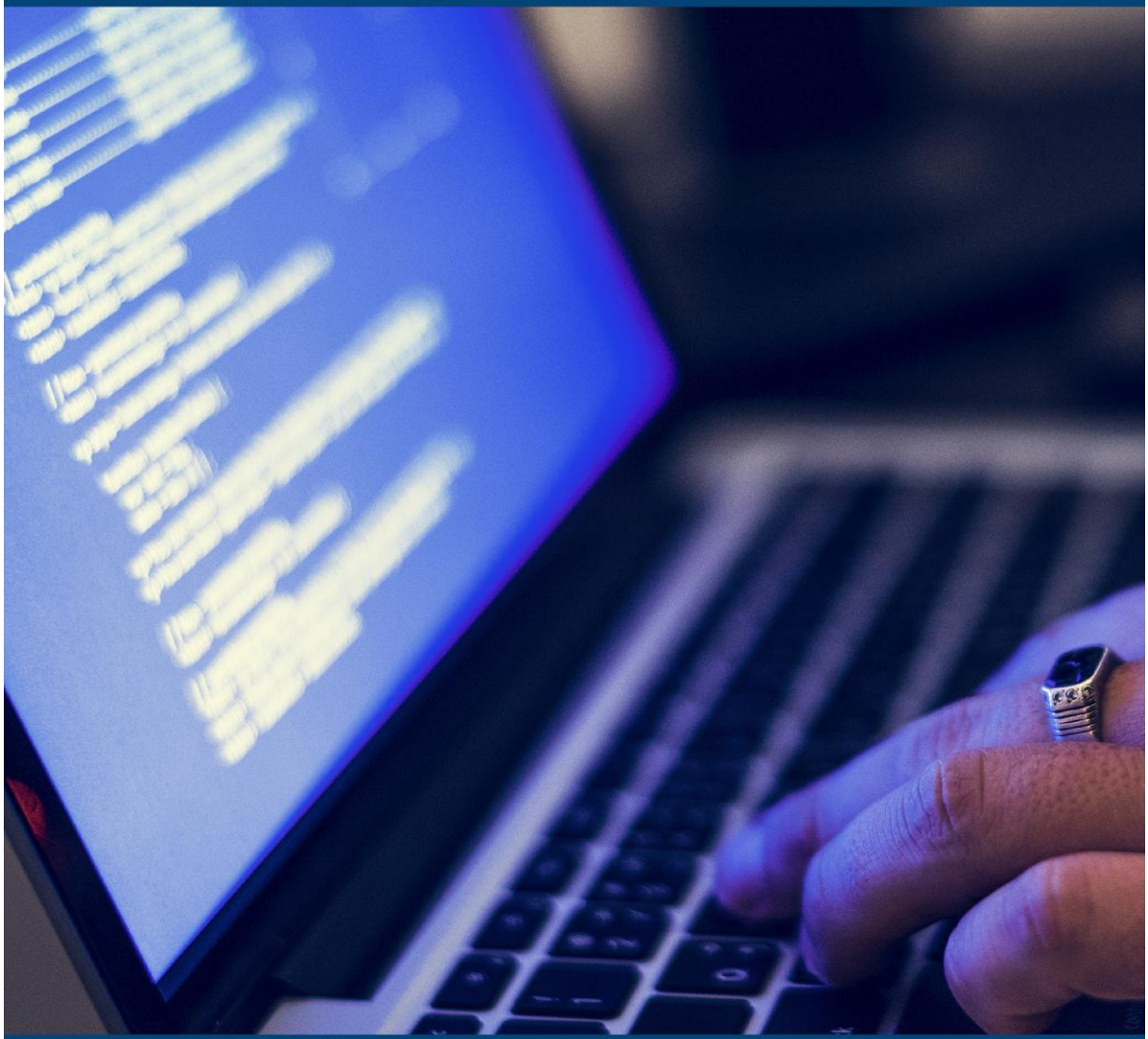
TIM BUCHALKA

PROGRAMMING CAREER GUIDE

GET ANSWERS SO YOU CAN MAKE DECISIONS!



www.learnprogrammingacademy.com

Table of Contents

Introduction

Thanks for downloading this free guide that answers some of the most pressing questions about moving into a new career as a software developer.

I started programming way back in 1980 (or "the Wild Wild West days" as my son used to tell me).

Back then I had no-one to ask for help and had to figure it all out on my own. I eventually moved into full-time software development where I worked for many companies including Mitsubishi, SAAB and Fujitsu.

These days, I almost exclusively teach people how to program through online video training, which I believe is far and away the best way to learn.

Learning to Program and Career Advice videos on Youtube.

I am now releasing regular videos on Youtube answering questions like the ones I answer in this e-book.

I suggest you click the link below which will take you to the Youtube playlist, and subscribe to my channel and click on that bell notification icon to get updates each time I release a video. I've adding about 5 new videos a week and currently have over 50 programming career related videos on the channel.

Video Playlist on Youtube

Details of my courses.

As I write this guide, I have 52 courses available. You can get an up-to-date list of my courses, along with course curriculums and even reviews from other students, at

Programming Courses

Ok so what's in this guide?

This guide answers to 35 of the most common questions people have asked me during my software development career. Having taught around 1,000,000 people how to program, chances are you'll find the answer you're looking for. But if you don't, there's a section at the end on how to get it.

Okay, that's enough of an introduction. Let's get started on answering the big questions about moving into a new career as a software developer.

1. Can programmers secure jobs with practical experience alone (e.g. from doing an online course), or do they need a computer science degree?

There are a few things to point out here.

Firstly, having a computer science (CS) degree doesn't necessarily make you a better programmer. I've met and worked with a lot of developers over the years—some with CS degrees and some without. And I never found a pattern to identify a better programmer based purely on a CS degree.

In my experience, having a CS degree doesn't guarantee you'll be a better programmer than someone without a degree.

Secondly, having been in numerous interviews as both an interviewer and interviewee, your education and qualifications bear nowhere near as much weight as the skills and experience you bring to the table. If you can demonstrate excellent skills in software development you'll have a big advantage over someone with a CS degree but no practical experience.

After all, you're being hired for what you can do for the company, not your qualifications.

That being said, if you *don't* have a CS degree and you're up against someone who has one as well as practical experience, then all else being equal they may be the preferred applicant.

There needs to be some way to separate job applicants, right?

You may even come across companies that insist on all applicants having a CS degree. But that's getting rarer and rarer these days, and with good reason. By ignoring anyone without a CS degree, they're probably missing out on a large pool of great programmers.

Bill Gates and Mark Zuckerberg are just two examples of programmers who achieved great success without a CS degree.

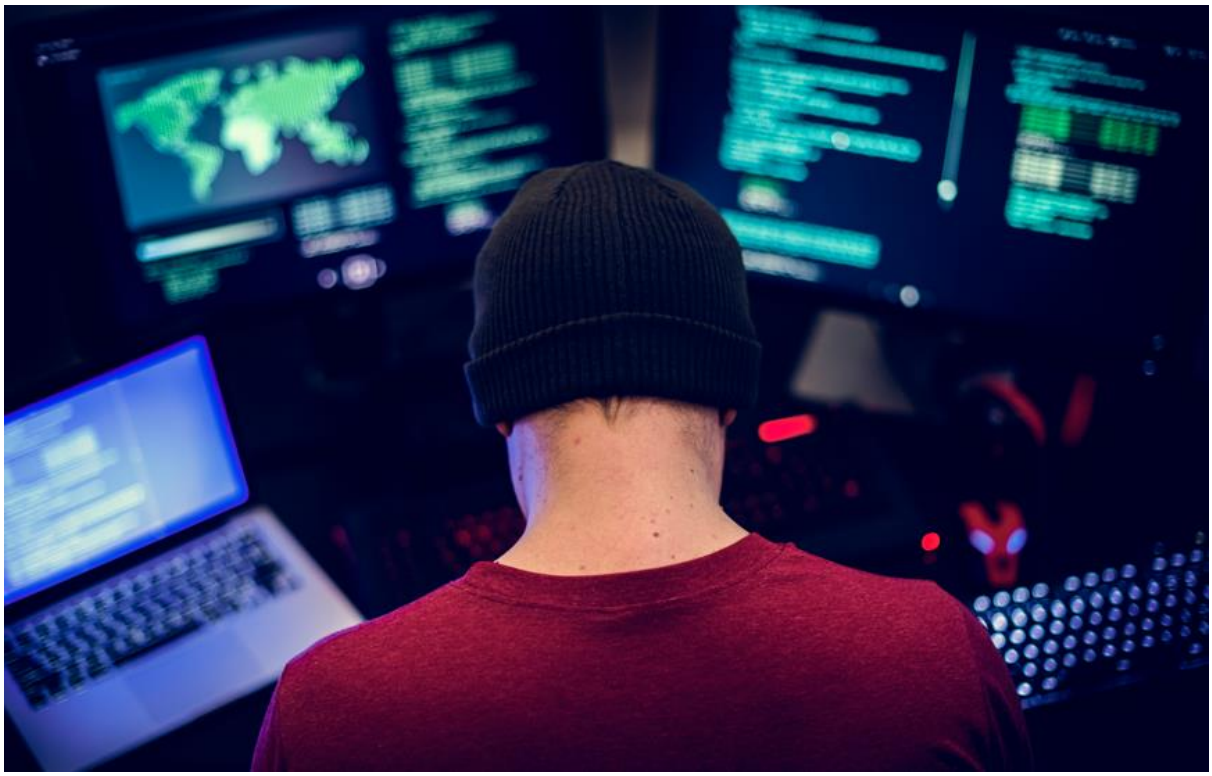
Over the past 35-odd years I've seen numerous people without degrees get jobs as software developers. In almost all cases it was the skills and practical experience they bought to the table that got them the position.

The bottom line is, if you have excellent software development techniques and can demonstrate these, it doesn't matter if you have a degree or not.

Start working on those software development skills today.

Useful related Youtube resource videos:

Do I need a degree to be a software developer video
When Can I Get a Job After Completing an Online Programming Course video
Should I Supplement My College Or University Training with Other Programming Resources video
What to Do Next After Completing a Programming Course video



2. Which programming language offers the best opportunities for career advancement and job security?

When choosing a programming language, you should consider something called *language maturity*.

Generally speaking, a newly released programming language will give you fewer job options in the short term. That's because most companies take a while (often a very long time) to commit to a programming language. And even then, it's usually only after the language has achieved a fair degree of market share.

No company wants to invest in a programming language only to discover it's no longer being supported or maintained, or that they can't find any programmers with skills in that particular language. When that happens, it can become a very expensive process to get all their programs rewritten in another programming language.

Think of it as risk aversion. Companies usually won't take significant risks on programming languages to avoid potential issues down the track.

Of course, there's no guarantee that even today's favourite programming language won't go out of favour tomorrow.

In recent years, languages such as Java, Python, C, and C++ have consistently been at the top of the popularity list.

Up-and-coming languages in 2019 that aren't as popular as the mainstream languages (at least not yet) include Rust and Kotlin.

The TIOBE index maintains a list of programming language popularity.

TIOBE Index for January 2019

January Headline: Python is TIOBE's programming language of the year 2018!

The Python programming language has won the title "programming language of the year"! Python has received this title because it has gained most ranking points in 2018 if compared to all other languages. The Python language has won 3.62%, followed by Visual Basic .NET and Java. Python has now definitely become part of the big programming languages. For almost 20 years, C, C++ and Java are consistently in the top 3, far ahead of the rest of the pack. Python is joining these 3 languages now. It is the most frequently taught first language at universities nowadays, it is number one in the statistical domain, number one in AI programming, number one in scripting and number one in writing system tests. Besides this, Python is also leading in web programming and scientific computing (just to name some other domains). In summary, Python is everywhere.

Other interesting positive moves of 2018 are MATLAB (#18 to #11), Kotlin (#39 to #31), Rust (#46 to #33), Julia (#47 to #37) and TypeScript (#167 to #49). The following languages had a hard time in 2018: Ruby (#11 to #18), Erlang (#23 to #50), F# (#40 to #64) and Alice (#26 to #66). Let's do one prediction for 2019: Kotlin will enter the top 20. We see a fast adoption in the industrial mobile app market of this language.

IMPORTANT NOTE. SQL has been added again to the TIOBE index since February 2018. The reason for this is that SQL appears to be Turing complete. As a consequence, there is no recent history for the language and thus it might seem that the SQL language is rising very fast. This is not the case.

