the data produced and consumed by today's organisations via their IT sys-tems.

DBA is a critically important role in many of today's IT departments, and by extension, their organisations overall.

Imagine having access to all of this big data, right at your finger-tips...

Important languages:

- SQL
- Python

Step 3: Start learning new languages

Once you've settled on a field, you need to learn the relevant languages that are relevant to that field.

For example, you'll need **CSS**, **HTML**, and **JavaScript** to become a front-end web developer, and Python if you want to become a data scientist.

Now, before it becomes overwhelming (and you start entertaining thoughts to stop reading beyond Step 3) remember this...

When you know how to code in one language, it'll be quicker to learn others.

Programming concepts are basically the same in every language.

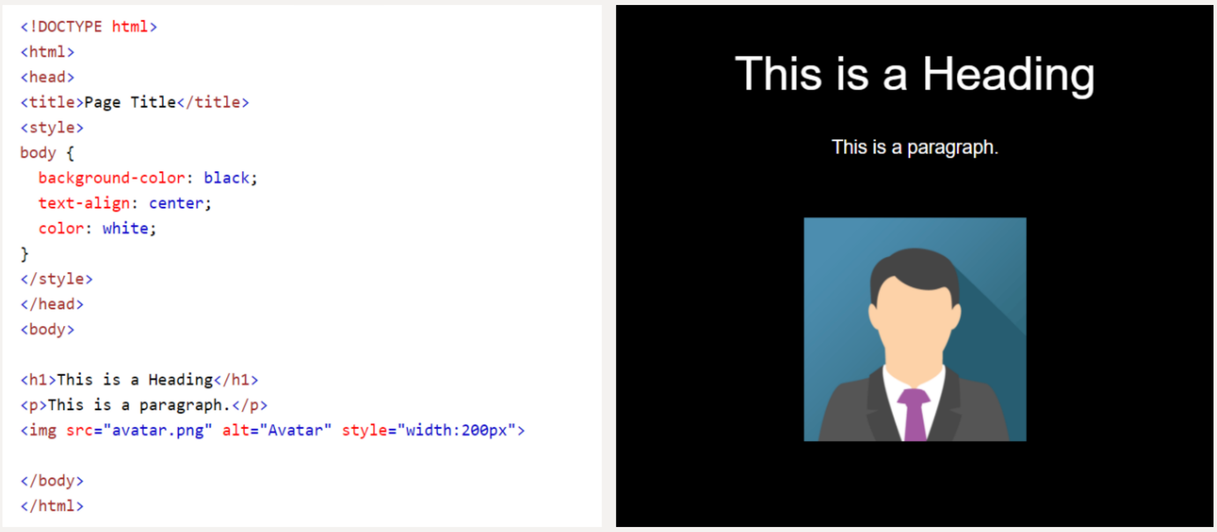
So, you can relax and read on :).

HTML & CSS

If you don’t have an idea of where to begin, **HTML** and **CSS** have always got your back.

This is how most programmers start out and it’s the most proven method that will get you into the world of coding.

It’s useful if you need to create a portfolio to show your projects, as you can create it by yourself. Additionally, it’s an easier language to start with—you can learn the fundamental bases and also create websites for your friends. You can try out this tutorial: HTML and CSS.



```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<style>
body {
  background-color: black;
  text-align: center;
  color: white;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<img src="avatar.png" alt="Avatar" style="width:200px">

</body>
</html>
```

This is a Heading

This is a paragraph.

HTML and CSS example from w3schools

Javascript

In simple terms, JavaScript is a programming language used to make websites interactive.

If you think about the basic makeup of a website, you have **HTML**, which describes and defines the basic content and structure of the website, then you have **CSS**, which tells the browser how this **HTML** content should be displayed—determining things like colour and font.

With just **HTML** and **CSS**, you have a website that looks good but doesn't actually do much.

JavaScript is what brings the website to life by adding functionality.

