

# Forum:[Hack Forums](#)

Thread: [What language should you learn?](#)

**Faith Connors** - August 14, 2016, 3:34 am

I've seen the same thread in here all the time so I'm gonna try to clear it up.

This question has existed in here as long as I remember my time in here. It shows up every few days, a new member asking what language is the best for hacking. Of course, it depends what you're trying to do, but I'm going to clear it up to the best of my ability. First of all there are three main language types, Markup, Programming, and Query.

## Markup Languages:

Markup languages are probably the most useless languages, but are the easiest to learn. I use that term a bit sparingly so I'll give my two cents on it.

Unlike programming languages, markup languages are very static in the terms that they don't actually do anything on their own. Markup languages are made to format text, maybe style it. If you've spent time in here, you'll see people recommend learning HTML and CSS. These languages are markup, they're for building websites. HTML is literally an acronym for "HyperText Markup Language." Now, you're wondering how they get all kinds of fancy buttons and how stuff actually happens in your favourite online game that claims to be written in HTML5. Well the truth is, the game isn't written in HTML5. Maybe the site hosting the game is, but the game itself is probably JavaScript or some other scripting language. I'll talk about scripting languages with programming languages a bit later, but for now we're on markup languages.

HTML and CSS are probably the most common languages people know because they're simple, but to be honest those are the only markup languages people will have heard of, there are many more, and some I find more useful than others. Although markup languages are much easier to code in, you can do some really interesting stuff with them, not just building websites.

Another really popular markup language is the eXtensible Markup Language, AKA XML, and it's HTML counterpart, XHTML. Many people have heard of this one but don't really know what to do with it. It may very well be my favourite markup language because of its flexibility. I even used it to fix up and make some mods for a game called Mabinogi, a few years ago. Another markup language is made by Adobe, a bit different than either of the ones I mentioned, but relatively powerful in conjunction with their application environment. This language is CFML, the ColdFusion Markup Language. It's for deploying your ColdFusion applications online. So yeah, markup languages aren't "completely useless," but they're much simpler than actual programming languages, and don't really do as much.

I would recommend trying to learn a markup language if you are a complete beginner and have never touched a line of code before, or even seen one.

## Programming Languages:

Everyone seems to classify languages as programming languages when they clearly aren't, so I'm gonna try and clear some of this up. Programming languages are the languages used to actually make applications. This is what your favourite online game is made in, that software at work you hate, the operating system and web browser you're using right now. Most of it is an actual programming language (maybe some XML for logs and stuff.)

Programming Languages are a little bit of a broad area. You see, first you have to divide it into two completely separate areas. There are compiled languages and there are interpreted languages.

### Compiled Languages:

Now, if you are a hacker, and you write malware, this is the kind of language you want to learn. When a program written in a compiled language runs, the code is hidden, and that makes it harder for an antivirus to detect that it's a virus. So as a result most malware is written in a compiled language. Now, it also depends what specific language you're after. There are two more sections we can divide into, Native and Runtime.

Remember the day you got your new computer and it was a while ago and you wanted to play Minecraft, but because Minecraft is a Java game, you needed to install Java first?

Well, first of all what exactly were you installing?

You installed the JRE, the Java Runtime Environment. You see, by default, Windows doesn't have support for running Java programs, so when it prompted you to go and download Java, it was asking you to download the libraries and all the packages you need to run Java applications. The other side of this is Native Languages

Now, a Native language means the language will work by default on the operating system without any runtime. This is because when it is compiled, it isn't compiled in a way that it requires all the libraries and packages that the runtime needs to provide it, instead when the compiler, well, compiles the program, it converts it all into machine code, aka the 0's and 1's that everyone thinks is hacking if you pull it up in a public place. This is the barebones code and operates on such a basic level that the hardware itself could interpret it on its own. When that's all converted to text, you can see it's just Assembly code, the one-step-above machine code that is incredibly hard to write on your own, but can be converted with the right compiler, like the GNU-Cross-Compiler (GCC.)

Enough about compiled languages, we want to get into interpreted languages.

### Interpreted Languages

I'm going to just copy and paste surelyenough's (UID=3217184) edit suggestion because it makes sense and there's probably no other way I could write it.

