**PROGRAMMING LANGUAGE LIST FROM EASY TO HARD**

A lot of people say “the hardest programming language is your first,” yeah that can be true for first few days, but there are some that cab remain the toughest programming language for a long time.

There are some difficult programming language that are not meant for everyone. Any language can be easy or hard depending on your educational background and also your limits. It mainly depends on your logic that you use in your coding to get the output and also the time you spend coding, obviously you need the practice.

Here is the list of programming languages on the basis of its difficulty. Its starts with the easiest and ends with the hardest.

1. **Go**

Go is a much simpler language than even Python or JavaScript. It is very similar to Python in terms of learning complexity, although library design is generally somewhat more consistent. Might be a bit harder to learn though, mostly due to masquerading as a low-level language when it’s really not, so there’s a bit of a disconnect sometimes. It is also easy to deploy a Go app running on Google Cloud Platform. The only two areas where you might have some problems with golang are interfaces and the concurrency features: goroutines and channels. Many beginners find concurrency hard to understand. Go was also ranked one of the most intuitive programming languages

EXAMPLE-

package main

import "fmt"

// this is a comment

func main() {

fmt.Println("Hello World")

}

1. **Python**

Python is an easy, general-purpose coding language designed with readability in mind. As one of the easiest programming languages, it’s often used as an introductory language for college students beginning their Computer Science study. It isn’t always consistent. So, there’s simply a lot to learn

EXAMPLE-

print('Hello, world!')

1. **Java**

Java has arguably a bigger collection of libraries to be aware of it, can be tricky. As you learn Java programming, you'll encounter some simple concepts like variables and functions, but there are also more abstract, complex ones like objects, bringing inheritance, and polymorphism that can be difficult to understand.

EXAMPLE

class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello World!");

}

}

1. **PHP**

Learning PHP will be easy or challenging depending on how you approach learning the language itself. Naturally, if you have a background in programming, you'll probably have a head-start in the process. PHP is much less consistent than Python. Newer versions are better, but are backwards-compatible, which means there’s still a lot of frankly weird stuff in the language, and a lot of horrifying code floating around. PHP is regarded as an easy programming language to master for people just starting to learn to program. As with any programming language, PHP has rules of coding, abbreviations, and logarithms.

EXAMPLE-

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My first PHP page</h1>  
  
<?php  
echo "Hello World!";  
?>  
  
</body>  
</html>

1. **JavaScript**

The base language is really simple, but it also includes more complex behaviour, like securely sending and retrieving information to another server. Also, lately, a lot of unnecessary syntax was added in new standards, so you’ll have to learn that as well. The use of HTML, CSS, and JavaScript together in this way is often referred to as front end development.

EXAMPLE-

<!DOCTYPE HTML>

<html>

<body>

<p>Before the script...</p>

<script>

alert( 'Hello, world!' );

</script>

<p>...After the script.</p>

</body>

</html>

1. **C**

C is hard, because computers are hard, basically. Once you are thorough with the syntax, it is just the approach you make for the problem counts. Remembering the syntax is not such a big issue in C. It’s rather easy if we’re talking about learning the language, but there’s this thing called “undefined behavior” that complicates matters a lot. If you’re using any of which is undefined, compiler’s free to do whatever it wants, and in most cases it will not notify you about that. It is fairly straight forward and easily readable to a beginner.

EXAMPLE-

#include <stdio.h>

int main() {

// printf() displays the string inside quotation

printf("Hello, World!");

return 0;

}

1. **Rust**

Rust code compiles to machine-native instructions, and the syntax and idioms around memory management—lifetimes and borrowing—make it difficult for memory-unsafe code to compile at all. You will fight with compiler a lot, because Rust tries to give you a lot of static guarantees. There are some unexpected things you might run into, but for the most part, it’s rather consistent and intuitive.

EXAMPLE-

fn main() {

println!("Hello, world!");

1. **Haskell**

Haskell is difficult because it's a functional language and most people are used to imperative OOP style programming. It is due to its terse syntax, abstractness, purity, and its community's love of one letter identifier names. Haskell is much easier if you have a math background and little programming experience. Accomplished programmers often struggle with Haskell because they need to break some preconceived stereotypes first. Haskell is full of very abstract concepts that might be hard to grasp at first. Google “monads” for an example. An intimidating list of language extensions complicates matters a bit as well.

Example-

main :: IO ()

main = putStrLn "Hello, World!"

1. **C++**

C++ is a difficult language to learn because of its huge feature size. C++ is basically three distinct languages you need to be aware of: preprocessor, C++ itself, and template metalanguage; all three use different syntax; all three have their own things. There are a lot of frankly weird things in C++ standard as well. As a consequence, C++ is full of often unintuitive rules you need to know. Add to that a huge standard library, that horrible thing called “undefined behavior” inherited from C, and quite a bit of new stuff being introduced with each new standard, and you get yourself a language that’s unintentionally very hard to learn. Memory management is difficult in C but even C programmers find the C++ curve to be a bit steep.

EXAMPLE

#include <iostream>

int main() {

std::cout << "Hello World!";

return 0;

}

**10.ASSEMBLY**

Assembly language is not difficult, but it's fussy. It is hard to read and understand. You have to be very closely aware of the hardware you're running on, especially the CPU and its registers. It is hard to debug and hard to maintain. If you need math functions, other than the simple add subtract multiply and divide that come with the CPU, you have to write them yourself. Programming in assembly language programming is time consuming. Improved compiler technology has eliminated the need for assembly language. There is no contest here assembly is the hardest.

EXAMPLE-

global \_main

extern \_GetStdHandle@4

extern \_WriteFile@20

extern \_ExitProcess@4

section .text

\_main:

; DWORD bytes;

mov ebp, esp

sub esp, 4

; hStdOut = GetstdHandle( STD\_OUTPUT\_HANDLE)

push -11

call \_GetStdHandle@4

mov ebx, eax

; WriteFile( hstdOut, message, length(message), &bytes, 0);

push 0

lea eax, [ebp-4]

push eax

push (message\_end - message)

push message

push ebx

call \_WriteFile@20

; ExitProcess(0)

push 0

call \_ExitProcess@4

; never here

hlt

message:

db 'Hello, World', 10

message\_end:

The list mentioned above is completely on our basis. It can be different for other people, it completely depends on your mindset if you think “Man!! This is hard” for every language all of these languages can become your enemy. So, change these mindset and work hard. Happy coding.