JavaScript is responsible for elements that the user can interact with, such as drop-down menus, modal windows, and contact forms. It's also used to create things like animations, video players, and interactive maps.

Nowadays, JavaScript is an all-purpose programming language—meaning it runs across the entire software stack.

The most popular application of JavaScript is on the client-side (aka **frontend**), but since Node.js came on the scene, many people run JavaScript on the server-side (aka **backend**) as well. When used on the client-side, JavaScript code is read, interpreted, and executed in the user's web browser. When used on the server-side, it's run on a remote computer.

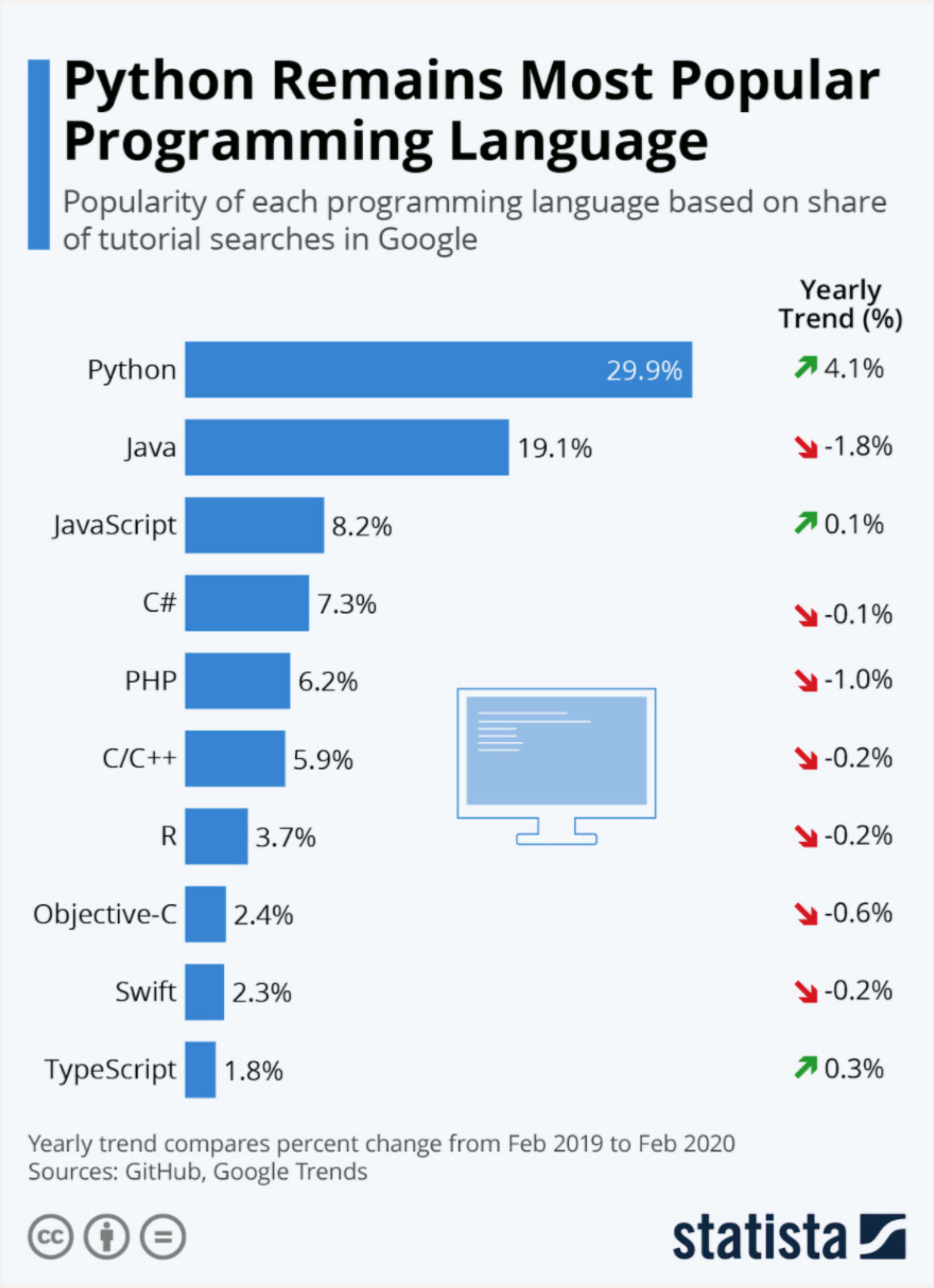
You can learn more about the difference between frontend and backend programming here.