(Aug 14, 2016 04:23 AM)surelyenough Wrote: [\[CITE\]](#) Memory formation is not why malware developers ignore Python. In fact, memory is much more complicated when you're using a language like Python because almost everything is an object that is dynamically allocated on the Heap and not the Stack segment of our memory segmentation. Write simple code in C, you'll see the memory formation actually is very very clear. Function prologues, stack formation, allocations etc. It all makes beautiful sense. Python on the other hand...

You don't write malware in Python because Python is interpreted, not compiled. Meaning that I'm supposed to execute your malware through python instead of accidentally executing it natively through whatever means (Opening it, or having my browser run it as an exploit, or having another

form of malware run it). If you'll want it bad enough, you can actually use py2exe or whatever it's called as a compiler to Python. However, it's slow, inefficient and not the purpose Python was made for. You write malware in fast, efficient languages that are as low as they can get to the bones of your PC. There is no point in sugar-coating code when dealing with malware.

Ruby is a pretty old language, been around since 1995 and it's still very commonly used today, in an environment called Metasploit. It is used to create payloads, think of it like malware but sent to specific targets so they can be exploited like vulnerabilities. It's a rather flexible language, been used for all kinds of things, and it's even making a comeback in game development, if you've ever heard of RPGmaker.

Python is another example of an interpreted language, a scripting language. It's easy to pick up because the syntax is simple, and it's powerful if you're using all the right libraries and know what you're doing. It's been recently used to start making payloads in Metasploit as well.

Honestly I just like interpreted/scripting languages because they're relatively simple and are good for automation. We all hate doing repetitive menial tasks, right?

### Query Languages:

Query languages are the last type of language I'll be talking about, although it's very different than anything we've seen here so far.

We're gonna take a look at SQL for the most part here. SQL is the acronym for Structured Query Language. It's what allows a user to communicate with a database and call up data. You give it a query, and it gives you a response.

MySQL is commonly mistaken as a query language, as I did prior to this edit. MySQL is actually a piece of software which acts like a query language, but has slight variations in syntax, and once it receives the query in its syntax, it'll convert it to the actual SQL query, which can be understood by the software hosting the database. This is what we call a Database Management Service, DBMS for short.

Now, a little bit about APIs. I'm aware that APIs are very different than actual query languages or whatever, but it's probably important that I talk about them a little bit in here. API is an acronym for Application Program Interface. In its own way it can help you program things, almost like using building blocks to figure out how you can get something to work.

Let's say I have something called a proxy rotator. What it does is calls up a server to give me a random proxy from a list. Now, I could implement this in a program to make it look like I'm accessing a website or some server from all kinds of different connections, just by making the API pull a proxy from its server every time my program wanted to make a request.

I could also use a scraping API. It would help me get data off of a website much more easily. Pastebin offers an API which can help you get the 1000 most recent pastes, for data mining, other general research and so on.

So in a way an API can be considered a "query language" but doesn't actually hit the mark because it isn't a language in itself, it runs off some sort of language already. APIs can be used in most of the most popular languages already, like Python and Node.JS

So what are you going to do with all this information? Well hopefully you understand the difference between different types of languages now, and know what kind of language you're hoping to learn. I've compiled a small list of resources you can use to try and find some languages and where to learn them.

<https://www.codecademy.com/> is a great resource for web-development. They have courses on HTML and CSS, JavaScript, PHP and Python.

<http://www.w3schools.com/> is a little like codecademy, but it doesn't hold your hand as much. It'll help you start to learn on your own.

<https://github.com/vhf/free-programming-books/blob/master/free-programming-books.md> is a fantastic resource to find every possible language you can think of, and some books for you to learn them.

<http://www.programmableweb.com/> A huge directory and listing of thousands of APIs. Take your pick.

Here's a list of good languages that are still used today and are powerful.

Perl - Scripting language, old but still used

Python - Scripting language, relatively new, widely used

Ruby - Old, making a comeback, flexible and powerful

C# - Been out for a while but people only started to notice it now for how powerful it is.

