How to Learn a Programming Language

If you have an interest in creating computer programs, mobile apps, websites, games or any other piece of software, you'll need to learn how to program. Programs are created through the use of a programming language. This language allows...

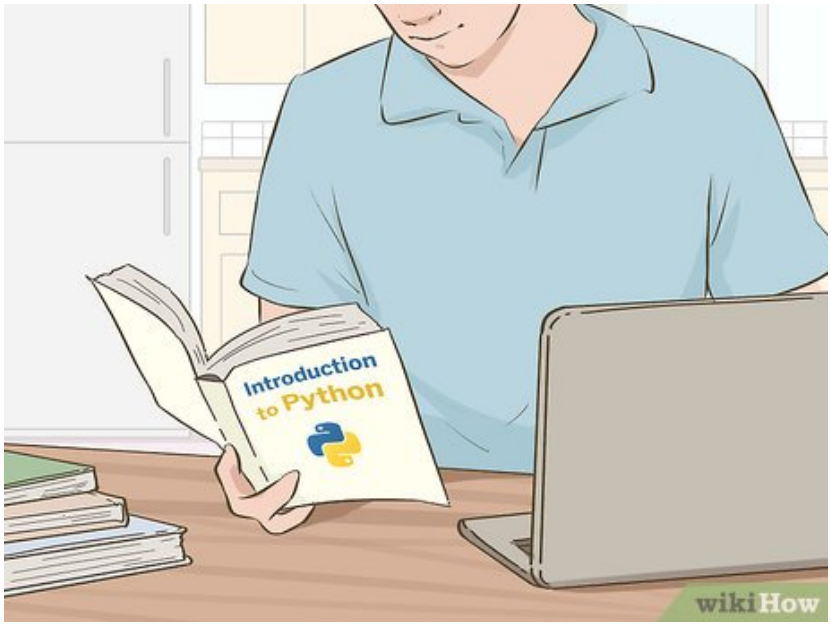
Part 1 of 6:

Choosing a Language

1. 

Determine your area of interest. You can start learning with any programming language (though some are definitely "easier" than others), so you'll want to start by asking yourself what it is you want to accomplish by learning a programming language. This will help you determine what type of programming you should pursue, and provide you a good starting point.

1. If you want to get into web development, you'll have a whole different set of languages that you'll need to learn as opposed to developing computer programs. Mobile app developing requires a different skillset than machine programming. All of these decisions will influence your direction.

2. 

Consider starting with a "simpler" language. Regardless of your decision, you may want to consider starting with one of the high-level, simpler languages. These languages are especially useful for beginners, as they teach basic concepts and thought processes that can apply to virtually any language.[1]

1. The two most popular languages in this category are Python and Ruby. These are both object-oriented web application languages that use a very readable syntax.
2. "Object-oriented" means that the language is built around the concepts of "objects", or collections of data, and their manipulation. This is a concept that is used in many advanced programming languages such as C++, Java, Objective-C, and PHP.

3. 

Read through some basic tutorials for a variety of languages. If you're still not sure which language you should start learning, read through some tutorials for a few different languages. If one language makes a bit more sense than the others, try it out for a bit to see if it clicks. There are countless tutorials for every programming available online, including many on wikiHow:

1. Python - A great starter language that is also quite powerful when you get familiar with it. Used for many web applications and a number of games.
2. Java - Used in countless types of programs, from games to web applications to ATM software.
3. HTML - An essential starting place for any web developer. Having a handle on HTML is vital before moving on to any other sort of web development.
4. C - One of the older languages, C is still a powerful tool, and is the basis for the more modern C++, C#, and Objective-C.

Score
0 / 0

Part 1 Quiz

If you want to learn the basics of web development first, which programming language should you start with?

Python

Not quite! Python is a great programming language to learn, but it isn't focused on the most basic concepts of programming languages. Instead, you can learn Python if you're interested in creating web applications and games. Pick another answer!

Java

Not exactly! Java is a common programming language, but if you want to learn the basic concepts first, you should try a different language. Instead, learn Java if you're interested in jumping right into web applications, gaming, and even banking software. Try another answer...

HTML

Yes! HTML software is arguably one of the most fundamental programming languages. HTML is an excellent starting place for web developers to learn the skills they need to start working on large web development projects. Read on for another quiz question.