JavaScript isn't only used to create websites. It can also be used to build browser-based games and, with the help of certain frameworks, mobile apps for different operating systems. The creation of new libraries and frameworks is also making it possible to build backend programs with JavaScript, such as web apps and server apps.

Python

Initially developed in the late 1980's by Guido Van Rossum, Python has been around for decades alongside other server-side languages like Java and C. Van Rossum modeled Python after the English language, eliminating unnecessary syntax to make it easier to read and write than other programming languages.

Python is an open-sourced language, and in recent years has increased in popularity due to its use in data science. It's also become more common to use Python for the backend of a website. Python has a strong community around machine learning, data modelling, data analysis, and artificial intelligence (AI), with extensive resources and libraries built for these purposes. You can learn more about python here.



SQL

Structured Query language (SQL), sometimes known as “See-Quel” is the standard language for dealing with Relational Databases.

SQL is effectively used to **insert, search, update, delete, modify database records** though it can do many more things too. If you want to learn more about SQL, take a look at this tutorial on [SQL Tutorial](#).

```
 * @link https://de
 *
 * @package _s
 */
if ( ! function_exists( 'incode_starter_setup' ) ) :
/**
 * Sets up theme defaults and registers support for various WordPress
 *
 * Note that this function is hooked into the after_setup_theme hook, whic
 * runs before the init hook. The init hook is too late for some features,
 * as indicating support for post thumbnails.
 */
function incode_starter_setup() {
```

Step 4: Practice—because practice, not talent, is the key to success

Unless you're born with coding skills (which is absolutely impossi-ble), the key to becoming good at coding is... **practice!**

Practice writing code from the moment you start learning.

Create as many projects as you can. Share them and ask other peo-ple to critique your code.

You can start by researching a simple project that you're interested in and try to implement it.

For example, a simple application to control financial expenses, solve mathematical formulas, or simply save your favourite movies. The important thing is **not to give up**.

But what if my code isn't working?

Don't worry, this is normal. People call programmers "problem solvers" and this is for a reason: every day they face different errors and blockers in their code and the way they face them determines their success as a programmer. It's important to persist and also talk with people about how to code.

Here are the steps to follow when your code doesn't work:

- Look for a syntax error
- Check if the files are in the correct directory
- Try Rubber Ducking

Next, ask yourself the right questions.

- Try to explain what your code was supposed to do and what happened instead?
- Check your code line by line
- Google it! Stack overflow is the best platform/forum for getting help with code bugs. Helpful Tips For Improving Your Googling Skills.

If none of these steps solves your problem, **ask for help**.

Talk with an instructor or a programmer. If you don't know any programmer or instructor, there are a lot of websites and forums with mentors prepared to help you.

Step 5: Build a portfolio of work

A programmer portfolio goes beyond the normal CV. It's a way to showcase to prospective employers that you can really do what you talk about in your CV.

Show off, but make sure you walk the talk.
If you're wondering whether it's worth all of that effort to create a programmer portfolio website, in a word: absolutely.

A well-rounded programmer portfolio is an important asset to have at your disposal. It's often a decisive factor for a prospective employer, and so can make all the difference when you're competing for a coveted role.

Plus, a portfolio website acts to showcase your previous work samples, as well as the site being an active example of what you can do!