Java - Was dominating the market because of cross-platform, but it's losing speed recently.

C - The original, the powerful, it'll teach you everything you need to know.

PHP - Used for web development and scripting. It's all server side.

SQL - Query language, for calling up data from databases online. Used in SQL injections, a web vulnerability/exploit.

If there is any bad info in here, let me know and I'll fix it. Sorry, I'm still coming down from painkillers.

To all new members, welcome to HackForums and enjoy your stay.

~Faith

EDITS:

-Added ProgrammableWeb to list of resources.

EDIT SUGGESTIONS COURTESY OF UID=3217184, surelyenough

-About native languages

-Some edits to APIs

-Edits to Query Languages

**SIM-2914476** - August 14, 2016, 3:35 am

"PHP - Used for web development and scripting. It's all server sided."

No need for the "d" at the end.

Also take advantage of Code [hr] to seperate.

Otherwise nice job OP.

**Faith Connors** - August 14, 2016, 3:42 am

(Aug 14, 2016 03:35 AM)ShinyMK Wrote: [\[CITE\]](#) "PHP - Used for web development and scripting. It's all server sided."

No need for the "d" at the end.

Also take advantage of Code [hr] to seperate.

Otherwise nice job OP.

Oh shit thanks for the heads up

**dendro** - August 14, 2016, 3:47 am

Thanks for the hq thread good stuff op :)

**Faith Connors** - August 14, 2016, 3:48 am

(Aug 14, 2016 03:47 AM)dendro Wrote: [\[CITE\]](#) Thanks for the hq thread good stuff op :)

When you're bed ridden, you've got time to kill, and when you have a laptop, you may as well waste it online :^)

**Z3r0C0d3** - August 14, 2016, 4:17 am

Quote:Programming Languages are a little bit of a broad area. You see, first you have to divide it into two completely separate areas

Actually , there are 3 types , Low level , High level(Compiled Languages) and Interpreted , the low level programming language doesn't require compiler because they are close to the machine code , example for low level language is Assembly

**surelyenough** - August 14, 2016, 4:23 am

(Aug 14, 2016 04:17 AM)Z3r0C0d3 Wrote: [\[CITE\]](#) Actually , there are 3 types , Low level , High level(Compiled Languages) and Interpreted , the low level programming language doesn't require compiler because they are close to the machine codeNot exactly accurate. Low Level programming (Assembly) requires "a compiler". It's called an assembler. Some examples of them: NASM, FASM, MASM, etc.

You're correct in the sense that assemblers are not actually turning code into object files only to group them together in order to perform linking according to the symbol table. That's the job of GCC. The assembler simply takes out the mnemonics we've put in, puts the machine code instead according to it's specific dialect (8086 & NASM on x86\_64 etc) so it's a simpler job, but it still creates a binary executable, it'll still segment your binary, it'll still sort out your ASM accordingly to the "compile" flags. Not a compiler, But it still needs to have something go through your assembly.

Anyway, Good thread, I like it. Some constructive criticism.

Quote:Languages like C and Delphi are examples of languages that will run by default on a Windows machine, without having to install anything else. This is usually because the operating system is already written in the same language.

Not really. If you'll take Rust, another generic-purpose high-level compiled language, and start writing code and compiling it you'll find out it'll work straight out the box. The reasoning for that is these are natively-compiled language. Meaning that these compilers that go through the language start a procedure that will eventually convert the code to machine code according to the demands of the enviroment and workspace. The output file is a binary executable that is written in 0's and 1's, if you'll go with it through a dissassembler you'll find out the machine code actually makes sense and can be viewed as Assembly. Your PC simply reads the instructions and does them. Now, if your writing tried to aim for "being able to write OS-depended services" these are libraries. \*nix has POSIX. Windows has WinAPI. Each provides a huge API for the coder to be able to talk with the OS. So yeah, basically any programming language can be native and it has nothing to do whether

the OS was written in the same language.

Quote: Interpreted languages make all the code visible in the memory when running, so if you try to write malware in these kind of languages, it's going to be very hard because antiviruses will automatically detect what you're trying to do as soon as it runs. But scripting languages aren't all useless in the hacking world.