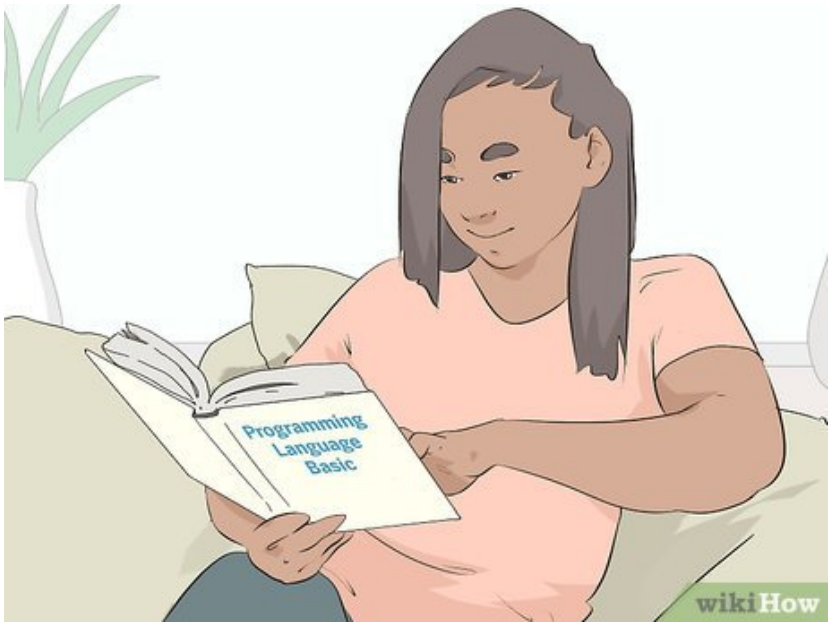
C

Try again! C is an older language that is still relevant today. C is useful to know because it is the basis for more common programming languages today, like C++, C#, and Objective C. However, C is not typically the best language to learn for web development basics. Choose another answer!

Want more quizzes?
Keep testing yourself!

Part 2 of 6:

Starting Small

1. 

Learn the core concepts of the language. While the parts of this step that apply will vary depending on the language you choose, all programming languages have fundamental concepts that are essential to building useful programs. Learning and mastering these concepts early will make it easier to solve problems and create powerful and efficient code. Below are just some of the core concepts found in many different languages:

1. Variables - A variable is a way to store and refer to changing pieces of data. Variables can be manipulated, and often have defined types such as "integers", "characters", and others, which determine the type of data that can be stored. When coding, variables typically have names that make them somewhat identifiable to a human reader. This makes it easier to understand how the variable interacts with the rest of the code.
2. Conditional Statements - A conditional statement is an action that is performed based on whether the statement is true or not. The most common form of a conditional statement is the "If-Then" statement. If the statement is true (e.g. x = 5) then one thing happens. If the statement is false (e.g. x != 5), then something else happens.
3. Functions or Subroutines - The actual name for this concept may be called something different depending on the language. It could also be "Procedure", a "Method", or a "Callable Unit". This is essentially a smaller program within a larger program. A function can be "called" by the program multiple times, allowing the programmer to efficiently create complex programs.
4. Data input - This is a broad concept that is used in nearly every language. It involves handling a user's input as well as storing that data. How that data is gathered depend on the type of program and the inputs available to the user (keyboard, file, etc.). This is closely linked to Output, which is how the result is returned to the user, be it displayed on the screen or delivered in a file.

2. 

Install any necessary software. Many programming languages require compilers, which are programs designed to translate the code into a language that the machine can understand. Other languages, such as Python, use an interpreter which can execute the programs instantly without compiling.

1. Some languages have IDEs (Integrated Development Environment) which usually contain a code editor, a compiler and/or interpreter, and a debugger. This allows the programmer to perform any necessary function in one place. IDEs may also contain visual representations of object hierarchies and directories.
2. There are a variety of code editors available online. These programs offer different ways of highlighting syntax and provide other developer-friendly tools.

Score
0 / 0

Part 2 Quiz

Which aspect of a program is responsible for storing and referring to changing data?

Conditional statements.

Nope! Conditional statements are not in charge of storing or referring to data. Instead, conditional statements are actions that are performed based on whether a statement is true or not. Pick another answer!

Data input.

Try again! Data input (new data) does not have much to do with storing and referring back to old data, even if the data has changed. However, data input is a part of almost every language, and is responsible for handling user input. Choose another answer!

