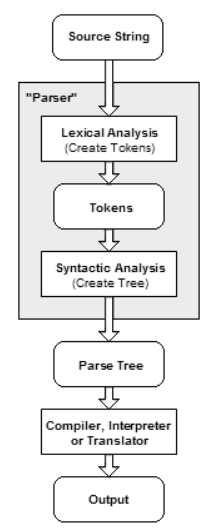
**JavaScript**

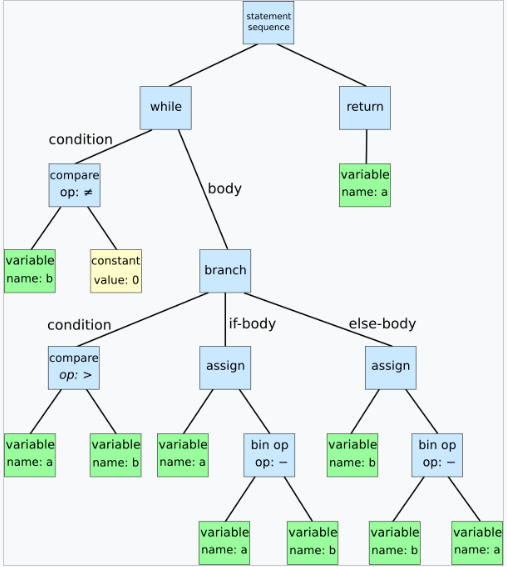
**Topic – 1 : Parser**

* The parser is responsible for breaking down the source code into its individual components, such as keywords, variables, and operators. It creates an AST that represents the structure of the code.



**Topic – 2 : ATS**

An **abstract syntax tree** (**AST**) is a data structure used in computer science to represent the structure of a program or code snippet. It is a tree representation of the  abstract syntactic structure of text (often source code) written in a formal language. Each node of the tree denotes a construct occurring in the text. It is sometimes called just a **syntax tree**.



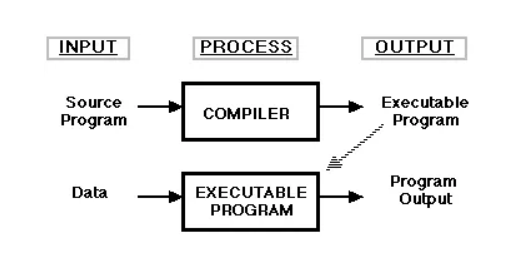
The syntax is abstract in the sense that it does not represent every detail appearing in the real syntax, but rather just the structural or content-related details. For instance, grouping parentheses are implicit in the tree structure, so these do not have to be represented as separate nodes. Likewise, a syntactic construct like an if-condition-then statement may be denoted by means of a single node with three branches.

This distinguishes abstract syntax trees from concrete syntax trees, traditionally designated parse trees. Parse trees are typically built by a parser during the source code translation and compiling process. Once built, additional information is added to the AST by means of subsequent processing, e.g., contextual analysis.

Abstract syntax trees are also used in program analysis and program transformation systems.

**Topic – 3 : Difference between compiler and interpreter**

**Compiler**

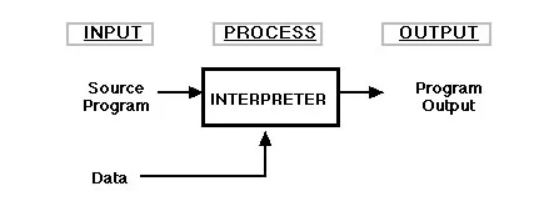
****

A Compiler is a computer program that translates computer code written in one programming language into another programming language. A Compiler takes the entire program and converts it into executable computer code. It takes an entire program because the computer only understands the program written in 0’s and 1’s and converts the executable program into machine code that is recognized by the computer. Examples of compiled programming languages are C and C++.

* “Compiler” is primarily used for programs that translate source code from a high-level programming language to a lower-level programming language.

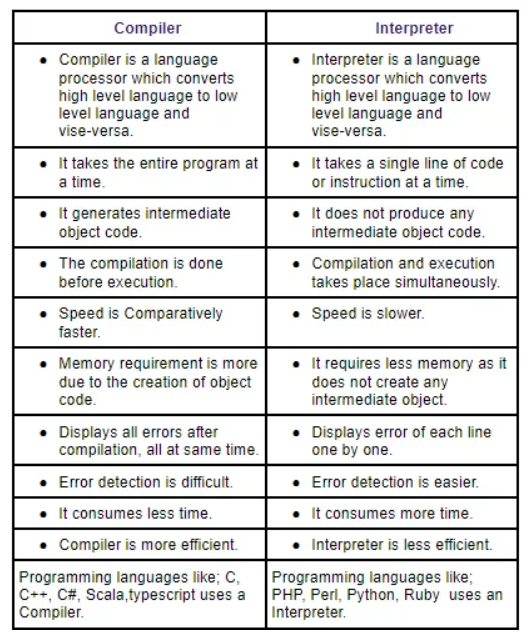
A Compiler is likely to perform many or all the operations: preprocessing, lexical analysis, parsing, semantic analysis, conversion of input programs to an intermediate representation, code optimizing and code generation.

