Introduction to Programming Languages

What is a **Programming language?**

A **programming language** is a computer language programmers use to develop software programs, scripts, or other sets of instructions for computers to execute.

Although many languages share similarities, each has its own syntax. Once a programmer learns the languages rules, syntax, and structure, they write the source code in a text editor or IDE. Then, the programmer often compiles the code into machine language that can be understood by the computer. Scripting languages, which do not require a compiler, use an interpreter to execute the script.

Types of programming languages ,

Each of the different programming languages mentioned in the next section can be broken into one or more of the following types (paradigms) of languages.

- High-level (most common) / low-level

  Sometimes abbreviated as **HLL**, a **high-level language** is a computer programming language that isn't limited by the computer, designed for a specific job, and is easier to understand. It is more like human language and less like machine language. However, for a computer to understand and run a program created with a high-level language, it must be compiled into machine language.

  The first high-level languages were introduced in the 1950s. Today, high-level languages are in widespread use. These include BASIC, C, C++, COBOL, FORTRAN, Java, Pascal, Perl, PHP, Python, Ruby, and Visual Basic.

  A **low-level language** is a programming language that provides little or no abstraction of programming concepts and is very close to writing actual machine instructions. Two examples of low-level languages are assembly and machine code.

- General-purpose / domain-specific

  A **general-purpose language** is a programming language that is capable of creating all types of programs. For example, C is a good example of a general-purpose language.

A language that is not a general-purposed language is called a **DSL** (**domain-specific language**). For example, HTML, Logo, MATLAB are examples of domain-specific languages

• Object-oriented / concurrent

Coined by Alan Kay, **object-oriented programming**, also known as **OOP** or **OO programming**, is a programming language paradigm. In an object-oriented program, the code can be structured as reusable components, some of which may share properties or behaviors.

Object-oriented programming can improve the developer's ability to quickly prototype software, extend existing functionality, refactor the code, and maintain it as it's developed.

List of computer programming languages,

Today, there are thousands of different programming languages. The following section contains an index of the different programming and scripting languages currently listed on our site. Clicking on any of the following languages displays an explanation and examples of that language.

A-C	**D-K**	**L-Q**	**R-Z**
ActionScript	D	LeLisp	R
ALGOL	DarkBASIC	Lisp	Racket
Ada	Dart Datalog	LiveScript	Reia
AIML *	dBASE	LOGO	RPG
Altair	Dylan	Lua	Ruby
BASIC	EuLisp	MACLISP	Rust
Assembly	Elixir F F#	Matlab	Scala
AutoHotkey	FORTRAN	Metro	Scheme
Babel	FoxPro	MUMPS	Scratch
BASIC	Franz Lisp	Nim	SGML *
Batch file	GameMaker	Objective-C	Simula
BCPL	Go	OCaml	Smalltalk
BeanShell	GW	Pascal	SPL
Brooks	Basic	Perl	SQL *
C	Haskell	PHP	Stanford
C#	HDML *	Pick	LISP Swift
C++	HTML *	PureBasic	Tcl
COBOL	InterLisp	Python	Turbo
CoffeeScript	ksh	Prolog	Pascal
Common	Java	QBasic	True
Lisp CPL	JavaScript		BASIC
CSS *	JCL		VHDL
Curl			Visual Basic

Application and program development

Application and program development involves programs you work with on a daily basis. If you are interested in developing a program, consider the following languages:

- C
- C#
- C++
- D
- Java
- Swift
- Tcl
- Visual Basic

Artificial Intelligence development

Artificial intelligence or related fields involve creating the character interactions in computer games, portions of programs that make decisions, chatbots, and more. If you're interested in developing an AI, consider the following languages:

- AIML
- C
- C#
- C++
- Prolog
- Python

Database development

Database developers create and maintain databases. If you're interested in creating or maintaining a database, consider any of the following languages:

- DBASE
- MySQL
- SQL
- Visual Fire Pro

Computer drivers or other hardware development

Computer drivers and programming hardware interface support are a necessity for hardware functionality. If you're interested in developing drivers or software interfaces for hardware devices, consider the following languages:

- Assembly
- C

Internet and web page development

Internet and web page development are the essence of the Internet. Without developers, the Internet would not exist. If you're interested in creating web pages, Internet applications, or other Internet-related tasks, consider the following languages:

- HDML
- HTML
- Java
- JavaScript
- Perl
- PHP
- Python
- XML

Script development

Although it is not likely to become a career, knowing how to create and develop scripts can increase productivity for you or your company, saving you countless hours. If you're interested in developing scripts, consider the following languages:

- Auto Hotkey
- awk
- bash
- Batch file
- Perl
- Python

Difference between an Interpreter and a Compiler,

Interpreter	Compiler
1. Translates program one statement at a time.	1. Scans the entire program and translates it as a whole into machine code.
2. Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.	2. Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.
3. No Object Code is generated, hence are memory efficient.	3. Generates Object Code which further requires linking, hence requires more memory.
4. Programming languages like JavaScript, Python, Ruby use interpreters.	4. Programming languages like C, C++, Java use compilers.