Memory formation is not why malware developers ignore Python. In fact, memory is much more complicated when you're using a language like Python because almost everything is an object that is dynamically allocated on the Heap and not the Stack segment of our memory segmentation. Write simple code in C, you'll see the memory formation actually is very very clear. Function prologues, stack formation, allocations etc. It all makes beautiful sense. Python on the other hand...

You don't write malware in Python because Python is interpreted, not compiled. Meaning that I'm supposed to execute your malware through python instead of accidentally executing it natively through whatever means (Opening it, or having my browser run it as an exploit, or having another form of malware run it). If you'll want it bad enough, you can actually use py2exe or whatever it's called as a compiler to Python. However, it's slow, inefficient and not the purpose Python was made for. You write malware in fast, efficient languages that are as low as they can get to the bones of your PC. There is no point in sugar-coating code when dealing with malware.

I'd recommend a rewrite for the Query Languages thing, the only actual query-structured language (that is popular) is SQL. API's aren't really languages, they are however critical system components and are important. There are specifications on how to write APIs such as SOAP while making them RESTful. These are technologies that have similarities to languages in how strict they appear however they're not languages.

MySQL is a DBMS not a language. It's a program that breaks down MySQL SQL dialect and executes it. Other DBMS include MsAccess, MSSQL, Oracle, PostgreSQL etc. The differences between them are minor, and usually revolve around syntax.

Quote: AKA XML, and it's HTML counterpart, XHTML.

Isn't XHTML simply a strict HTML standard. Just like HTML 4.0. I learnt something new today as well, see? :) Thanks.

Quote: When a program written in a compiled language runs, the code is hidden, and that makes it harder for an antivirus to detect that it's a virus

This isn't how it really works... The "code" isn't hidden. There is no code. There is machine code, and it's located at the code segment: [https://en.wikipedia.org/wiki/Code\\_segment](https://en.wikipedia.org/wiki/Code_segment)

As for AV bypassing, it has little to do with language choice and more to do with your abilities as a malware developer. Knowing what to hook, what to dynamically link, what to use where is the real skill. I can not detail about this in public. PM me if you have questions in this field.

I'd recommend you do an entire overhaul to this thread because it has serious potential to be very good.

**Faith Connors** - August 14, 2016, 5:15 am

(Aug 14, 2016 04:23 AM) surelyenough Wrote: [\[CITE\]](#) Not exactly accurate. Low Level programming (Assembly) requires "a compiler". It's called an assembler. Some examples of them: NASM, FASM, MASM, etc.

You're correct in the sense that assemblers are not actually turning code into object files only to group them together in order to perform linking according to the symbol table. That's the job of GCC. The assembler simply takes out the mnemonics we've put in, puts the machine code instead

according to it's specific dialect (8086 & NASM on x86\_64 etc) so it's a simpler job, but it still creates a binary executable, it'll still segment your binary, it'll still sort out your ASM accordingly to the "compile" flags. Not a compiler, But it still needs to have something go through your assembly.

Anyway, Good thread, I like it. Some constructive criticism.

Not really. If you'll take Rust, another generic-purpose high-level compiled language, and start writing code and compiling it you'll find out it'll work straight out the box. The reasoning for that is these are natively-compiled language. Meaning that these compilers that go through the language start a procedure that will eventually convert the code to machine code according to the demands of the environment and workspace. The output file is a binary executable that is written in 0's and 1's, if you'll go with it through a disassembler you'll find out the machine code actually makes sense and can be viewed as Assembly. Your PC simply reads the instructions and does them. Now, if your writing tried to aim for "being able to write OS-dependent services" these are libraries. \*nix has POSIX. Windows has WinAPI. Each provides a huge API for the coder to be able to talk with the OS. So yeah, basically any programming language can be native and it has nothing to do whether the OS was written in the same language.

Memory formation is not why malware developers ignore Python. In fact, memory is much more complicated when you're using a language like Python because almost everything is an object that is dynamically allocated on the Heap and not the Stack segment of our memory segmentation. Write simple code in C, you'll see the memory formation actually is very very clear. Function prologues, stack formation, allocations etc. It all makes beautiful sense. Python on the other hand...