**Interpreter**



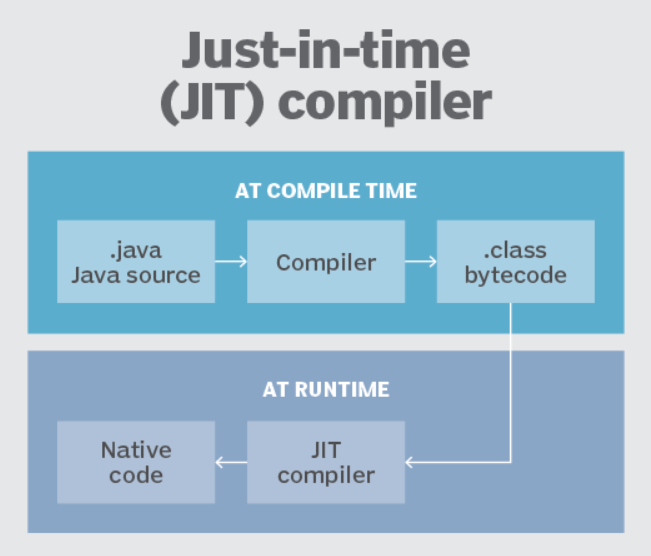
* An Interpreter is a computer program, which converts each high-level program **statement** into machine code. This include source code, pre-compiled code, and scripts.
* An Interpreter is defined as a machine program that directly executes a set of instructions without compiling them. Examples of interpreted languages are Perl, Python and Matlab.

Both **Compiler**and**Interpreter** do the same job which is converting high-level programming language to machine code. However, a compiler will convert the code into machine code **before the program run**. Interpreter covers code into machine code when **the program is run**.



**Topic – 4 : Explain JIT compiler**

A just-in-time (JIT) compiler is a program that turns [bytecode](https://www.techtarget.com/whatis/definition/bytecode) into instructions that can be sent directly to a computer's [processor](https://www.techtarget.com/whatis/definition/processor) (CPU). Typically, [compiler](https://www.techtarget.com/whatis/definition/compiler)s are key in deciding the speed of an application for developers and end users. Just-in-time compilers can be used for performance optimization to improve application runtime.



Just-in-time compilation is a method for improving the performance of interpreted programs. During execution the program may be compiled into native code to improve its performance. It is also known as dynamic compilation.

Dynamic compilation has some advantages over static compilation. When running Java or C# applications, the runtime environment can profile the application while it is being run. This allows for more optimized code to be generated. If the behavior of the application changes while it is running, the runtime environment can recompile the code.

Some of the disadvantages include startup delays and the overhead of compilation during runtime. To limit the overhead, many JIT compilers only compile the code paths that are frequently used.

**Topic – 5 : Source code Vs Machine code Vs Byte code**

**Source code**

A set of instructions written in a programming language, like Python, Java, C, or C++. Source code is the foundation for creating software, and it can be edited, customized, or developed further.

**Machine code**

A set of binary instructions made up of 1's and 0's, called bits, that are directly communicated to a computer's hardware via the CPU. Machine code is the lowest level of code, and it's made up of 8-bit combinations called bytes.

**Byte code**

A low-level representation of code that is generated from the compilation of source code. Byte code is made up of compact numeric codes, constants, and references. A virtual machine, like the Java Virtual Machine (JVM), processes byte code. Byte code is non-runnable and needs to be translated into machine code by an interpreter before it can be understood by a machine.