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Jan 2019	Jan 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.904%	+2.69%
2	2		C	13.337%	+2.30%
3	4	▲	Python	8.294%	+3.62%
4	3	▼	C++	8.158%	+2.55%
5	7	▲	Visual Basic .NET	6.459%	+3.20%
6	6		JavaScript	3.302%	-0.16%

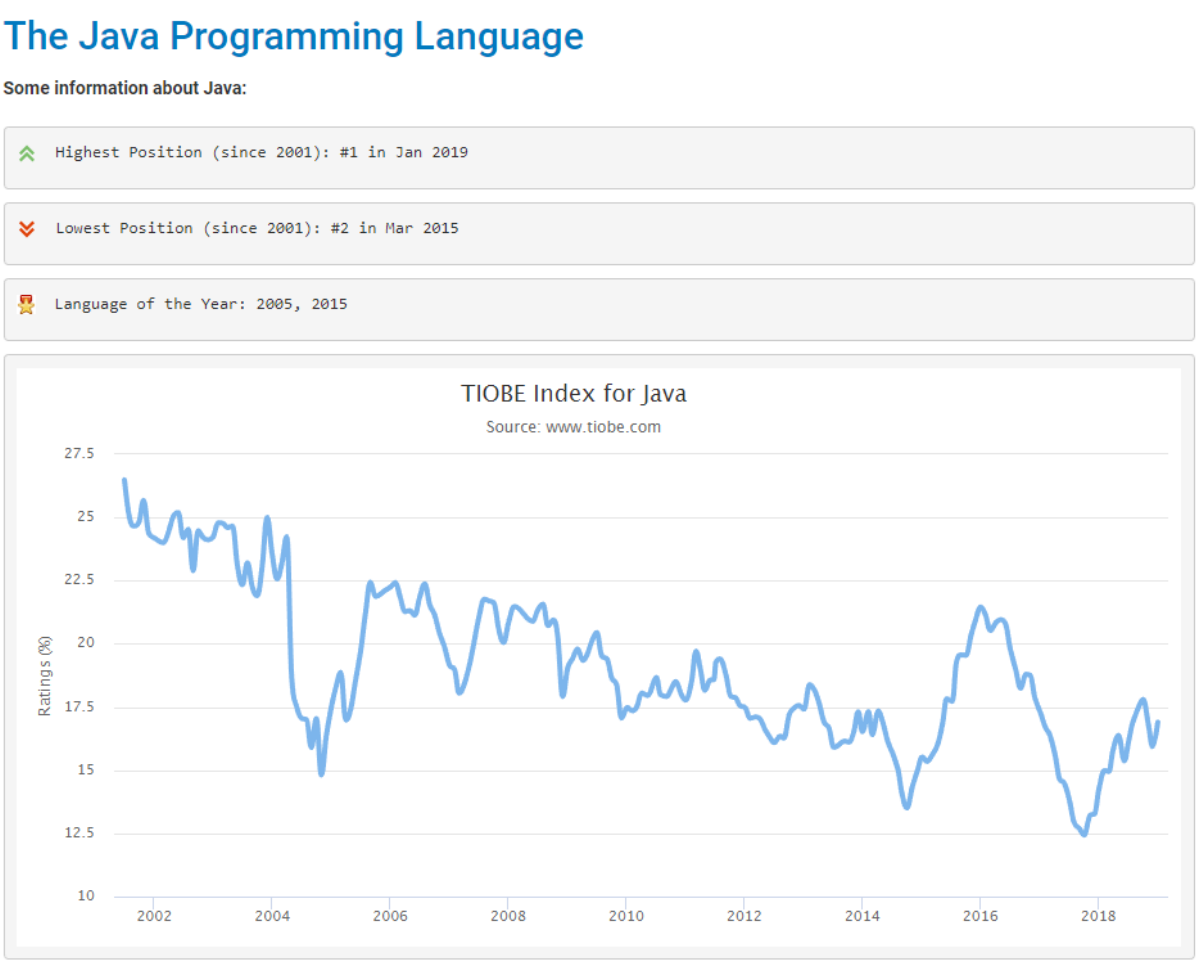
Source: http://www.tiobe.com/tiobe_index

As you can see, these languages are very popular. And with that popularity comes opportunity and jobs.

Java, C, C++ and Python are excellent choices, as these languages have been close to the top for many years. And It's unlikely they'll disappear any time soon because they're well entrenched in companies all over the world.

These languages have been around for some time. They're mature, and are being used extensively. If you were trying to decide on a particular programming language, most of these would be a safe bet

Later on, I'll talk about how to decide on a particular language to learn.



Source: http://www.tiobe.com/tiobe_index?page=Java

Useful related Youtube video resource videos:

Career Paths in Computer Science
How Long Does It Take to Become a Software Developer
The Best Skill You Need to Have to Be a Software Developer
What Programming Language to Learn to Earn As Quickly As Possible

3. Which is the best programming language to start out with?

We just talked about the languages that will give you the best job opportunities and job security. But what's the easiest language to start with if you've never programmed before?

Personally, I'd lean towards Python, Java, Ruby, and JavaScript. You may also want to consider C and C++.

Some people may disagree with those last two languages, as they're traditionally listed as being anything but easy to learn. But with the right instructor and training material, you really can start off learning either one. (The right teacher and right training material can make even the hardest language almost easy to learn.)

No matter what language you choose, I recommend going into the experience of learning knowing it will take time to master and giving yourself time to learn the foundations.

Any worthwhile skill takes time to master. And over time, concepts that seemed almost impossible to understand at first will become simple.

The best advice I can give is to be persistent. In fact, I've written an entire article on persistence and its role in software development. There is also a YouTube video about it here:

Also check out this YouTube video which talks more about the best programming language.

4. What's the best software development area to get into?

What should you focus on?

Web development?

Mobile apps (iOS or Android)?

Desktop applications for PCs and Macs?

Enterprise applications (e.g. stock market, real-time and applications alone these lines)?

Embedded applications that run on hardware to perform functions such as flying a drone, controlling pumps or traffic lights, etc?

Games?

There's no correct answer. There's only a correct answer for you.

Ideally, you want to pick something you're interested in learning. For example, if you've dabbled with setting up web pages in the past then perhaps you should look at web application development.

Maybe creating apps for smart devices such as iPhones, iPads, Android phones and tablets takes your fancy?

In 2018, mobile apps and web development are hot (i.e. in demand, which means more potential work opportunities). And Enterprise apps aren't far behind.

The mobile app market

Mobile app development has been growing year after year. In 2017 there were around 197 billion app downloads across both iOS and Android.

Source: http://www.businessofapps.com/data/app-statistics

That's a lot of devices, and a lot of users accessing those devices. But while Android has far and away the largest userbase, iOS has traditionally had the greater revenue.

Web application development

Don't confuse web application development with web design. They're two entirely different things.

Web design is the process of creating websites with images, text, etc. But web application development is about creating applications that perform functions over the internet, such as providing web services to allow controlled access to corporate information.

One example is Remember the Milk, which is a task list manager:

See: https://www.rememberthemilk.com

Another example is Quandl, which has an Application Programming Interface (API) that lets programmers access stock market information via a program:

See: https://www.quandl.com/blog/api-for-stock-data

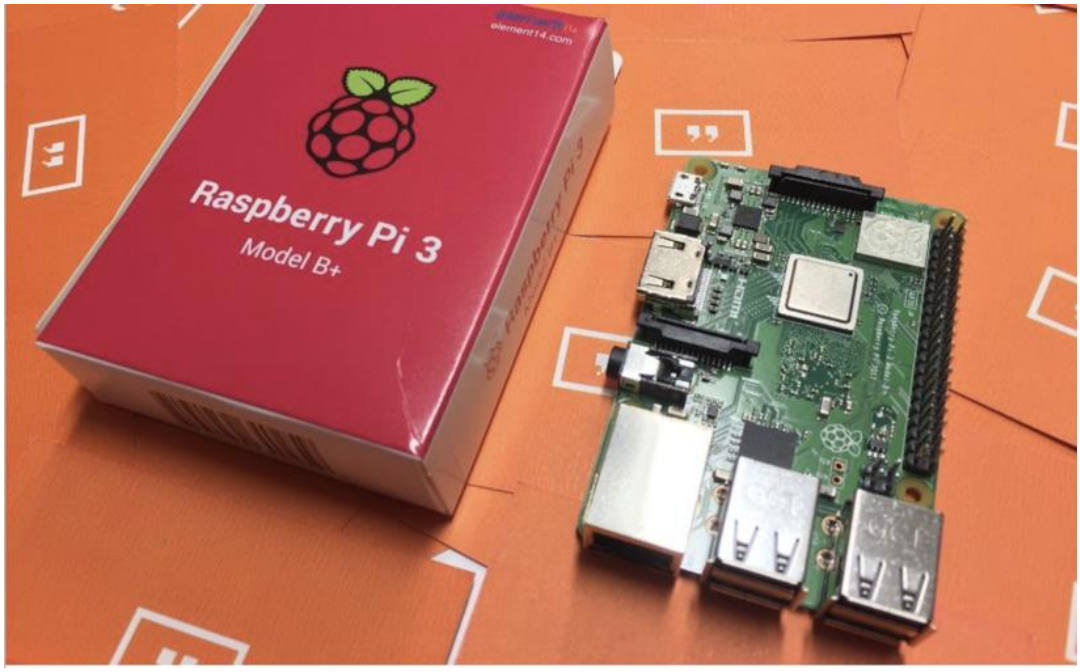
Enterprise applications

Think of these as broad versions of web applications tailored for larger companies and more complex applications. They're often real-time, mission critical applications with a team of programmers looking after them.

Embedded applications

This is another popular software development category, though probably less so than the other categories because it's such a specialised market.

This field is changing rapidly with the advent of devices such as the Raspberry Pi, and is opening up a variety of applications such as home automation, drones and weather recording.



Source: https://www.yodeck.com/news/behold-raspberry-pi-3-model-b-plus

Yes, this is an actual computer that can be used for a variety of purposes.

Other areas of embedded applications would be in devices such as refrigerators, cars, and anything else that has a microcomputer as part of its design. (It's surprising how many household items use this technology today.)

Desktop applications and games

Finally, we have:

- desktop applications for PCs and Macs.
- games for PCs, Macs, mobile devices and game consoles.

With so many options, how do you choose?

As you can see, there's a wide range of areas to choose from. But you don't need to make that choice straight away. You just need to choose a popular programming language a lot of them use.

Learn the language well and then, with that experience under your belt, start looking at a particular field to specialize in.

A good language choice here would be Java. It can be used to create desktop apps, mobile apps, embedded applications, web applications and games.

Microsoft C# is another language that's becoming popular. It's a good choice for desktop apps, games and web development. The other categories also use it, though to a lesser degree.

C++ and C are also popular languages. While they're not really suited to web application development and enterprise applications, they're often good choices for games and embedded applications. However, they're generally not the language of choice for mobile apps.

Python is a language that's well suited to many of the options here. It has an advantage over the bigger languages in that you can usually create programs with fewer lines of code. It's one of the three core languages Google uses, which is a testimony to the value it places on Python as a language. (The code that processes Google's search results is written in Python.)