You don't write malware in Python because Python is interpreted, not compiled. Meaning that I'm supposed to execute your malware through python instead of accidentally executing it natively through whatever means (Opening it, or having my browser run it as an exploit, or having another form of malware run it). If you'll want it bad enough, you can actually use py2exe or whatever it's called as a compiler to Python. However, it's slow, inefficient and not the purpose Python was made for. You write malware in fast, efficient languages that are as low as they can get to the bones of your PC. There is no point in sugar-coating code when dealing with malware.

I'd recommend a rewrite for the Query Languages thing, the only actual query-structured language (that is popular) is SQL. API's aren't really languages, they are however critical system components and are important. There are specifications on how to write APIs such as SOAP while making them RESTful. These are technologies that have similarities to languages in how strict they appear however they're not languages.

MySQL is a DBMS not a language. It's a program that breaks down MySQL SQL dialect and executes it. Other DBMS include MsAccess, MSSQL, Oracle, PostgreSQL etc. The differences between them are minor, and usually revolve around syntax.

Isn't XHTML simply a strict HTML standard. Just like HTML 4.0. I learnt something new today as well, see? :) Thanks.

This isn't how it really works... The "code" isn't hidden. There is no code. There is machine code, and it's located at the code segment: [https://en.wikipedia.org/wiki/Code\\_segment](https://en.wikipedia.org/wiki/Code_segment)

As for AV bypassing, it has little to do with language choice and more to do with your abilities as a malware developer. Knowing what to hook, what to dynamically link, what to use where is the real skill. I can not detail about this in public. PM me if you have questions in this field.

I'd recommend you do an entire overhaul to this thread because it has serious potential to be very good.

Shit man thanks for the heads up, I'll be adding stuff to it later, I feel like I'm about to pass out right now lmao.

**surelyenough** - August 14, 2016, 5:16 am

(Aug 14, 2016 05:15 AM)Faith Connors Wrote: [\[CITE\]](#) Shit man thanks for the heads up, I'll be adding stuff to it later, I feel like I'm about to pass out right now lmao.no worries it's all good

**Faith Connors** - August 14, 2016, 6:04 am

(Aug 14, 2016 05:16 AM)surelyenough Wrote: [\[CITE\]](#) no worries it's all good  
Alright mostly edited to your suggestions. Decided not to put in the last bit because malware = bad and bad is not allowed on HF. But still, great suggestions.

**Faith Connors** - August 18, 2016, 2:00 am

Just gonna bump this thread in regards to recent threads of users asking where to start and other users responding. Lots of information here, take advantage of it.

**~ Left HF For Good ~** - August 18, 2016, 2:08 am

Faith, would you mind if I added your thread link into my thread? Of course, it would link to your thread, and you will have credits!

**Faith Connors** - August 18, 2016, 2:09 am

(Aug 18, 2016 02:08 AM)ChilledWhiteHat Wrote: [\[CITE\]](#) Faith, would you mind if I added your thread link into my thread? Of course, it would link to your thread, and you will have credits!  
I already asked you in like the third post lmao

**~ Left HF For Good ~** - August 18, 2016, 2:10 am

(Aug 18, 2016 02:09 AM)Faith Connors Wrote: [\[CITE\]](#) I already asked you in like the third post lmao  
Oh my, I am so sorry! I must not have seen it. Let me go back to my thread. ?

Edit: Added it to my thread, as well gave you credits. Thank you very much, Faith!

**Arkks** - August 18, 2016, 6:06 am

tbh it all depends what you trying to do. if you heading into botnets C and web dev is a good place to go but if your working on tools and other fun stuff go towards c# and .net

**Faith Connors** - August 18, 2016, 6:23 am

(Aug 18, 2016 06:06 AM)Arkks Wrote: [\[CITE\]](#) tbh it all depends what you trying to do. if you heading into botnets C and web dev is a good place to go but if your working on tools and other fun stuff go towards c# and .net

Well I mean first of all, .NET is a framework, not a language, and also a lot of tools can be written in C which arguably makes them better because of the versatility of it running natively on a system. Also proven by the NSA in the latest leak, I've been busy with some people reversing the tools and almost all of them are written in C except for the scripts.

