Comparison between types of programming languages

What is the programming languages

It is a way of communication that humans understand because it is close human language convert inside the machine into 0 ,1

Types of programming languages

- Low level , High level
- Interpreted , Compiled
- Programming ,Scripted
- Open source , not open source
- Support OOP , not supporting OOP

1. Low level Vs High level

Low level definition :

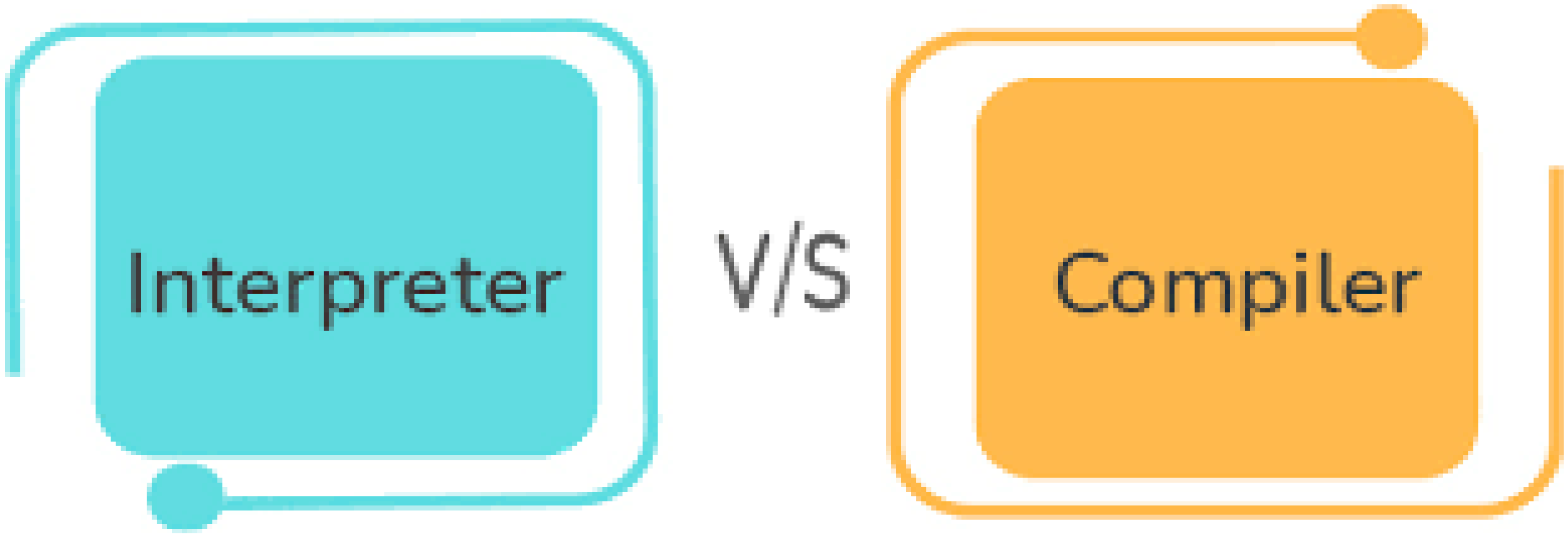
Low-level programming languages exist to communicate to computers and other machines in commands they can understand , ex , Machine language , Assembly Language .

High level definition :

High-level programming languages are easier for humans to learn, read, and comprehend. They are compiled or translated into a low-level language for computers to execute. This action allows programmers to skip many tedious and time-consuming steps within coding to build more complex code , ex , python , c++ , etc.. .

The difference between high level and low level languages:

It’s programmer friendly language	It is machine friendly language
It is easy to understand	It is tough to understand.
Debugging is easy.	Debugging is complex comparatively.
It is simple to maintain.	It is complex to maintain comparatively.
It is portable.	It is non-portable.
It can run on any platform.	It is machine-dependent.
It needs compiler or interpreter for translation.	It needs assembler for translation.
It is used widely for programming.	It is not commonly used now-a-days in programming.
High level language is less memory efficient.	Low level language is high memory efficient.



InterviewBit

2. Interpreted VS Compiled

Interpreted Definition :

An **Interpreter** directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code. Examples of interpreted languages are Perl, Python and Matlab.

Compiler Definition :

A compiler takes entire program and converts it into object code which is typically stored in a file. The object code is also referred as binary code and can be directly executed by the machine after linking. Examples of compiled programming languages are C and C++.

Aspect	**Compiler**	**Interpreter**
Process	Translates entire source code into machine code at once.	Translates and executes code line by line.
Execution	Generates an executable file that runs independently.	Executes code directly; requires the interpreter during runtime.
Speed	Faster execution since machine code is pre-generated.	Slower execution due to on-the-fly translation.
Error Handling	Reports all errors at once after compilation.	Stops and reports errors one at a time during execution.
Memory Usage	Uses less memory at runtime (only machine code is executed).	Uses more memory as both source code and interpreter are needed.
Use Case	Used in languages like C, C++, Java (with bytecode).	Used in languages like Python, JavaScript, Ruby.


SCRIPTING
VS
PROGRAMMING
© www.SoftwareTestingHelp.com

COMPARISON BETWEEN

Programming Language

A programming language is an organized way of communicating with a computer.

Traditional programming is based on low level languages.

General programming leads to closed software applications.

More code needs to be written.

&

Scripting Language

A scripting language is a programming language that support scripts.

Scripting prefers high level languages.

Scripting promotes open projects and is used for web applications.

Less coding needs in scripting.


OPEN SOURCE
VS
CLOSED SOURCE

OPEN SOURCE:

ACCESSIBLE CODE:
THE SOURCE CODE IS FREELY AVAILABLE TO THE PUBLIC.

TRANSPARENCY:
USERS CAN SEE AND MODIFY THE SOURCE CODE.

COMMUNITY COLLABORATION:
DEVELOPERS WORLDWIDE CAN CONTRIBUTE TO THE PROJECT.

NO LICENSING COSTS:
TYPICALLY, OPEN SOURCE SOFTWARE IS FREE TO USE.

CUSTOMIZABILITY:
USERS CAN MODIFY THE SOFTWARE TO SUIT THEIR NEEDS.

CLOSED SOURCE:

RESTRICTED CODE:
THE SOURCE CODE IS NOT PUBLICLY AVAILABLE.

LIMITED COLLABORATION:
DEVELOPMENT IS USUALLY DONE BY A SPECIFIC COMPANY.

OPAQUE:
USERS CANNOT VIEW OR MODIFY THE SOURCE CODE.

LICENSING COSTS:
USERS OFTEN NEED TO PAY FOR LICENSES.

LIMITED CUSTOMIZATION:
CUSTOMIZATION IS LIMITED TO WHAT THE SOFTWARE ALLOWS.

Difference Between

Functional Programming and OOP

OOP	Functional Programming
OOP is based on the concept of objects, instead of just functions and procedures	It emphasizes on the use of function calls as the primary programming construct
OOP follows the imperative programming model.	It's tightly connected to declarative programming
It Does not Support parallel programming	It Support parallel programming
Basic elements are objects and methods	Basic elements are variable and functions
OOP brings together data and its associated behavior in a single location	Data and its associated behavior are considered different entities and should be kept separate