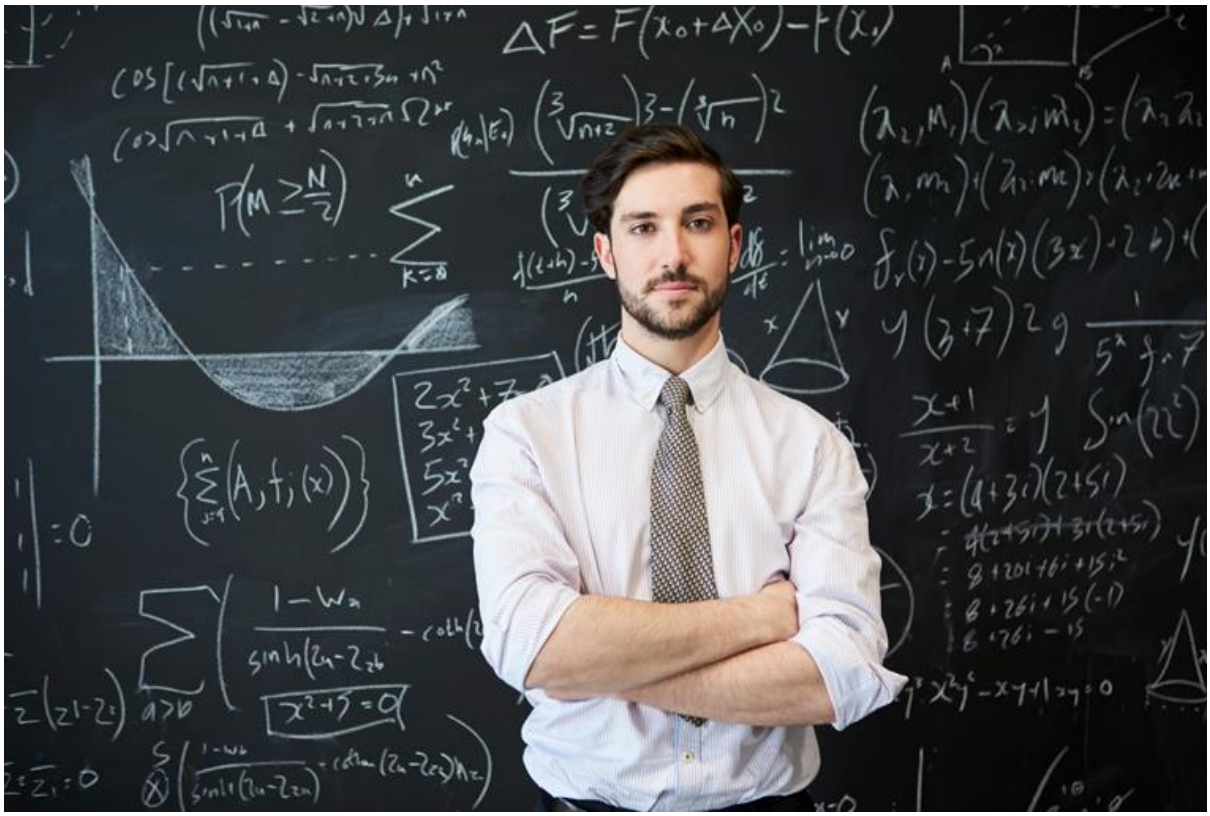
My advice? Just pick an area and a language and get started. Don't overthink it. If you try one combination and decide it's not for you, try another one. And keep trying until you find something you enjoy.

But spend some time working on each area and language before moving onto another one. Whenever you start something new, it's going to take time to get comfortable with it.

Or you could pick a "safe" language such as Java, Python, C++ or C# and learn it as you dabble in the various areas (mobile apps, web app development, etc).

Some useful Youtube video resources:

Can You Get a Programming Job With Knowledge on One Language or Framework
Should You Move Into Artificial Intelligence, Machine Learning or Big Data
Will AI Replace Programmers and Take Over Jobs

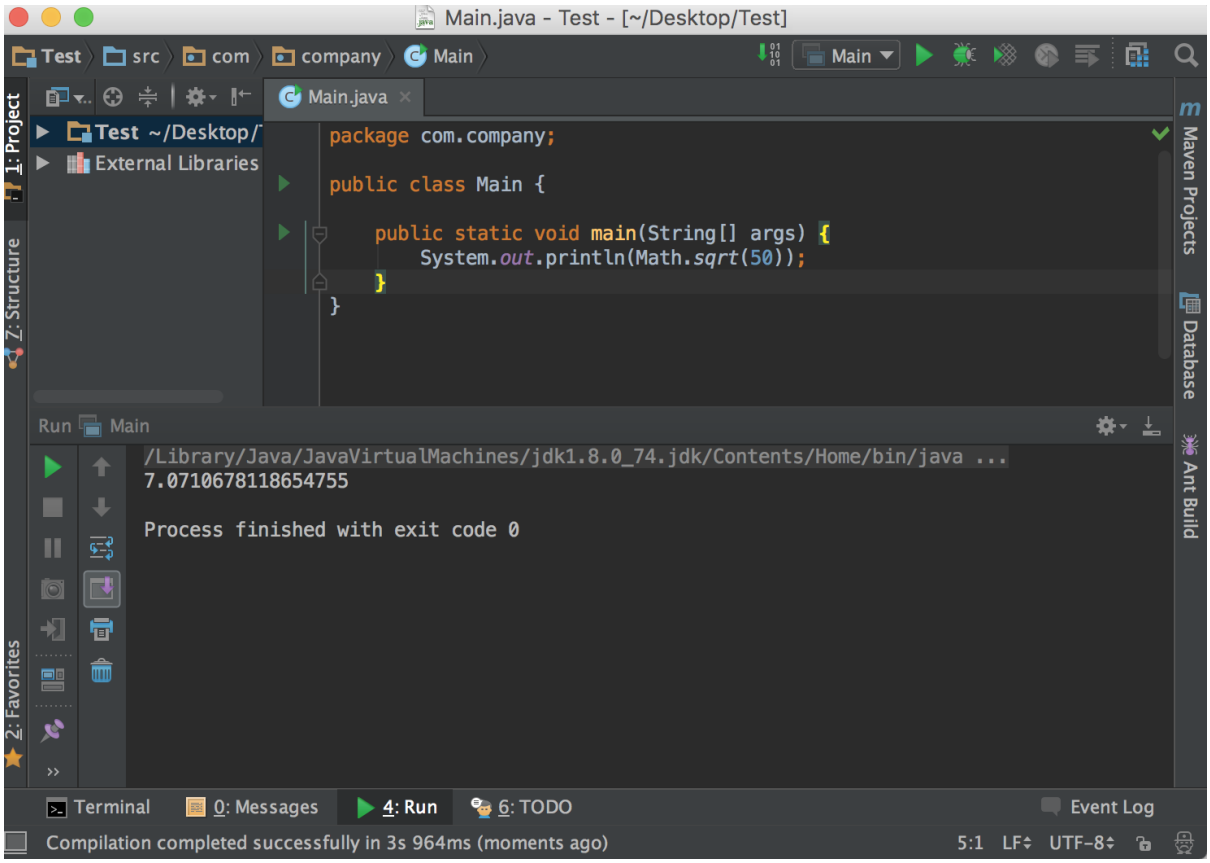


5. Do I need to be good at maths to be a programmer?

Good news! You don't need any specific math to be a programmer. Why not? Because the computer usually does all the calculations for you.

For example, do you know how to calculate the square root of a number? Well, you don't need to. You just need to know the command that tells the computer to calculate it for you.

Here's some Java code to print the square root of 50 (next page):



See the **Math.sqrt(50)** in the code and the answer (7.07106...) at the bottom of the image. The computer calculated the square root of 50 and gave you the answer.

When programming, you need to know the right instructions to give the computer, not how to perform the given calculation.

And with a good online video training course you'll quickly learn those instructions.

I've also uploaded a Youtube video on this topic.

6. Do I need to be a genius to become a programmer?

Do genius programmers exist? Yes.

Are all programmers geniuses? No.

Do you need to be a genius to become a programmer? No.

As with most career fields, you'll find people of varying skills and abilities. Some people are really driven and reach the top of their field. Others have different priorities, but still perform well.

There's always room for these and other types of people in the programming industry. Most people have similar yet different goals, and there are usually software development opportunities for all types.

The truth is, software development (particularly team-based software development) takes all types of programmers: junior programmers, senior programmers, and everyone in between.

Collectively they develop the company's software, each contributing a piece of the 'puzzle' to the finished product.

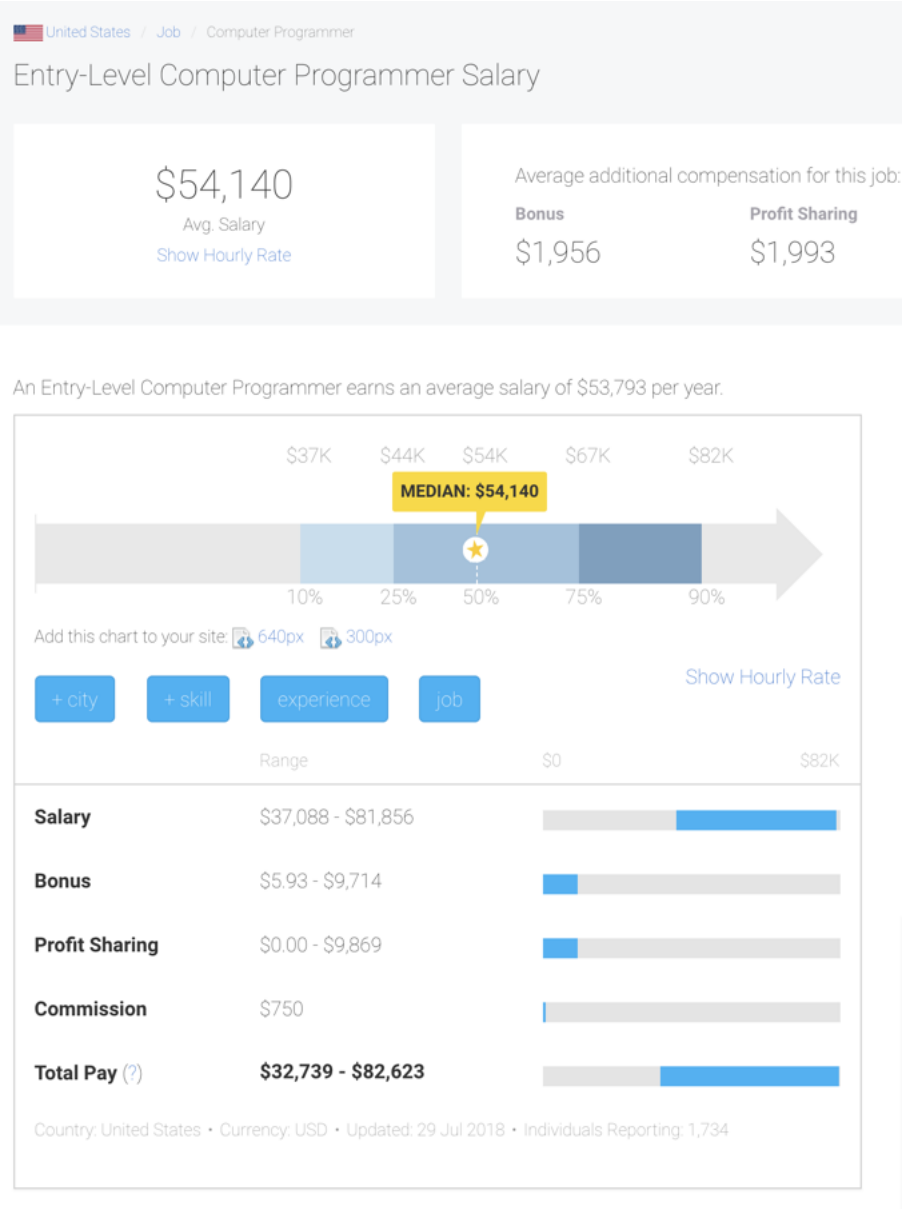
So, whether you're a genius or not, you'll find a lot of opportunities as a software developer. I've also uploaded a video about this on youtube:

7. Is programming a well-paid job? It is worth my time to become a software developer?

Yes, software development is well paid. After all, how many companies exist that don't use computers every day?

Technology is here to stay, and programmers are very much in demand. And that demand gives you the opportunity to be paid very well for your skills.

In 2018, the average salary for an entry-level programmer in the US is $54,000.



Source:
http://www.payscale.com/research/US/Job=Computer_Programmer/Salary/9fadb9f1/Entry-Level

And that's just an entry-level position. With experience, the average salary jumps to around $80,000-$90,000. And from there it can easily exceed six figures.

The average salary of a Google programmer in the UK is currently around £200,000 (US$262,000).

Source: http://www.dailymail.co.uk/news/article-5592639/Google-staff-working-UK-paid-average-200-000-year.html

I'm not suggesting you can walk into a job at that level. But the sky really is the limit. Become a skilled software developer and you'll be very well rewarded.

When I started my IT career in the '80s it was a hot niche to get into. And that demand and opportunity has been growing ever since.

Of course, technology has changed a lot since I started. Your smartphone has more power and memory than the devices used by some of the biggest companies back then. But even back then programmers were paid well.

Software development has a long history of paying well. And that's unlikely to change in the future.