Web-dev can be good for botnets and stuff, but not front-end stuff in most cases, usually you'd be aiming for something like PHP for the back-end, although knowing front-end can also be very useful.

Choosing a language does really depend on what you're trying to do, but I tried to give a good variety in suggestions,with brief descriptions, which you can go back into the information to check on to see if they fit what you're trying to do.

**r421x0** - August 18, 2016, 7:01 am

This is a really nice thread you've got here.

Bumping it up ;)

~ **Left HF For Good** ~ - August 18, 2016, 8:55 am

(Aug 18, 2016 06:23 AM)Faith Connors Wrote: [\[CITE\]](#) Well I mean first of all, .NET is a framework, not a language, and also a lot of tools can be written in C which arguably makes them better because of the versatility of it running natively on a system. Also proven by the NSA in the latest leak, I've been busy with some people reversing the tools and almost all of them are written in C except for the scripts.

Web-dev can be good for botnets and stuff, but not front-end stuff in most cases, usually you'd be aiming for something like PHP for the back-end, although knowing front-end can also be very useful.

Choosing a language does really depend on what you're trying to do, but I tried to give a good variety in suggestions, with brief descriptions, which you can go back into the information to check on to see if they fit what you're trying to do.

As Faith said, .NET is not a language it self. It is simply an Integrated Development Environment (IDE).

**Faith Connors** - August 18, 2016, 9:02 am

(Aug 18, 2016 08:55 AM)ChilledWhiteHat Wrote: [\[CITE\]](#) As Faith said, .NET is not a language it self. It is simply an Integrated Development Environment (IDE).

Nope, it's a framework. Not quite IDE. An IDE is the place where you write the code itself, the framework is more like the set of default libraries and resources that the program can use out of the box.

.NET = Framework  
Metasploit = Framework

Visual Studio = IDE  
Eclipse = IDE

**wolfcham** - August 18, 2016, 12:41 pm

Good stuff man thanks lots of usefull information for people that start

**fogstone** - August 18, 2016, 1:44 pm

I liked this thread. It was well explained. I knew of these languages and I've dabbled in some of them but I don't know a few of them well. I am wondering if bash could count as a language that might be worth learning. Just as a question. I definitely will learn python again one day when I get good at what I have now. Someone told me Java is crap. It sounds like you don't agree. I don't know what to think about whether or not Java is crap because one of my computer science teachers actually loves the language. Anyways, its nice to see such a cool thread to give these coding languages as a recommendation to the user. Thanks.

**Lante** - August 18, 2016, 3:31 pm

(Aug 18, 2016 01:44 PM)fogstone Wrote: [\[CITE\]](#) I liked this thread. It was well explained. I knew of these languages and I've dabbled in some of them but I don't know a few of them well. I am wondering if bash could count as a language that might be worth learning. Just as a question. I definitely will learn python again one day when I get good at what I have now. Someone told me Java is crap. It sounds like you don't agree. I don't know what to think about whether or not Java is crap because one of my computer science teachers actually loves the language. Anyways, its nice to see such a cool thread to give these coding languages as a recommendation to the user. Thanks.  
I read your reply and it immediately made me want to ask you what your goals are with

programming? Can't really learn random languages without you requiring to learn or master them for something specific. I'm currently playing around with my Raspberry and am learning Python to control electronics and read sensory output using GPIO. When an artist wants to create an art piece, he won't start learning pencil drawing, painting, sculpturing, video editing, photoshop, photography, ... He will focus on the method required to achieve the goal that has been set, everything else is obsolete to reach that goal and will only make the whole learning curve experience bloated.

Java is fine for a lot of things, but definitely not the best pick. Too many lines of code required compared to other major languages, requires JRE (not native), horrible memory management and garbage collection, it's not something you'd want a large application to run on but it can do fine for small things.