Functions

Not exactly! Programming language functions or subroutines are not involved in the storing and referencing of data. Instead, functions are smaller programs inside larger programs that allow for the creation of more complex applications. Try again...

Variables

That's right! Variables are a core concept in most programming languages. They are involved in storing and then referring to changing data in a program, and they can also be manipulated. Read on for another quiz question.

Want more quizzes?
Keep testing yourself!

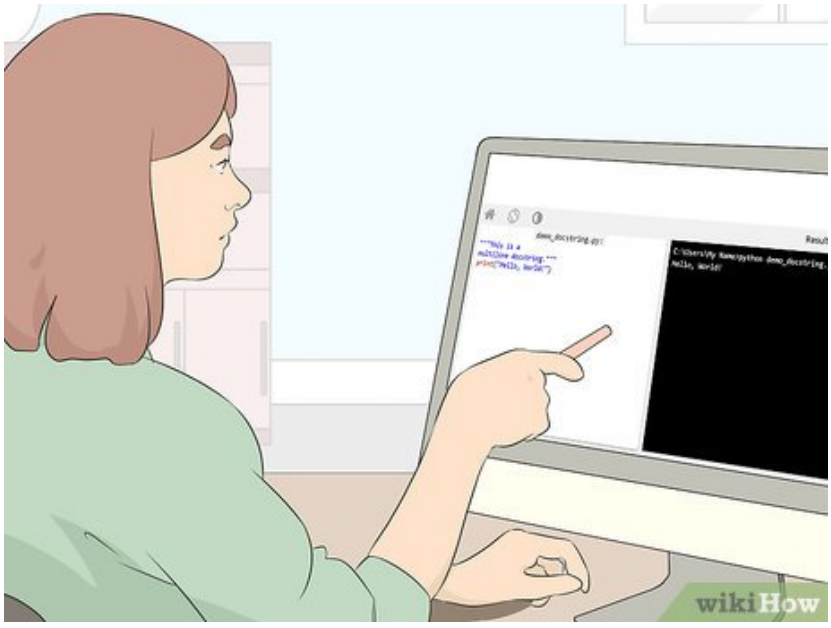
Part 3 of 6:

Creating Your First Program

1. 

Focus on one concept at a time. One of the first programs taught for any language is the "Hello World" program. This is a very simple program that displays the text "Hello, World" (or some variation), on the screen. This program teaches first-time programmers the syntax to create a basic, functioning program, as well as how to handle displaying output. By changing the text, you can learn how basic data is handled by the program. Below are some wikiHow guides on creating a "Hello World" program in various languages:

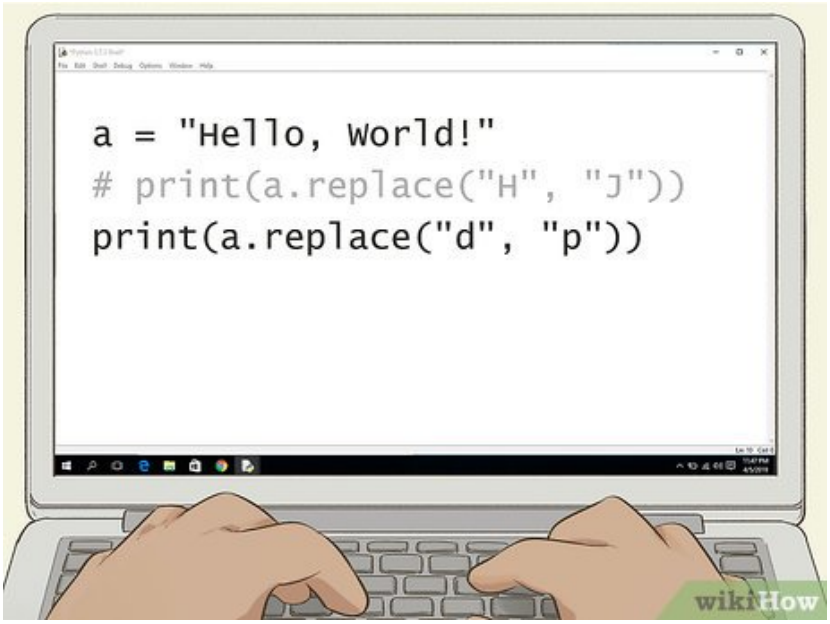
1. Hello World in Python
2. Hello World in Ruby
3. Hello World in C
4. Hello World in PHP
5. Hello World in C#
6. Hello World in Java

2. 