8. Is it feasible to switch to a programming career and not lose my current salary level?

Firstly, it's certainly possible to change careers later in life. I've spoken to many, many people who have changed careers in their 30s, 40s and even 50s.

In fact, making the switch later in life is better in many ways. Here's why.

In our early years we're still feeling our way through life, and don't really know what we want. Life keeps giving us new experiences to deal with, and our career isn't always a priority. And for a variety of reasons we don't always make the best decisions.

When we get older (i.e. more mature) we tend to know more about ourselves and what we want. (And most importantly, what we *don't* want.) We're less likely to stay in a job or industry we hate, and often more committed to achieving the goals we set for ourselves.

In many respects, setting goals later in life gives us a much better chance of achieving them. That's why most of the people I know who changed careers later in life have achieved their goal and are happier as a result.

Don't let age be a barrier to reaching your goals. If you want to do something, your only real obstacle is making the decision to go for it.
By the way, I've also created a video discussing how to change careers.

I'm not suggesting you quit your current job today. But perhaps you could pick up a programming course and start learning outside of work hours.

Invest in an online training course on a programming language, and make a commitment to study a little every day.

Of course, having released 52 courses on programming I'm a little biased. But why not check out my online video courses and make a start?

In my experience, studying regularly—even if the sessions are only short—is far more beneficial than cramming all your study into a weekly or fortnightly session.

Investing 20–30 minutes a day (either watching a training course video or completing a course exercise) adds up to 3–4 hours of study a week. And you can get through a lot of training in that time.

Can't find 20–30 minutes a day? What about watching videos while commuting to work? You can easily watch my (and most other) online video training on a mobile device.

Not an option? How about watching one less TV program at night, or spending less time on computer games and using that time to study?

Getting the necessary skills now means you won't be starting at the bottom of the ladder, which could lead to more opportunities and bigger salaries.

And despite popular opinion, mature career switchers are offered programming jobs just like the "young guns".

Can you do it without losing your current salary level? Well, if you stick with your current job and learn programming outside of work hours, combining your other life and career skills with your new development skills will probably give you an equivalent salary. Even if you can't match it initially, your new-found motivation and drive, together with your life skills, should give you more opportunities in your new career.

Useful Youtube video resources.

Changing Jobs: How To Transition To A Software Development Career From A Different Field
Are There Long-Term Career Opportunities in Software Development
Is It Possible to Get a Freelance Online Job as Programmer

9. How long does it take to learn programming?

Unfortunately, the only real answer is, "It depends".

How much time do you have each day? How committed are you?

One thing I will say is not to hurry. Like most skills, programming takes time to learn and even more time to master.

As much as I'd love to say you can become an expert programmer in seven days, it's not going to happen.

That's not to say you won't learn a heck of a lot in seven days. What tends to happen with programming is that the more you learn, the more you realize what you don't know and need to learn.

I can guarantee you'll learn something new every day as a developer. And that's a good thing.

As I mentioned earlier, regular learning sessions will get you up to speed faster than one long study session every week or fortnight.

It's like studying for an exam. You're much better off setting yourself regular study sessions in the weeks or days before the exam than having an all-night cram session.

Leaving things until the last minute is setting yourself up for (potential) failure. What happens if our crammer gets sick, or can't study that night because of a personal emergency?

It's the same with learning a programming language. You can easily catch up if you miss one small study session. But if you miss a large study session and have to wait a week for the next one, you'll probably need to re-watch the videos because you've forgotten what you learned.

The other point I should make is that we all learn at different speeds. We're not all the same. You may be overwhelmed by certain parts of your learning journey and need to slow right down. But you may breeze through other parts because they seem incredibly easy and you feel totally in control.

And there's nothing wrong with that. It happens to almost everyone learning how to program.

If you feel a little lost, re-watching a few training videos or giving that course exercise another go can be immensely helpful.

Watching study material more than once can help you pick up things you may have missed the first time around. It's like watching a movie for the second time and finding it makes much more sense because you've noticed things you missed before.

Because you don't have a strict deadline (most online video courses are self-paced), you avoid the pressure of having to get everything finished by a certain date.

The other thing to consider is that online courses are often taught in a logical order. You need to understand each concept before you move on to the next section because it will probably assume you understood the previous concept.

And that's why it's important not to rush through the lessons. Take the time to understand what's being taught. By understanding and mastering the material as you go along, you'll become a better programmer.

Great advice Tim, but you didn't answer the question. How long does it take?

The average computer science degree is 2–4 years. But I'd be surprised if you couldn't pick up the major programming concepts within a few months and be proficient in 3–6 months.

Again, it varies from person to person. But the more time you invest in yourself, the quicker you'll be able to achieve your goal.

Youtube video resources

How Long Does It Take to Become a Software Developer
Career Paths in Computer Science
Applying for Programming Jobs You're Not Qualified For
Do I Need a Degree to Be a Software Developer

10. Am I smart enough to learn how to program? Will I understand and 'get' software development?

No-one is born a programmer. Every programmer out there had to learn how to do it.

And based on the number of people I've met in my career, I'm convinced anyone can learn how to program. Some people may pick things up faster than others, and perhaps 'get' the concepts faster. But almost anyone who commits themselves to becoming a programmer will get there in the end.

So, the question you need to answer honestly is, "Am I willing to commit myself to becoming a programmer?"

If you're thinking, "I'll give programming a try but if it doesn't work out, I'll try something else,", you're setting yourself up for failure by giving yourself permission to quit as soon as it gets a little hard.

Don't be that person.

I'm not going to sugar-coat this. You'll be frustrated at times. Things won't work out as you expect, and you'll hit a wall. You may start questioning yourself, wondering if you'll ever understand it and whether you're cut out to be a programmer.

Are you going to give up? Or are you going to dig in and persevere?

When you commit to finishing and keep moving forward, that's when the magic starts happening and things start to get easier.

Everyone faces this challenge, including my students. I even wrote an article to help them get through it:

If programming is something you want to do, then make the decision today that you're going to become a programmer—no ifs, ands or buts. Commit yourself to not give up, and to persevering, and to reaching your goal.

If you do that, there's every likelihood that you'll succeed.

Again, no-one is born a programmer. Every programmer out there had to learn to do it.

I've uploaded a video on Youtube discussing this.

And now it's your turn.

11. How do you find the best training courses to become a programmer?

To make sure you pick a training course that will help you become a great programmer, here are six things you should consider.

1. **The course instructor's credentials.**

   Sadly, people with no real professional software development experience are creating classes on the internet.

   It's vital you find a course that has an instructor with commercial programming experience. You're much more likely to be taught the right way to code, based on software development standards used by companies you may well be working for in the future.

   There are many ways to solve a problem with software development. So do you want to learn from an expert, or something who just dabbles with programming?

2. **Course updates.**

   Remember how I said technology is moving fast? Well, it's the same with programming.

When you learn a programming language, you're learning a particular version of that language. And programming languages get updated quite often.

If you don't invest in a course that's updated regularly, you may well be learning skills that are obsolete or no longer relevant to the current version of the language.

To avoid this, make sure the course is regularly updated.

3. **Reviews.**

You probably look at product reviews before you buy something online. And you should do the same before you choose an online course.

Reviews can tell us a lot about the quality of any product we buy. The sales pages for training courses are always positive, aren't they? But reviews written by students who've studied the material will give you a pretty accurate take on the course. (Just make sure the reviews really are written by people who've done the course.)

How did they find it? Were they happy with the quality? What did they love about it? What did they hate about it? Did it teach them programming?

Taking the time to read the reviews or testimonials is an excellent way to confirm the course's quality—good or bad.

4. **Access to material.**

Do you get lifetime access to the content? Or do you have to pay a monthly fee for ongoing access?

I'm not saying that having to pay a fee for continued access is necessarily a bad thing. If the course is regularly updated, paying a monthly fee for the updates is probably a good idea. But you should find out before you start investing.

5. **Support.**

At some point in your journey you'll get stuck. You'll probably have questions, and need a place to get them answered.

Many course instructors don't offer any support in their classes. They upload the materials to a website, which you get access to once you've paid. But there's no way for you to ask questions or get answers.

You may be able to figure the problems or challenges out for yourself. But sometimes it's great to get a little help.

Make sure there's a support mechanism provided, preferably by the course instructor. After all, they probably know the material better than anyone else.

Having support from the course creator should also give you confidence they'll be around for the long term, and are committed to giving you the best possible training.

It's also useful to have a forum or area where you can see questions other students have asked (and the answers to those questions), and even a way to engage with other students.

6. Trial offer.

A low-cost trial is an excellent way to assess the quality of a training course and determine whether it's right for you.

Is there a way you can try out a lesson before investing in the complete course? It's a great way to sample the course and decide if it's for you—especially if the course is very expensive.

7. Useful Youtube resources.

Programming Course to Master Software Development
Books or Video Courses to Learn Programming: Which One Is Better
What to Do Next After Completing a Programming Course

12. Are there still programming jobs on offer, or are companies outsourcing more and more?

There's no doubt that many businesses use outsourcing. Paying someone in India, the Philippines or another country about 10-15% of a programmer's salary in the US can seem very economical, and many companies have jumped at the chance.

But in many cases it hasn't worked out as well as expected. The reality is there's more to programming than just coding. There's also an element of discussion to understand the problem. And it's often much harder to explain over the internet than in person.

Outsourcers also generally need to be managed. They're often working on a component of the entire project, and so someone needs to have the full picture and understand the exact requirements of the work being completed.

The US Bureau of Labor Statistics predicts the occupation that's growing by 17% between now and 2024—much faster than the average growth of other occupations. And it's all due to the increasing demand for software.

Summary

Quick Facts: Software Developers	
2014 Median Pay	$97,990 per year $47.11 per hour
Typical Entry-Level Education	Bachelor's degree
Work Experience in a Related Occupation	None
On-the-job Training	None
Number of Jobs, 2014	1,114,000
Job Outlook, 2014-24	17% (Much faster than average)
Employment Change, 2014-24	186,600

Software developers design computer programs.

What Software Developers Do

Software developers are the creative minds behind computer programs. Some develop the applications that allow people to do specific tasks on a computer or another device. Others develop the underlying systems that run the devices or that control networks.

Work Environment

Many software developers work for firms that deal in computer systems design and related services or for software publishers.

How to Become a Software Developer

Software developers usually have a bachelor's degree in computer science and strong computer programming skills.

Pay

The median annual wage for software developers was $97,990 in May 2014.

Job Outlook

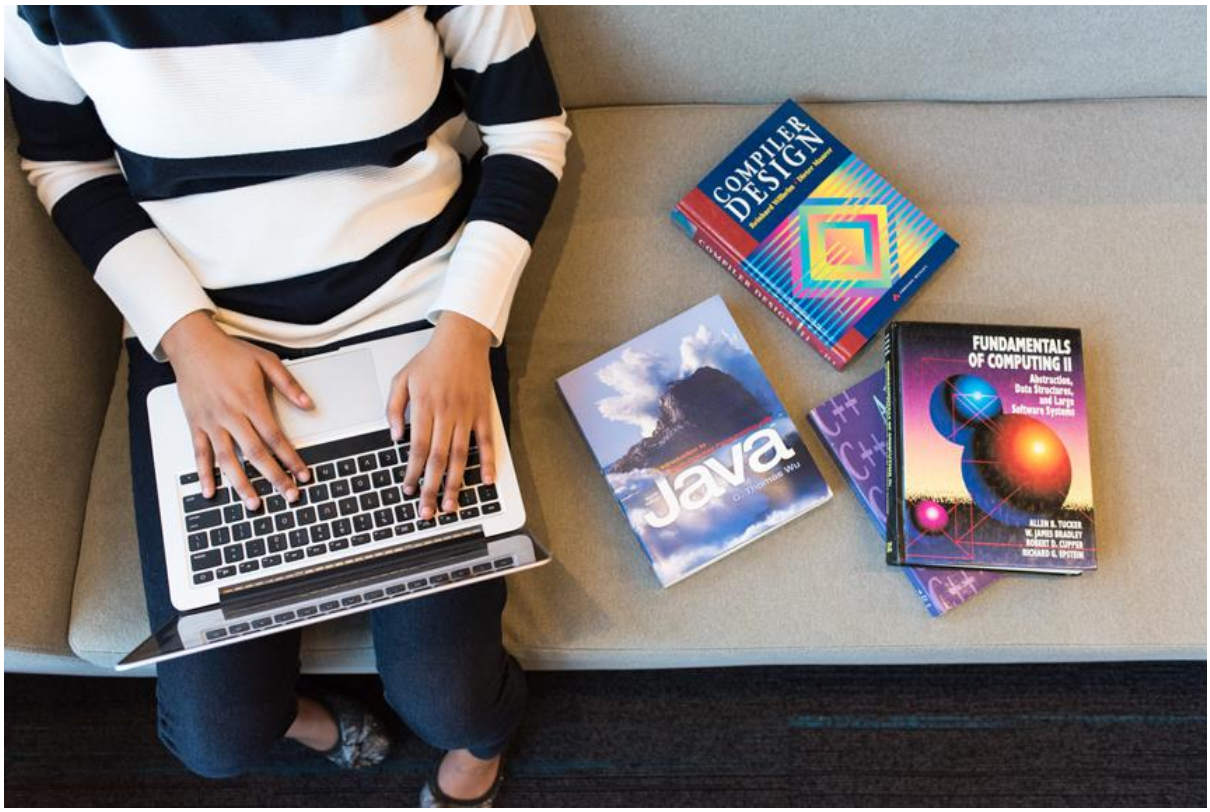
Employment of software developers is projected to grow 17 percent from 2014 to 2024, much faster than the average for all occupations. The main reason for the rapid growth is a large increase in the demand for computer software.