**Faith Connors** - August 18, 2016, 4:45 pm

(Aug 18, 2016 01:44 PM)fogstone Wrote: [\[CITE\]](#) I liked this thread. It was well explained. I knew of these languages and I've dabbled in some of them but I don't know a few of them well. I am wondering if bash could count as a language that might be worth learning. Just as a question. I definitely will learn python again one day when I get good at what I have now. Someone told me Java is crap. It sounds like you don't agree. I don't know what to think about whether or not Java is crap because one of my computer science teachers actually loves the language. Anyways, its nice to see such a cool thread to give these coding languages as a recommendation to the user. Thanks.

Well, your first concern about Bash, if you're planning on using a Linux environment at all, whether it be for a computer science class or just for at home, yeah it's essential you learn it, because you're gonna be able to do jack-shit without it. The Linux environment relies entirely on the Bash shell so you'd have to learn the ins and outs, at least most of the common commands, like ls, cd, cat, mkdir, chmod, tar, rm, and so on.

As for whether it's a language or not, well you see that's where I'm a little torn. Personally I wouldn't call it a language, I would call it more of an environment but even that term isn't fitting for what it does. It's basically just a shell for you to do simple commands, but you can use it to launch all kinds of scripts and programs, like the Gnu-Cross-Compiler (GCC) or even the Xorg server (X window Manager). But .sh files, as you might be aware of, are associated with bash scripts. So in a way you can "program" with it, although all it really just does is lists out commands to do in order, checks if they complete successfully, and stuff like that. So it's not really a "programming" language, as you're still gonna be using tools developed in other languages like Python, C, Perl and Ruby within the shell.

Second part of your question, about Java.

Well, I may not express it that much in the thread because I don't want it to sound biased in any way, but personally I'm not a huge fan of Java. It's an incredibly old language, made in 1995 by SUN and for its time, it was somewhat revolutionary. It derived heavily from C and C++, and the runtime environments allowed it to be cross-platform and work really well without having to be a native language. It took off, started being used for all kinds of applications, but honestly I think it hit its "spike" around the time of the development of Minecraft. Since then, development in the language has gone down, more languages are available that are arguably better in areas where Java fell short of (virtualization) and are still cross-platform.

Java is a relatively simple language and it's really beginner friendly if you want to look at different concepts like Object-Oriented Programming. There are all sorts of resources online because it was extremely widely used. It still is, but that number has gone down drastically. Personally I'm giving it

five years til it dies out altogether (only a few people left that use it for day-to-day programming) but some applications will still probably stay written in Java, like programs in cars, your coffee machine, etc. as it's just not worth the trouble of changing it. But I can understand why people would like the language, with it's inviting atmosphere, massive community and simplicity.

As Lante said, when choosing a language it really depends what your end-goal is. I tried to give some examples of what different types of languages are meant to accomplish so people can make a better decision. People who want to develop malware should be learning HTML, just doesn't make sense, (and mind you it's not even possible) but they're be better off with an actual programming language, probably Native like C or Delphi.

~ Left HF For Good ~ - August 18, 2016, 7:33 pm

(Aug 18, 2016 09:02 AM)Faith Connors Wrote: [\[CITE\]](#) Nope, it's a framework. Not quite IDE. An IDE is the place where you write the code itself, the framework is more like the set of default libraries and resources that the program can use out of the box.

.NET = Framework  
Metasploit = Framework

Visual Studio = IDE  
Eclipse = IDE

Really now. This can honestly be debated. Some people claim that C# is a IDE, while others claim its a actual Framework. How ever, I am 50/50, not sure which is actually true.

Faith Connors - August 18, 2016, 7:37 pm

(Aug 18, 2016 07:33 PM)ChilledWhiteHat Wrote: [\[CITE\]](#) Really now. This can honestly be debated. Some people claim that C# is a IDE, while others claim its a actual Framework. How ever, I am 50/50, not sure which is actually true.

An IDE is basically a text editor specifically made for writing code, let's put it that way.

~ Left HF For Good ~ - August 18, 2016, 7:43 pm

(Aug 18, 2016 07:37 PM)Faith Connors Wrote: [\[CITE\]](#) An IDE is basically a text editor specifically made for writing code, let's put it that way.

So, wouldn't java be a IDE for example? I mean I know what they are but, I am just curious. I would call java an IDE.