Learn through deconstruction of online examples. There are thousands of code examples online for virtually every programming languages. Use these examples to examine how various aspects of the language work and how different parts interact. Take bits and pieces from various examples to create your own programs.

3. 

Examine the syntax. The syntax is the way the language is written so that the compiler or interpreter can understand it. Each language has a unique syntax, though some elements may be shared across multiple languages. Learning the syntax is essential for learning how to program in the language, and is often what people think of when they think about computer programming. In reality, it is simply the foundation upon which more advanced concepts are built.

4. 

Experiment with changes. Make changes to your example programs and then test the result. By experimenting, you can learn what works and what doesn't much quicker than by reading a book or guide. Don't be afraid to break your program; learning to fix errors is a major part of any development process, and new things almost never work right the first time.[2]

5. 

Start practicing debugging. When you're programming, you're invariably going to come across bugs. These are errors in the program, and can manifest virtually anywhere. Bugs can be harmless quirks in the program, or they can be major errors that keep the program from compiling or running. Hunting down and fixing these errors is a major process in the software development cycle, so get used to doing this early.

1. As you experiment with changing basic programs, you're going to come across things that don't work. Figuring out how to take a different approach is one of the most valuable skills you can have as a programmer.

6. 

Comment all of your code. Nearly all programming languages have a "comment" function that allows you to include text that is not processed by the interpreter or compiler. This allows you to leave short, but clear, human-language explanations of what the code does. This will not only help you remember what your code does in a large program, it is an essential practice in a collaborative environment, as it allows others to understand what your code is doing.

Score
0 / 0

Part 3 Quiz

Why should you make changes to your example programs when you're learning a programming language?

You can learn to fix your mistakes.

Almost! Fixing mistakes, or "debugging" your code, is a vital part of the learning process. If you start making changes to the example codes you're practicing with, you can go back through and learn how to fix mistakes that you make. While this is correct, there are also other reasons you should make changes to your programs. Try again...

You can learn what works and what doesn't.

You're partially right! If you make changes to your example programs, you'll quickly learn which changes will work and which won't. This trial and error process is an essential part of learning the language you chose. This is true, but there are other reasons you should change your example programs. Try again...

You can typically learn faster than with a book.

You're not wrong, but there's a better answer! Going off-script, or making changes to the example programs you're learning from, can often help you learn faster. You'll gain firsthand experience with making changes that aren't already scripted. Choose another answer!

All of the above.

Correct! All of these reasons explain why you should try making changes to your example programs. Learning how to debug your programs and recognize what works (or doesn't work) firsthand can help you discover the programming language faster than if you only follow along with an example program. Read on for another quiz question.

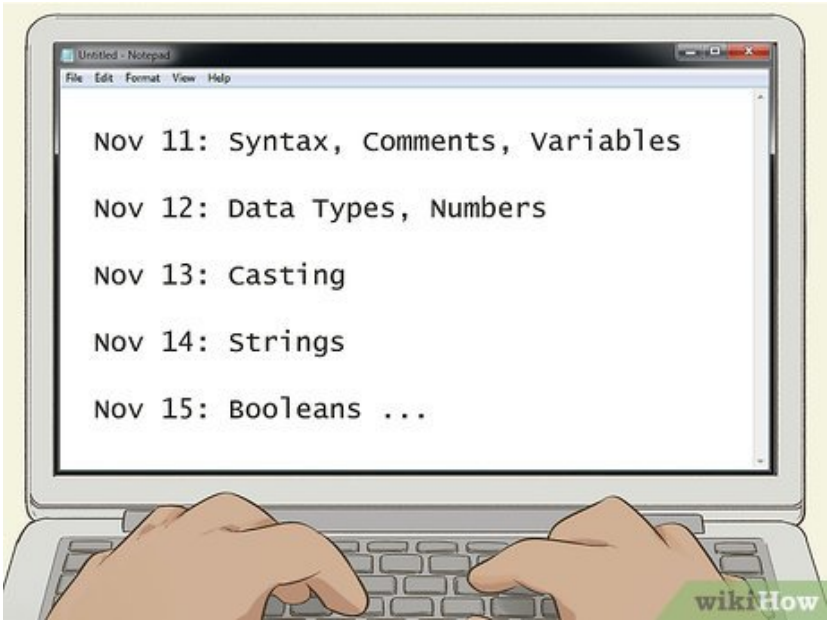
Want more quizzes?
Keep testing yourself!