Source: http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

The bottom line is that on average the opportunities for programmers are increasing at a faster rate than other occupations. So local jobs are not only currently available for programmers, but will also continue to be available in the foreseeable future.

Youtube video resources.

How Do You Know If Programming Is for You
Are Programming Jobs Still in Demand
Is Java Certification Worth It?
Entry Level Job Requirements for Learners and Inexperienced Programmers.



13. Should I invest in a 'how to program' book or an online course?

My answer will obviously be a little biased because I focus on creating online video training courses. But let me give you a bit of history to explain why I believe online video courses are the way to go.

Back in the day, when I was first learning how to program, the only way to do it was to buy a book. But the selection of books was pretty limited, and so you were more or less left to your own devices.

What's more, when I was learning to program the internet didn't exist. No web browsers, no YouTube, no forums, and virtually no way to ask anyone for help.

So I was on my own. And I'll admit that I struggled.

I eventually learned to program through sheer determination. But looking back, there's really no comparison between how I learned and an online training course. Here's why.

Firstly, I find many computer programming books to be stuffy, if not downright boring. Yes, they teach you the basics of programming. But do they have to put you to sleep at the same time? I don't know about you, but I have a serious problem staying awake when reading computer programming books.

I find the medium of reading to learn how to program inefficient. It's all too easy to get distracted, making it hard to absorb the material.

I much prefer someone *showing* me how to do something, whether it's what the screen is meant to look like or how to put a program together.

Being able to follow what's shown in the video step by step is a much better learning experience. And in most cases far more enjoyable.

"You mean you can actually enjoy learning?"

Yes.

For me, there really is no comparison. An online training course will get you up to speed faster and teach you programming much better than a book ever could.

Here is a current list of my online video courses.

Youtube video resources.

Books or Video Courses to Learn Programming: Which One Is Better?
Using Books To Supplement Programming Video Course Training
How to Avoid Distractions While Studying Programming
What Courses to Study to Learn Java and Become a Java Developer
Best Way To Go Through an Online Programming Course

14. What's the best way to learn programming?

Some of the options for learning how to program include:

- bootcamps (which I'll be talking about soon)
- YouTube and other free training
- attending university
- online video courses.

There's a lot of good quality training on YouTube (and online in general) if you're prepared to hunt for it. I certainly use it when I want to find out how to do a particular 'thing'.

But it's not the ideal way to learn how to program because the tutorials are often incomplete or, worse still, created by someone who's just learning themselves. They know enough to create the tutorial, but not enough to know the technique they're showing you isn't the best way to solve a particular problem.

The people who create these free tutorials usually have the best intentions, and some of them really are good. But I've found a lot of tutorials that are bad because they're:

a) Teaching bad practice.

One of the worst things you can do is learn programming from someone who isn't a programmer themselves and/or doesn't know the right way to code a given solution (and the reasons you *wouldn't* use a particular method).

b) Incomplete.

By its very nature, free material is usually incomplete. People have spent a bit of time putting something together, but then life gets in the way and they never get around to finishing it.

c) Outdated.

You may have found the best free tutorial on the internet. But if it's out of date and doesn't work properly with the latest version of the language, it won't be of much value. And as a beginner you may not even know it's out of date until it's too late.

d) Not providing any support.

This is a big one. What happens if you can't get the tutorial to work? Who can you talk to? With free tutorials you're often left on your own to figure it out.

My advice would be to use paid material. Let's face it: we all need money to exist on this planet. Basic necessities such as food and shelter all require money in most cases.

Someone who's being paid for their work is more likely to keep it up to date and provide support. That's just the way the world works.

That doesn't mean you should buy the most expensive course on the internet. But when a course is being sold, the author is receiving an income from it and can (hopefully) keep it up to date and relevant.

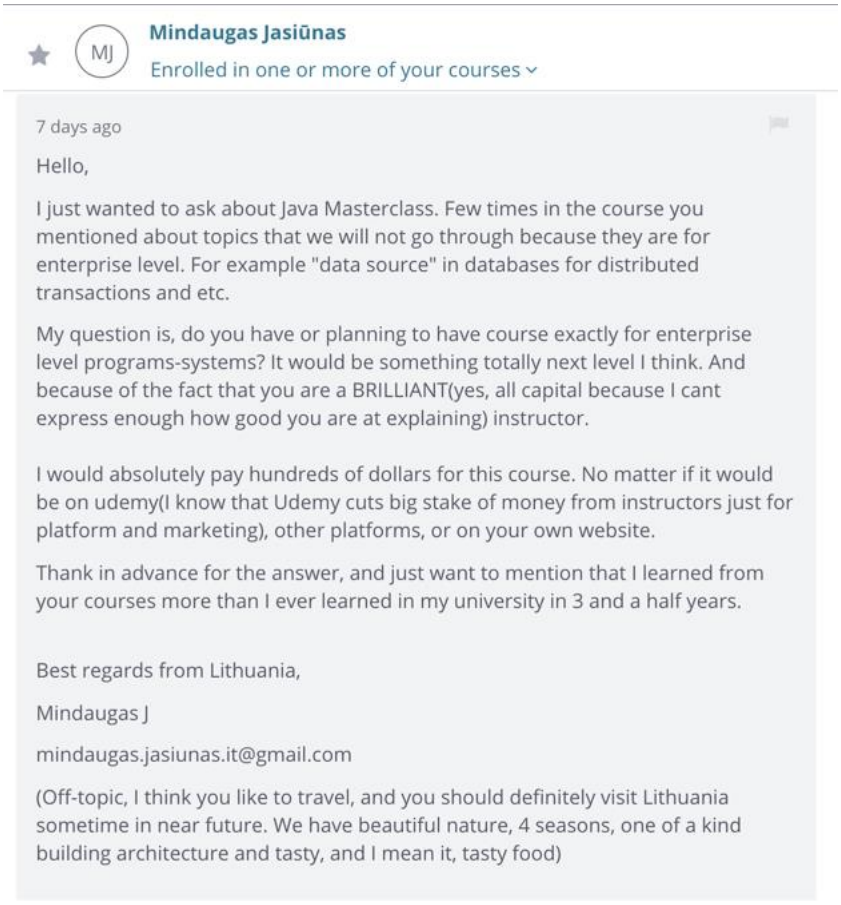
Learning to program at university certainly has its advantages, providing you have the time and money. (In many countries, including Australia and the USA, it costs a lot to go to university).

One big advantage is having the degree on your résumé. Having a university qualification looks great, and can increase your chances when competing for jobs with applicants who aren't university qualified.

But don't assume that university training is better than other forms of training. It really depends on the instructors, the training material, and how it's presented.

I get a decent number of people who are going through or have completed university courses asking for help because they felt lost with the material being presented.

Some people even tell me that my online video courses actually taught them better than university training. Here's a message I recently received from a student:



Mindaugas Jasiūnas

Enrolled in one or more of your courses

7 days ago

Hello,

I just wanted to ask about Java Masterclass. Few times in the course you mentioned about topics that we will not go through because they are for enterprise level. For example "data source" in databases for distributed transactions and etc.

My question is, do you have or planning to have course exactly for enterprise level programs-systems? It would be something totally next level I think. And because of the fact that you are a BRILLIANT(yes, all capital because I cant express enough how good you are at explaining) instructor.

I would absolutely pay hundreds of dollars for this course. No matter if it would be on udemy(I know that Udemy cuts big stake of money from instructors just for platform and marketing), other platforms, or on your own website.

Thank in advance for the answer, and just want to mention that I learned from your courses more than I ever learned in my university in 3 and a half years.

Best regards from Lithuania,

Mindaugas J

mindaugas.jasiunas.it@gmail.com

(Off-topic, I think you like to travel, and you should definitely visit Lithuania sometime in near future. We have beautiful nature, 4 seasons, one of a kind building architecture and tasty, and I mean it, tasty food)

As you can see, Mindaugas actually learned more from my Java Masterclass course than he did in 3 ½ years of university training.

I didn't include his letter to boast, but rather to show that university training isn't always the best way to learn.

Two other ways to learn programming are via online video courses and books. And if you read my answer to the previous question then you already know what I think of learning from books.

Which is why I really do believe online video training is the best way to learn to program.

Youtube video resources.

Books or Video Courses to Learn Programming: Which One Is Better
How Should You Go Through an Online Programming Course?
Goal Setting and Planning

15. Are bootcamps a great way to become a programmer?

That depends on what you are trying to achieve.

Bootcamps are generally full-on and very intense (not to mention expensive), so you need to go in pretty committed. On the other hand, students who complete bootcamps are sometimes offered jobs afterwards.

There is a huge quality gap between bootcamps. You can't assume a particular bootcamp will be good because just because of what it says on their internet page. You should practice due diligence by looking at reviews from former students.

Bootcamps usually don't cover things in depth. They teach you a broad range of skills without going into detail in a lot of areas. Depending on your goals this may be or may not be a good thing.

You really should define what your goals are before signing up to a bootcamp. If your sole intention is to become a programmer, there may be other ways to become one that are cheaper and less intense.

But if you like intense environments and putting in the hard work for a long time to increase your chances of a job offer, and have the funds to do it, then bootcamps may be worth it.

Just keep in mind there are other ways to learn programming.

16. What are the best programming books?

This is a hard question to answer. Here's why.

As I mentioned earlier, I learned a lot about programming from books when I was starting out. But that's only because there weren't a lot of options.

The internet didn't exist back then so there were no online video courses or video training of any kind. And bootcamps didn't really exist either.

University was an option, but it didn't appeal to me.

The other option was books. So, I bought a lot of books and learned to program that way.

And it was hard. Books can be so dry in the way they teach. And of course, they're very static. You can't update a book like you can an online video course. Instead, any changes that need to be made are incorporated into a new version of the book.

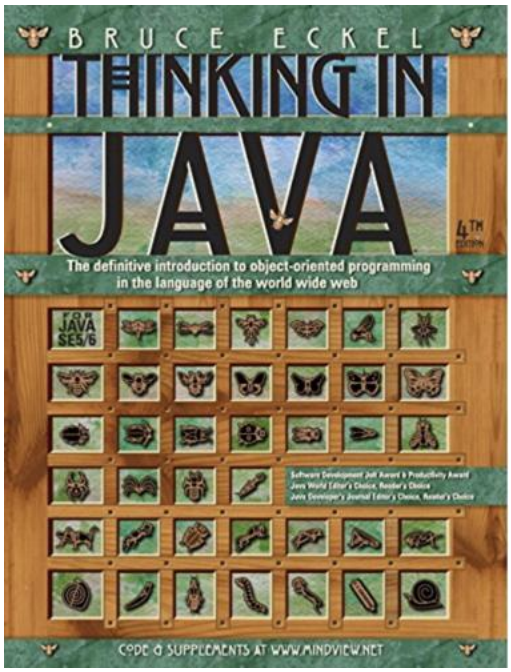
If you had no other choice, a book might be a good way to go. But now that we have online video course training, I can honestly say I would never use a book.

Online video training is visual, and usually interactive. You watch the video and follow along on your computer. And if you get stuck you can visit the course's support section where you can get help from the instructor or other students.