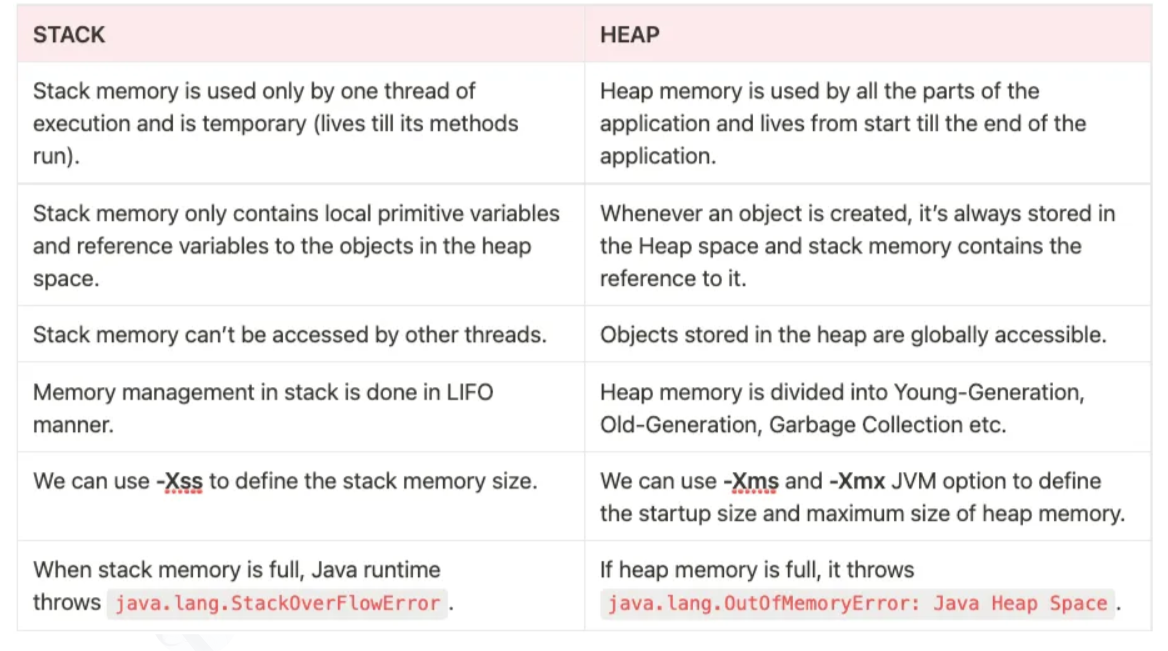
**Source Code and Byte Code**

| **S.NO.** | **Source Code** | **Byte Code** |
| --- | --- | --- |
| 01. | Source code is written by a human or programmer. | Byte code is not written by humans or programmers. |
| 02. | It is written by using some high-level programming language. | Byte code is an intermediate code between the source code and machine code. |
| 03. | It is the input to the compiler and it is translated by the compiler or other language translator. | It is the input to the interpreter |
| 04. | The source code is not directly understandable by the system/machine. | Byte code is executable by a virtual machine. |
| 05. | The source code may contain comments. | Byte code does not contain comments. |
| 06. | Source code is in the form of plain text similar to the English language. | Byte code is in the form of numeric codes and constants. |
| 07. | Source code is more understandable by humans. | Byte code is less understandable by humans. |
| 08. | Its speed is minimum than the byte code. | Its speed is maximum than the source code. |
| 09. | The performance of source code is less than byte code. | The performance of the byte code is more than the source code. |
| 10. | It is a high-level code. | It is an intermediate-level code. |

**Byte Code and Machine Code**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Byte Code** | **Machine Code** |
| 01. | Byte Code consisting of binary, hexadecimal, macro instructions like (new, add, swap, etc) and it is not directly understandable by the CPU. It is designed for efficient execution by software such as a virtual machine.intermediate-level | Machine code consisting of binary instructions that are directly understandable by the CPU. |
| 02. | Byte code is considered as the intermediate-level code. | Machine Code is considered as the low-level code. |
| 03. | Byte code is a non-runnable code generated after compilation of source code and it relies on an interpreter to get executed. | Machine code is a set of instructions in machine language or in binary format and it is directly executed by CPU. |
| 04. | Byte code is executed by the virtual machine then the Central Processing Unit. | Machine code is not executed by a virtual machine it is directly executed by CPU. |
| 05. | Byte code is less specific towards machine than the machine code. | Machine code is more specific towards machine than the byte code. |
| 06. | It is platform-independent as it is dependent on the virtual machine and the system having a virtual machine can be executed irrespective of the platform. | It is not platform independent because the object code of one platform can not be run on the different Operating System. Object varies depending upon system architecture and native instructions associated with the machine. |
| 07. | All the source code need not be converted into Machine code(or Object Code) for execution by CPU. Some source code written by any specific high-level language is converted into byte code then byte code to object code for execution by CPU. | All the source code must be converted into machine code before it is executed by the CPU. |

**Topic – 6 : Explain difference between heap and stack**

****

**Topic – 7 : Explain web API**

A web API, or application programming interface, is a set of rules that define how a client application can access a web server's services. Web APIs are a key part of the World Wide Web and are used in web development to connect web browsers and web servers.

* Web APIs and web services: All web services are APIs, but not all APIs are web services.
* REST API: A special type of web API that uses a standard architectural style.
* Web API components: Web APIs define the rules that client applications need to follow, including output formats, input data formats, access mechanisms, and underlying protocols.
* Web API flexibility: Web APIs are flexible, fast, secure, manageable, and easy to use.
* API management: API management platforms provide authentication, authorization, and provisioning support to streamline access to an API.