Part 4 of 6:

Practicing Regularly

1. 

Code daily. Mastering a programming language takes time above all else. Even a simpler language like Python, which may only take a day or two to understand the basic syntax, takes lots of time to become truly proficient at. Like any other skill, practice is the key to becoming more proficient. Try to spend at least some time each day coding, even if it's only for an hour between work and dinner.

2. 

Set goals for your programs. By setting attainable but challenging goals, you will be able to start solving problems and coming up with solutions. Try to think of a basic application, such as a calculator, and develop a way to make it. Use the syntax and concepts you've been learning and apply them to practical uses.

3. 

Talk with others and read other programs. There are lots of programming communities dedicated to specific languages or disciplines. Finding and participating in a community can do wonders for your learning. You will gain access to a variety of samples and tools that can aid you in your learning process. Reading other programmers' code can inspire you and help you grasp concepts that you haven't mastered yet.[3]

1. Check out programming forums and online communities for your language of choice. Make sure to participate and not just constantly ask questions. These communities are usually viewed as a place of collaboration and discussion and not simply Q&A. Feel free to ask for help, but be prepared to

show your work and be open to trying different approaches.

2. Once you have some experience under your belt, consider attending a hack-a-thon or programming jam. These are events where individuals or teams compete against the clock to develop a functional program, usually based around a specific theme. These events can be a lot of fun and are a great way to meet other programmers.

4. 

Challenge yourself to keep it fun. Try to do things that you don't know how to do yet. Research ways to accomplish the task (or a similar one), and then try to implement that in your own program. Try to avoid being content with a program that "basically" works; do everything you can to make sure every aspect works flawlessly.

Score
0 / 0

Part 4 Quiz

Why should you join a programming jam?

They are helpful Q&A sessions that teach you basic concepts.

Nope! While some programming jams may have a supplementary Q&A session, the events themselves are not designed as question and answer segments. Instead, if you are struggling to find answers to your questions, seek out other programmers who can help you, or read other programs to learn more. Guess again!

They are challenging competitions that motivate you to learn.

Yes! Programming jams and hack-a-thons are competitions. Multiple programmers will come together and compete to develop a functional program first. You can learn a lot from a programming jam, and signing up for one can motivate you to spend more time learning the language. Read on for another quiz question.

They are online forums where you can learn to collaborate with others.

Try again! Programming jams are not the same as programming forums. However, you can use a programming forum to collaborate in a similar way that you can in some programming jams. There's a better option out there!

Want more quizzes?
Keep testing yourself!

Part 5 of 6:

Expanding Your Knowledge

1. 

Take a few training courses. Many universities, community colleges, and community centers offer programming classes and workshops that you can attend without having to enroll in the school. These can be great for new programmers, as you can get hands-on help from an experienced programmer, as well as network with other local programmers.

2. 

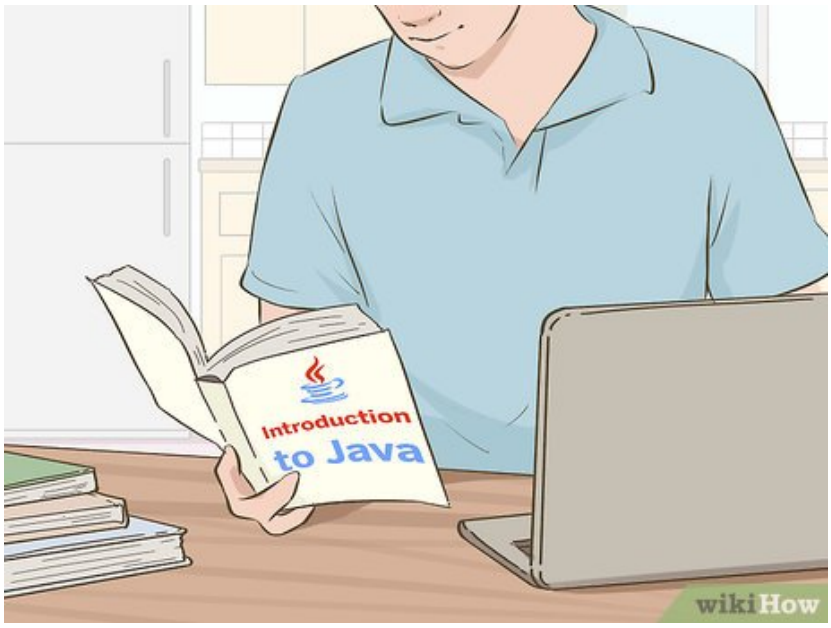
Buy or borrow a book. There are thousands of instructional books available for every conceivable programming language. While your knowledge should not come strictly from a book, they make great references and often contain a lot of good examples.