Thanks Tim, but you haven't answered the question. What are the best programming books?

Well, my recommendation is to not use books most of the time because you don't need to. And with my video training the material is designed to be standalone so you don't need a book or any other material.

Here's an example of a book I used to recommend.



In 2006 it was a great book that I loved, and I recommended to everyone.

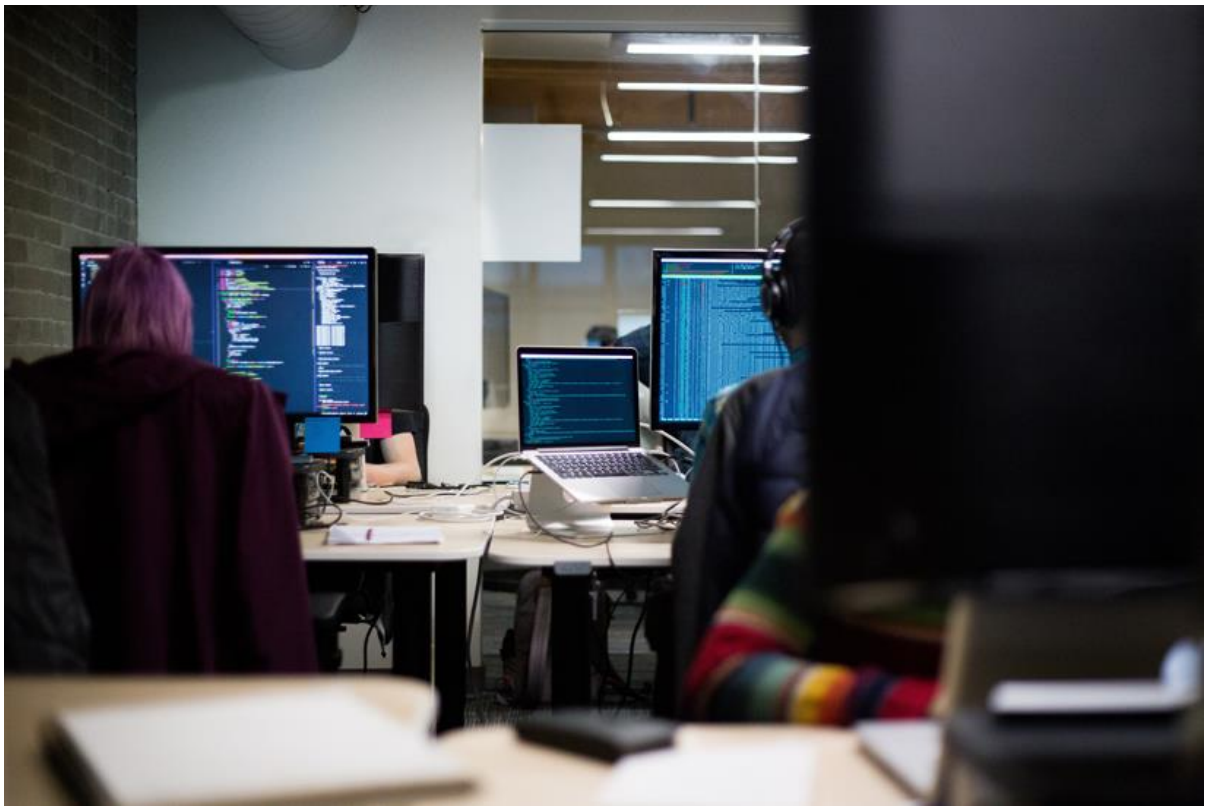
But it's 12 years old now, and totally out of date. Believe me, even a few years is an eternity in the software development industry.

Sure, some aspects of the book may still be relevant. But as the note on the Amazon page says, it's been fully updated for Java SE5—Java Standard Edition version 5. As I write this, we're at Java version 10, and version 11 will be out in a few months' time.

Universities promote the idea that you need to buy a textbook to follow along with the course. And that's probably why some students think they need a book to help them learn when taking an online course. But that's usually not the case.

Youtube video resources.

Books or Video Courses to Learn Programming: Which One Is Better?
Using Books To Supplement Programming Video Course Training.



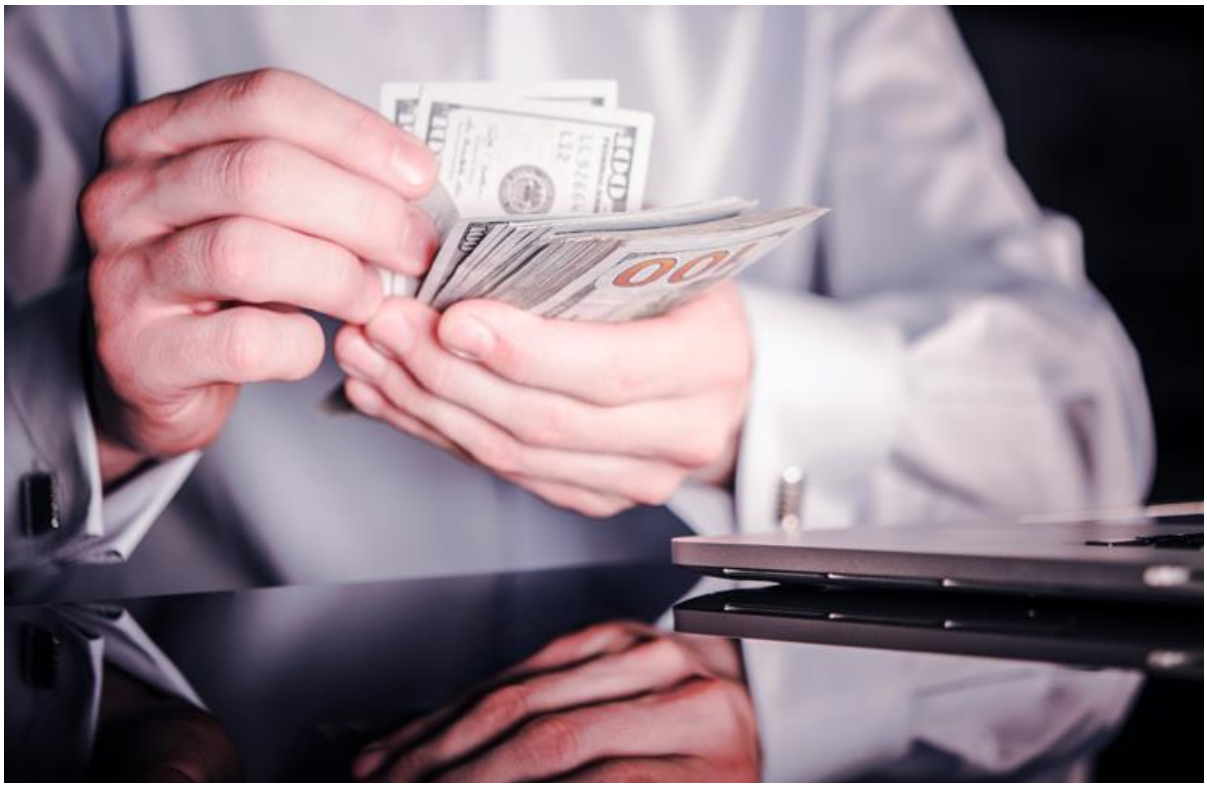
17. What is the difference between a programmer and a software engineer?

A programmer writes program code. A software engineer may also write code, but then be involved in architecting (designing) software components.

That being said, it's not uncommon for a programmer to be referred to as a software engineer. It really depends on the job.

In general, I suggest you use both terms interchangeably. But if you really want to know what a particular job entails then refer to the job description, which should outline the specifics.

This video I uploaded to Youtube can also help with the definitions mentioned above.



18. How quickly can I start earning money with programming?

This really depends on you and your ability to learn programming. Programming isn't something you're going to learn overnight. You can learn the basics pretty quickly, but becoming a good programmer takes time and practice.

A person who wants to practice medicine can't start operating on people after watching a few videos. They need to go through a process of studying, practicing, and so on.

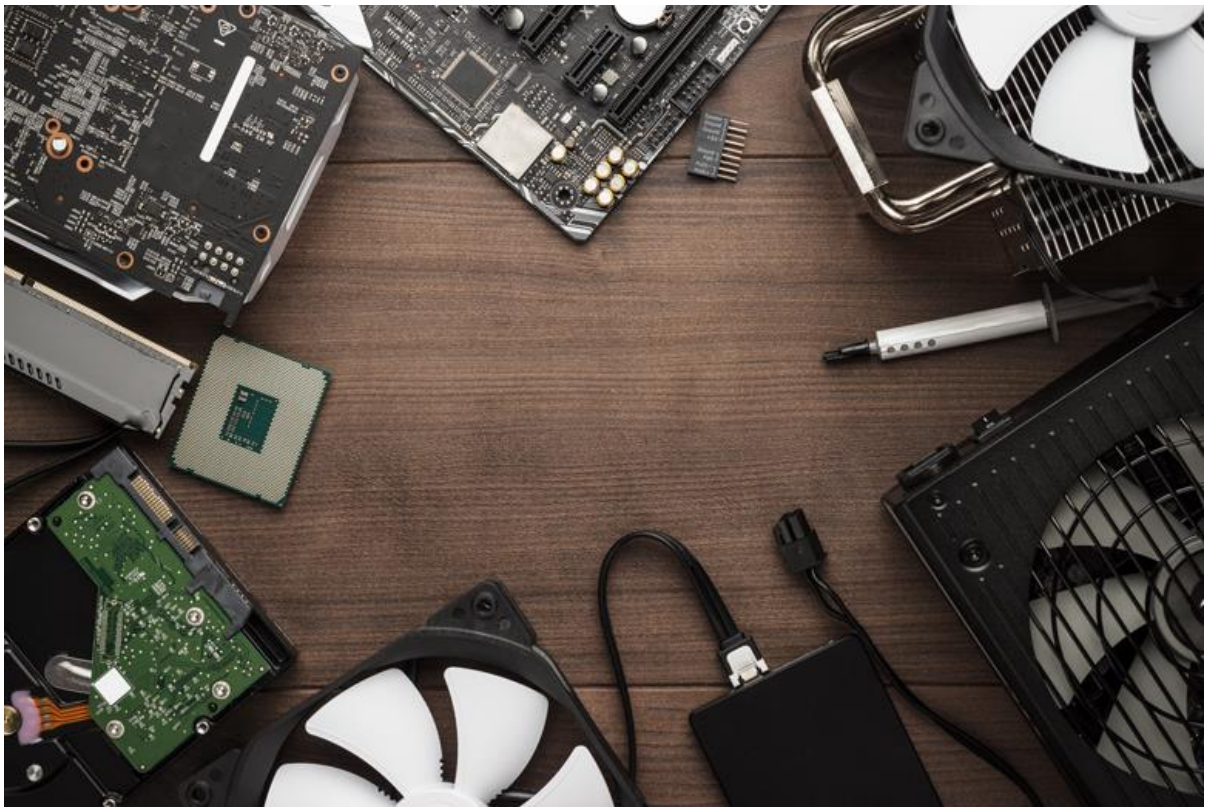
It's the same with programming, although most people could be earning money as a programmer much faster than they would as a doctor.

I've had students with no previous experience go through my training and start their first job within a few months.

Don't think of learning to program as a race. Students who rush through the training tend to miss key concepts and get tripped up later in the advanced material that assumes they know those key concepts.

Take your time to really understand what you're learning. It will make you a better programmer in the long term, which in most cases means you'll earn a better salary too.

Be sure to watch this youtube video which goes into more detail about this topic:



19. What computer equipment do I need?

To a certain degree it depends on the language. But most modern computers (i.e. released in the past 4–5 years) should be fine for software development.

Languages such as Java (and the associated programs used to create Java programs) tend to use a lot of memory, so having a decent amount of RAM is useful.

As a general rule, I'd say 8GB of RAM is the minimum, although you may get away with less.

The only other computer component you may need to choose specifically is the CPU. If you're developing Android applications, I'd recommend choosing an Intel CPU over an AMD CPU because the "emulators" (programs that run on computers to let you test Android apps) often need an Intel CPU.

That being said, there are other options. Google own Android, and has a free IDE (Integrated Development Environment) called Android Studio that you can install on your computer to help create Android apps.

This program tends to be resource hungry, so again I'd recommend 8GB of RAM for it, although you may get away with less.