Finally, your portfolio website can also help you craft your personal brand.

You can use it as an opportunity to go beyond the work and express your personality, highlighting those crucial soft skills that are also needed. This gives potential employers an idea as to how you might fit within the company/team, or whether you'll succeed in the role.

You can learn more about this step here: [E-guide: how to create a programmer portfolio.]

Frequently Asked Questions (FAQs)

We figured you might still have some doubts regarding these, so here you go :)

1. Is it possible to become a professional programmer, even if I start learning late?

First of all, it's never too late to learn.

But if you want to achieve a professional level, you have to put a certain amount of hard work into learning and improving yourself. **Discipline, focus, and tenacity** are essentials to learn anything, not just how to code. If you have these three main characteristics, it doesn't matter what age you are or your learning pace.

Based on stats, people from all walks of life can code despite their age, country, culture, and formation... To conclude, there's no right age nor time to start programming.

2. Is coding for me?

Before you start programming it's important to know a couple of things.

Remember: Learning programming requires more time and practice than you might expect.

- To be good at coding, you must learn how to build products, not only write code.
- To be a web developer, you must be able to create a website, not just write HTML tags
- To be a mobile developer, you must be able to build an app, not just throw objects together in Objective-C or Java.

You get the point, right?

Golden rule: **Prepare yourself** to be introduced to a new topic that won't make any sense to you and be brave enough to suck at something new—until you don't.

This scenario is going to happen over and over again. How you handle that situation, each time it happens, **is the crucial factor that will determine your success.**

Some final takeaways before you jumpstart your coding career:

1. **Never compare yourself to others.** Everybody has their own time to process new information. It’s super important to focus on the things you’re learning and also to remind yourself that it doesn’t matter about the time but more about where you’re going. If you’re practicing regularly you’ll get there, this we’re 100% sure of.

2. **Practice regularly and record your progress every day (if possible).** Share your knowledge with others as this really helps to build information and fix things. You’re going to be helping another person to learn a new thing with your own learning journey.

3. **Never ever hesitate to ask for help.** It’s normal to get stuck, it’s normal to have questions and to be insecure. You’re not alone. Ask for help from professionals, instructors, mentors. Etc.

Get the right kind of support throughout your coding journey

We can't brush off the fact that gender disparity exists in the tech space, and we know it will take support from all sides to fix it.

This is why we created CodeOp, a tech school for women+ (inclusive of trans and nonbinary) who want to transition to tech or upskill their careers.

Why CodeOp is different:

- We provide the safe space, training, and resources needed to encourage, support, and equip women+ with the skills they need to thrive in the tech industry
- We keep class sizes small, guaranteeing a ratio of 1 instructor for every 5 students to provide 1-1 attention and faster learning
- Our students enjoy a transformative learning experience in an environment that fosters collaborative learning over competitive learning
- We provide education and preparation in translating past careers

into technical careers, salary negotiation, and leadership
- Our students get access to role models and mentors who understand the journey and experience unique to underrepresented groups in tech.

Why Our Full Stack Development Course is different:

- Our rigorous Full Stack Development course is based on the strongest technical curricula in tech education
- We don't just cover the fundamentals of the tools you'll need. Unlike other bootcamp curriculums we also:
    1. Teach additional tools, techniques, libraries, and best practices that are actually used in "real life"— the type that any company who hires you will want you to use.
    2. Utilise "flash lectures", to introduce you to necessary tools that aren't often covered, ensuring you end up more prepared for building real apps from day one.
- You'll have the choice to study online, or at our fantastic campuses in Barcelona and Malaysia
- Our support extends far beyond the classroom, with tailored career advice, workshops, and focused 1-1 coaching sessions to help you get where you want to go in your tech career.

You've got the motivation, so why not get the skills? Join our free online coding class for beginners to kickstart your journey and find out what it's like to learn to code.