Faith Connors - August 18, 2016, 7:48 pm

(Aug 18, 2016 07:43 PM)ChilledWhiteHat Wrote: [\[CITE\]](#) So, wouldn't java be a IDE for example? I mean I know what they are but, I am just curious. I would call java an IDE.

Java is a programming language lmao. They have a runtime environment which provides a framework for running their applications, though, if that's what you mean.

The IDE primarily used for Java development is Eclipse. However IntelliJ IDEA is gaining popularity because it also supports .NET applications out of the box and has a decompiler.

~ Left HF For Good ~ - August 18, 2016, 7:59 pm

(Aug 18, 2016 07:48 PM)Faith Connors Wrote: [\[CITE\]](#) Java is a programming language lmao. They have a runtime environment which provides a framework for running their applications, though, if that's what you mean.

The IDE primarily used for Java development is Eclipse. However IntelliJ IDEA is gaining popularity

because it also supports .NET applications out of the box and has a decompiler.

Wow, I feel like a complete dumb ass. I am not big in Java, have only spent like 2 days coding in it. Anyways, thanks for clearing it up for me.

**sweeptheleg** - August 18, 2016, 7:59 pm

Great thread, OP. Nice to find this all in one place.

**lakyet1984** - August 20, 2016, 7:28 pm

thanks if really help ..nice

**Hamekaze** - August 21, 2016, 5:09 pm

(Aug 18, 2016 07:43 PM)Ominous Wrote: [\[CITE\]](#) So, wouldn't java be a IDE for example? I mean I know what they are but, I am just curious. I would call java an IDE.

Not sure what planet you live on.

Java is a programming language, whereas Eclipse is an IDE for Java.

**Ltyrosine** - August 21, 2016, 5:30 pm

Ill be back but this is an important thread for me good info here

**Pharaoh1100** - August 21, 2016, 5:32 pm

I'm going to read the whole OP, just posting in advance to give my appreciation to OP

**VedItSource** - November 16, 2016, 7:50 pm

best and most comprehensive introduction I have ever read about these different languages. I BARELY knew the difference even after attending college.

Thank you.

**Faith Connors** - November 16, 2016, 7:53 pm

(Nov 16, 2016 08:50 PM)VedItSource Wrote: [\[CITE\]](#) best and most comprehensive introduction I have ever read about these different languages. I BARELY knew the difference even after attending college.

Thank you.

Kinda surprised that you actually found this thread, it's kinda old.

I was actually thinking of rewriting it as I didn't do a great job of explaining stuff.

But I'm glad I could help nonetheless, thanks for reading.

**Ravioli** - November 16, 2016, 10:07 pm

This was a really good read, I'd give you rep if I could lmao. Cleared up a few misconceptions about SQL for me. Definitely going to link new members this when they ask. ?

**ffatsehtnahtretrams** - December 10, 2016, 11:55 am

(Nov 16, 2016 08:53 PM)Faith Connors Wrote: [\[CITE\]](#) I was actually thinking of rewriting it as I didn't do a great job of explaining stuff.

Please, do. I fully support you, incase you decide to do that.

**retramsnahtetffats** - December 14, 2016, 12:08 pm

I have a challenge for you. (Except the nwo\*\*\* one)

The challenge is "Advantages and disadvantages of programming in Linux."

Not pushing you or anything, the earlier-mentioned challenge is already in process (.pdf). Whenever you can, I just want a good source for comprehensive and logic information.

I would very much want to see your work on this topic.

Faith Connors - December 14, 2016, 12:38 pm

(Dec 14, 2016 01:08 PM)retramsnahteffats Wrote: [\[CITE\]](#) I have a challenge for you. (Except the n wo\*\*\* one)

The challenge is "Advantages and disadvantages of programming in Linux."

Not pushing you or anything, the earlier-mentioned challenge is already in process (.pdf). Whenever you can, I just want a good source for comprehensive and logic information.

I would very much want to see your work on this topic.

Honestly I don't program on Windows and keep everything related to that on Linux lmao

But I could give it a shot, got a lot of friends interested in comp sci who use a variety of OSes. One guy I know just got accepted to Waterloo compsci