You can develop on Mac, Windows, or Linux. There's rarely a preferred operating system to develop on, so stick with what you're used to.

Graphics cards, hard drives are rarely an issue unless you intend on writing graphic-intensive games, in which case you may need a more powerful GPU (video card).

If you're writing programs for iOS (iPhones or iPads) or Android (phones and tablets), it's good to have a physical device for testing. And if you're writing iOS programs, you'll probably need a Mac computer to write them in. (You can set up a Mac environment with a PC by creating what's known as a 'Hackintosh', but I wouldn't recommend it.)

Source: https://en.wikipedia.org/wiki/Hackintosh

20. What software do you need to buy?

When you're starting out, you probably don't need to buy any software to help you become a programmer. You *can*, but it's not essential. Most programming languages have a decent number of free tools you can use.

For Java, my recommendation is IntelliJ Community Edition, which is free to use. It's the tool I use and recommend in my Java and Python courses. (IntelliJ also has free plugins for working in Python and other languages.)

https://www.jetbrains.com/idea

And as I mentioned earlier, Google offers a free program called Android Studio for Android app development.

https://developer.android.com/studio

Most good online video courses will recommend a free tool, and show you how to download and use it.

21. Can I really create my own applications?

Yes, you can. And I know this for a fact because many of the students who've done my courses have sent me messages saying they've done it.

Creating applications involves a range of skills. But the more you immerse yourself in learning how to do it (which in my and other high-quality online courses), the better you get.

Don't expect your first application to be the best one out there. The most successful applications often have teams of highly skilled programmers working on them. But there's no reason why you can't release an application you've created.

Some kids are making millions with apps, and there's no reason why you can't do the same.

Here's just one example.

Source: http://www.dailymail.co.uk/news/article-4415560/Ben-Pasternak-17-making-millions-teen-focused-apps.html

22. What are the four skill stages of becoming a programmer?

I believe in the four stages of competence whenever you're learning a new skill. And programming is no different.

If you've never heard of these four stages, here's a summary.

1. **Level 1 - Unconscious Incompetence**

   The individual does not understand or know how to do something and does not necessarily recognise the deficit. They may deny the usefulness of the skill. The individual must recognize their own incompetence, and the value of the new skill, before moving on to the next stage. The length of time an individual spends in this stage depends on the strength of the stimulus to learn.

2. **Level 2 - Conscious Incompetence**

   Though the individual does not understand or know how to do something, they recognize the deficit, as well as the value of a new skill in addressing the deficit. The making of mistakes can be integral to the learning process at this stage.

3. **Level 3 - Conscious Competence**

   The individual understands or knows how to do something. However, demonstrating the skill or knowledge requires concentration. It may be broken

down into steps, and there is heavy conscious involvement in executing the new skill.

4. **Level 4 - Unconscious Competence**

   The individual has had so much practice with a skill that it has become "second nature" and can be performed easily. As a result, the skill can be performed while executing another task. The individual may be able to teach it to others, depending upon how and when it was learned.

Source: https://en.wikipedia.org/wiki/Four_stages_of_competence

When you start out as a programmer, you're usually at level two rather than level one because you've already decided programming is a valuable skill to learn. You just don't know how to do it yet.

But once you get started in one of my online training courses, it won't be long before you reach level three.

You'll be able to follow along and mimic what's on the screen on your computer. Yes, you'll make some mistakes, but that's a natural part of learning.

As you work through the course, you'll make slow but steady progress.

There will be times when the material you're learning seems overwhelming and hard to understand. But if you persevere and battle on, it will become easier and easier.

A common challenge students find at this level is that while it's relatively straightforward to understand how to do something when being shown, it's harder to do when developing something yourself.

But again, this is a normal part of learning a new skill. Keep working and persevering, and don't be too stressed about getting to level four. It will happen in its own time.

Eventually you'll reach level four – unconscious competence – and be able to apply your skills without even thinking about it.

This stage is like driving a car or riding a bike. Once you've mastered it, you don't need to think about things like changing gears and turning. You simply decide where you want to go, and your hands and feet know what to do.

Of course, if you get into another vehicle or hop onto another bike you may need to re-familiarise yourself with it.

In programming terms, this would be knowing how to program. If you need to create a new method or class, you'll know how to do it. But you won't know everything there is to know about programming. But you'll have the skills you need to research and find the information you need.

That's what professional programmers do. They're good at programming, and good at research. So if they need to do something they haven't done before, they have the skills to find the information they need and incorporate it into their program.

You may be surprised to hear that even the professional programmers need to look up how to do things sometimes. But that's how programming works. No-one in the world knows everything about programming.

Researching is part of the job, which is why I show you how to use the help documentation and research things in general. It's a skill you need to master to become a great programmer.

23. Won't programmers become obsolete when artificial intelligence kicks in?

Leaps in artificial intelligence are undoubtedly making some jobs redundant. But there's no Skynet yet. Computers can't think for themselves, and it will be quite a while before they can (if ever).

Humans create the artificial intelligence programs that give computers the capability of making "decisions" that in turn help the program to "learn".

But the computer isn't thinking for itself like a human. It's merely following instructions created by a human.

If programmers ever *do* become obsolete, it won't be for some time. If anything, non-programming jobs are much more at risk of being lost to automation and artificial intelligence.

Youtube video resources.

Will AI Replace Programmers and Take Over Jobs
Should You Move Into Artificial Intelligence, Machine Learning or Big Data
Fear of Coding and How to Overcome It
Can You Get a Programming Job With Knowledge on One Language or Framework.

24. Should you learn more than one programming language?

I recommend you learn one language well, and then look at learning another.

It's becoming more and more essential skill to know multiple programming languages. The good news is that once you've learned one programming language, learning others is usually a lot easier.

When I started out in the 1980s, the dominant languages were COBOL and Fortran in businesses and BASIC at home.

In those days you could work with a single language all your working life, and many programmers did with COBOL and Fortran.

That was partly due to the computer hardware at that time, which was very basic compared to modern computers. Today's smartphones have more power than the large computers companies used back then to service tens or even hundreds of users.

Having more powerful computers, smartphones and game consoles (not to mention the computer technology in many of today's appliances) has brought with it a greater choice of programming languages.

There's no one dominant programming language that serves all purposes.

Companies that used a single language in the past most likely use several languages now.

For example, Google uses a wide range of programming languages including Java, C++ and Python.

Source: https://www.quora.com/Which-programming-languages-does-Google-use-internally

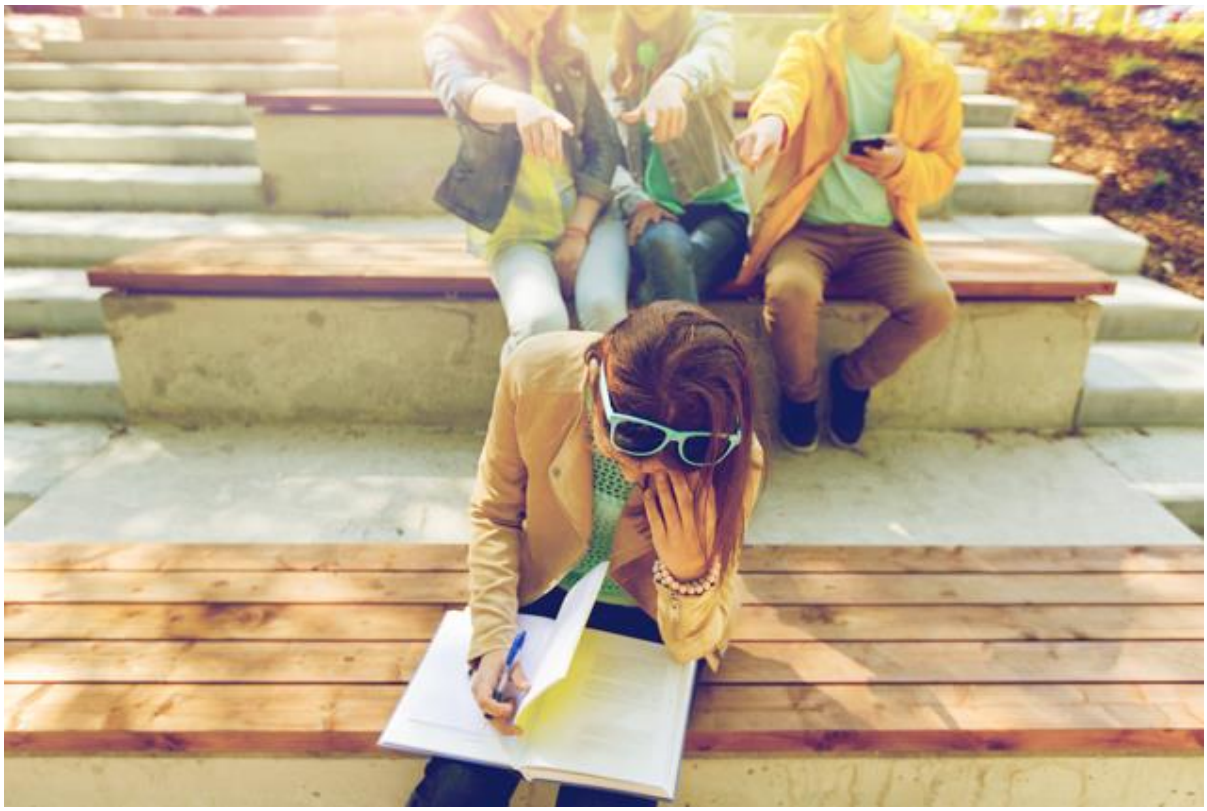
So yes, I recommend learning more than one language. It will give you more career options.

But don't fall into the trap of learning a little about a lot of languages. Learn your first language well (Java or Python are two good languages to start out with) and then once you feel comfortable look at a second language.

This approach makes sense if you think about. People who can speak several languages learned their native language first as a child, and then mastered new languages one by one.

Youtube video resources.

What Is the Best Programming Language?
Learning Multiple Programming Languages – Is It Possible?
The Best Skill You Need to Have to Be a Software Developer.
Java or Kotlin for Android Development – Which One Is Better?
C# vs Java: Which One Is Better?



25. Do only nerds "fit in" when working as a programmer?

Society tends to (or at least used to) see programmers as someone like this. (No offence to this guy.)

Yes, I've seen a lot of people like this who are great programmers. But these days you'll find a pretty diverse range of people working as computer programmers.

You certainly don't need to be a nerd to fit in. You just need to be a computer programmer.

26. What opportunities are there for women as programmers today?

Let me start by pointing out that women actually pioneered computers as we know them.

Jean Jennings Bartik, Betty Snyder and Grace Hopper are just three women who are famous for contributing greatly to the development of computers:

Source: https://en.wikipedia.org/wiki/ENIAC
Source: https://en.wikipedia.org/wiki/Grace_Hopper

In 2018 there are more and more opportunities for women. I won't say there's a 50/50 split of men and women in programming jobs, because unfortunately that's not the case at the moment.

But the landscape *is* changing. And companies are starting to make better decisions on choosing the right person for the job.

For example, back in 2005 Intel pledged $300 million dollars for a more diverse workforce:

Source: https://www.cnet.com/news/intel-pledges-300m-to-build-a-more-diverse-work-force
So while the problem still exists to a degree, it seems to be reducing.

Women make up just under 31% of Google's workforce, which is up a little from the previous year.

Source: http://fortune.com/2018/06/15/google-diversity-report-2018

In my experience, more and more companies are making smart employment decisions. And I hope these numbers continue to improve over the coming years.

There are plenty of opportunities for women programmers in 2018. And the situation will only get better as time moves on.

27. Are online video course instructors just failed programmers?

How does that saying go?

"Those who can, do. Those who can't, teach." — George Bernard Shaw

Source: https://www.quora.com/Is-there-any-truth-to-the-phrase-Those-who-cant-do-teach-If-not-where-did-it-come-from-Do-people-still-believe-in-this-or-are-there-examples-in-common-thought-that-go-against-this-idea

But while there may be some truth to this, it's a massive generalisation.

Are there failed programmers out there teaching programming? No doubt. University lecturers, online course creators, even programming book authors.

But I've also met some great programmers who found teaching their true calling. Some people are simply in the business of helping people. They enjoy doing it, and so have make a conscious decision to leave programming and to teach.

Still, you should check the credentials of any instructor you come across to ensure they have the skills you need and can trust.