3. 

Study math and logic. Most programming involves basic arithmetic, but you may want to study more advanced concepts. This is especially important if you are developing complex simulations or other algorithm-heavy programs. For most day-to-day programming, you don't need much advanced math. Studying logic, especially computer logic, can help you understand how best to approach complex problem solving for more advanced programs.

4. 

Never stop programming. There is a popular theory that becoming an expert takes at least 10,000 hours of practice. While this is up for debate, the general principle remains true: mastery takes time and dedication. Don't expect to know everything overnight, but if you stay focused and continue to learn, you may very well end up an expert in your field.[4]

5. 

Learn another programming language. While you can certainly get by with mastering one language, many programmers help their chances of success in the field by learning multiple languages. Their second or third languages are usually complementary to their first one, allowing them to develop more complex and interesting programs. Once you have a good grasp on your first program, it may be time to start learning a new one.

1. You will likely find that learning your second language goes much quicker than the first. Many core concepts of programming carry over across languages, especially if the languages are closely related.

Score
0 / 0

Part 5 Quiz

True or false: You need to practice programming for at least 1,000 hours before you can call yourself an expert.

True

Nope! The prevailing theory, which may or may not be accurate, is that you need to practice for 10,000 hours to call yourself an expert, not 1,000. There is no hard or fast rule about how many hours you should practice in order to become fluent in a particular programming language, but you should make sure you work on the language regularly, so you can learn quickly and not become rusty. Try another answer...

False

Yup! There are no strict rules on when you can call yourself an expert at a skill, but the general theory is that you need to practice for 10,000 hours, not 1,000. However, if you practice regularly and truly apply yourself to learning the programming language, you can become functional in less time than you might expect. Read on for another quiz question.

Want more quizzes?
Keep testing yourself!

Part 6 of 6:

Applying Your Skills

1. 

Enroll in a four-year program. While not strictly necessary, a four-year program at a college or university can expose you to a variety of different languages, as well as help you network with

professionals and other students. This method certainly isn't for everyone, and plenty of successful programmers never attended a four-year institution.

2. 

Create a portfolio. As you create programs and expand your knowledge, make sure that all of your best work is saved in a portfolio. You can show this portfolio to recruiters and interviewers as an example of the work you do. Make sure to include any work done on your own time, and ensure that you are allowed to include any work done with another company.

3. 

Do some freelance work. There is a very large freelance market for programmers, especially mobile app developers. Take on a few small freelance jobs to get a feel for how commercial programming works. Oftentimes you can use the freelance jobs to help build your portfolio and point to published work.

4. 

Develop your own freeware or commercial programs. You don't have to work for a company to make money programming. If you have the skills, you can develop software yourself and release it for purchase, either through your own website or through another marketplace. Be prepared to be able to provide support for any software you release for commercial sale, as customers will expect their purchase to work.

1. Freeware is a popular way to distribute small programs and utilities. The developer doesn't receive any money, but it's a great way to build name recognition and make yourself visible in the community.

Score
0 / 0

Part 6 Quiz

Why would you want to enroll in a four-year program?

You can learn more than one language faster.

Almost! In a four-year program, you'll likely be exposed to a wide variety of programming languages. Learning more than one language can take a significant amount of time, but college CS courses accelerate the learning process. While this is correct, there are also other reasons someone might enroll in a four-year program. Guess again!

You can meet professional programmers.

You're partially right! Four-year programs will expose you to professionals that you can network with. You can often use these connections to find a job after you graduate from the program. Choose another answer!

You can make a name for yourself in the programming field.

You're not wrong, but there's a better answer! If you do well in a four-year program, you can often introduce yourself to the programming network and make a name for yourself in the field. Four-year programs can help

you get your name out there and establish you as a potentially valuable addition to the programming market. Choose another answer!

All of the above.

Exactly! All of these are reasons that many people join a four-year program. Knowing more than one programming language is a valuable skill set, and a four-year program can help you learn them faster. You can also network with professionals and other students and start making a name for yourself in the field. Read on for another quiz question.

Want more quizzes?
Keep testing yourself!