Another thing to watch out for is people who have never programmed professionally before or been a programmer teaching.
This happens more than you think. In my opinion, if someone hasn't worked commercially as a programmer they have no place teaching it. There's only so much

theory you can learn. To be able to teach others, you really need to have working in the trenches as a programmer for a company in a professional capacity.

If you find someone who's been a professional programmer, their training material will almost certainly focus on teaching you the right way to program.

In todays "make money online" world, people who've never programmed before are creating online programming courses and claiming to be experts. So, you need to be careful.

Once good sign to look for is their website or portfolio of courses. If they have a range of courses (e.g. programming, cooking, digital marketing), they're probably either a publisher (such as Udemy) or not a dedicated programming teacher.

You should only buy from instructors who specialise in programming. It's okay to have a range of programming related courses, but they should all be related to programming. I have 37 online video courses, and they're all related to programming—computer languages, mobile app development, enterprise development, and so on.

You want whoever you're trusting with your education to be totally focused on programming and nothing else that's unrelated.

Another point I should make is that not everyone can teach. You may find someone who's a brilliant programmer but a very poor teacher.

Knowledge of a topic isn't enough to be able to teach it effectively. Teaching is a skill that needs to be learned and refined, and not everyone has the ability or the patience to do it well.

Sometimes the only way to know for sure whether your chosen instructor can teach and resonates with you is to watch some of their material first. Many websites will offer you a preview of their teaching material, so check them out.

28. Is it financially worthwhile to become a programmer?

Like industries if you're a good programmer then you're usually paid very well.

Expect a salary based on the value you provide to your employer. Fortunately, entry-level programming positions are usually paid pretty well compared to entry level positions in other industries.

But if your sole reason for considering a career in programming is the money, I suggest you stop and ask whether it's really for you.

In my experience, people who go into industries purely for the money often struggle getting high-paying jobs because they don't love what they do. And they tend to do only the bare minimum rather than doing everything they can to improve their skills.

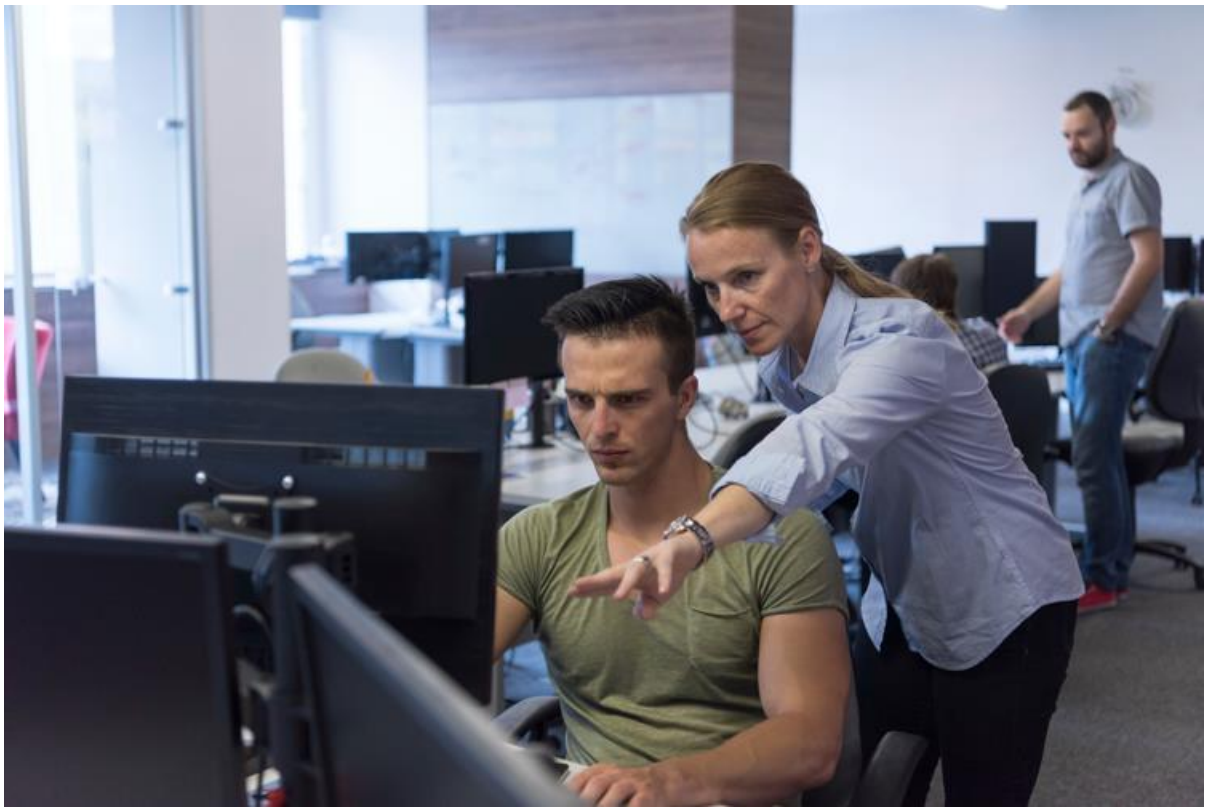
People who love what they do also tend to work on improving themselves outside of work hours, which improves both their skills and their lives.

When I started out as a programmer, I spent long hours working on it. Even when I got my first programmers' job, I still messed around outside work hours with programming. Why? Because I loved doing what I was doing.

I'm not saying you need to love what you're doing or spend every waking moment immersed in programming to build a successful career as a programmer.

But loving what you do will certainly help, and make things easier for you in the long run.

In other words, for the right people it *is* financially worthwhile to become a programmer.



29. Do I need a mentor to become a programmer?

Strictly speaking, no. I learned how to program without a mentor, and I'm sure others have too.

However, the right mentor can help you by giving you key insights and shortcuts to help you reach your goals. They can also draw on their industry experience and teach you the right way to go about programming.

One-on-one mentorship can be hard to find. There's no point working with a mentor unless they're really good at what they do. If they just know more about programming than you do, it may not be the best approach.

If you're considering having a mentor, make sure they're suitably qualified to be a mentor and have the right skills and industry experience.

Technically, anyone can say they're a mentor. But just knowing about programming isn't enough. They also need the skills to be able to move you forward.

The next best thing to finding a one-on-one mentor is to find an instructor you resonate with and take their training courses. In some cases, you can get indirect access to them through the support forum for their courses.

Don't expect one-on-one mentoring in this situation. Buying a course doesn't mean the instructor is available for questions outside the course content.

This is particularly relevant on the Udemy platform where the courses are so inexpensive. I'm not saying you *can't* contact the instructor for help outside the general support of the course. But if they don't have the time to offer one-on-one training you should respect their decision.

In my opinion, a good online programming video course with good support is what you need most to become a programmer. That, and hard work and persistence.

30. I tried out programming, but I'm stuck. Is programming really for me?

Here's a great article on this topic.

https://LearnProgrammingAcademy.com/programming/programming-is-just-too-hard-for-me

I want you to know this is a common challenge that pretty much anyone learning to program experiences at one time or another.

Not only that, it's something people experience while learning other skills as well.

Once you realise it's just part of learning you can take some deep breaths whenever you get stuck. Leave the keyword for a few hours (or perhaps until the next day) and go back to where you're stuck.

I've done the same thing countless times, and each time whatever I was stuck on suddenly made sense.

But if you quit the moment you feel stuck, it will never make sense.

Persistence is important. Keep at it, and with the right training you'll succeed and overcome any hurdles that pop up.

Youtube video resources.

The Most Important Skill You Need to Have to Be a Software Developer.
How Do You Know If Programming Is for You?
Do You Need To Be Smart To Become A Software Developer?
Am I Too Old to Learn Coding?
When Can I Get a Job After Completing an Online Programming Course?
Can Shortcuts Increase Your Learning Time As A Programmer?

31. How many hours should I study/practice each day?

My advice would be as much as possible, providing you're feeling alert and motivated.

When they're learning a new skill, everyone reaches a point when they've had enough for the day. That may be tired, or a little frustrated because something isn't making sense. There may even be a distraction that's making it difficult to study or practice.

If you reach a point where you're not taking in the concepts you're working with, or your thoughts aren't on the study material, it may be time to shut down the computer and do something else.

Take a nice walk, do another activity, go and watch a movie, visit friends, or get outside in the sun.

Or simply leave it until the next day. Sometimes I find that despite my best intentions I'm just not in the mood for programming.

So, I leave it and come back later, because when I *am* in the mood I tend to get much better results.

This is very much an individual thing. We're all different, and what works for one person may not work for another.

Try and find the balance that works for you so you can get into a groove.

32. Why should I invest in my future programming career?

If you make the commitment to become a programmer, once you have the skills it will pay you a handsome dividend with work as a programmer, or perhaps a programming consulting for a client).

In other words, the programming skills you learn can actually make you money.

So, it stands to reason that investing in your career by paying for training material and making time to go through the material is like depositing money into a bank account (assuming it pays interest).

Just as putting money in a bank account or buying shares can make money for you, buying a course and studying the material can give you similar results.

Every now and then I get a message from someone asking for free access to a course, saying they'll pay for the course once they secure a job.

But that's not how investing works. The bank doesn't pay you interest before you put money in the bank. And you can't make money on the stock market until you buy some shares.

It's how life works. First you buy the item, *then* you reap the benefits.

But always remember that investing in yourself will pay big dividends in the long run.

33. Am I too old to learn how to program?

Absolutely not!

Just yesterday I got some valuable feedback from an 84-year-old man who's done a number of my courses. He loves programming because it keeps his mind active.

When I was about 18 (early 1980s) I met a man in his late sixties who absolutely loved programming. It was a passion he found after retiring. He wished he'd discovered it earlier, but was still glad to have found it and was working hard to learn it.

So it's never too late to start. I've chatted with people who started programming in their 50s and got their first programming job as a result.

Don't let age hold you back. And check out question eight, which talks about switching careers when you have worked in one industry for a while and want to change.

This Youtube video confirms you are never too old to learn how to program:

34. Am I too young to learn how to program?

Again, absolutely not. I've got eight-year-olds in some of my courses. If you interested in programming, then get started. And if you have a child who's interested, consider giving them access to the right materials to learn.

Scratch is a great language for children to get started with.

https://scratch.mit.edu

And here's a story about a ten-year-old who creates apps and has already met Tim Cook (CEO of Apple).

https://www.smh.com.au/technology/apple-wwdc-10yearold-app-developer-from-melbourne-meets-tim-cook-20170605-gwkejp.html

35. How do I get started?

If you're serious about getting started, I suggest you invest in yourself by taking one of the 52 courses I have available.

Here's a full list of my courses on Udemy:

https://LearnProgrammingAcademy.com

For just $10 (US) you can have lifetime access to a comprehensive course that will teach you how to become a programmer.

Here are my top four recommendations for programming language courses.

- Java course - https://LearnProgrammingAcademy.com/courses/complete-java-masterclass
- Python course - https://LearnProgrammingAcademy.com/courses/complete-python-masterclass
- C++ course - https://LearnProgrammingAcademy.com/courses/beginning-c++-programming-from-beginner-to-beyond
- C course - https://LearnProgrammingAcademy.com/courses/c-programming-for-beginners-master-the-c-language

If mobile app development takes your fancy, take a look at these.

- Android Kotlin course - https://LearnProgrammingAcademy.com/courses/android-kotlin-development-masterclass-using-android-oreo

- Android Java course - https://LearnProgrammingAcademy.com/courses/android-java-masterclass-become-an-app-developer

- Xamarin (Android and iOS) course - https://LearnProgrammingAcademy.com/courses/the-complete-xamarin-developer-course-ios-and-android

Any one of these courses will teach you valuable software development skills, and you'll come away with skills you can take into your next job.

Having personally taught around 1,000,000 students, I regularly receive messages like this from students taking one of my courses:

Life-changing stuff.

As I said, I've taught around 1,000,000 people how to program. And I can teach you too. So why not get started today?

I've made it as easy as I can for you. The price is a no-brainer, and it's created for beginners—no previous experience required.

You will get instant access to the videos so you can get started straight away. And the sooner you get started, the sooner you'll reach your goal.

Here again is the link to all my courses.

https://LearnProgrammingAcademy.com

I look forward to working with you.

Cheers,

Tim Buchalka

P.S. If you have any other questions relating to your career, visit my blog and ask away. I'll be happy to answer any questions you post.