Programming languages and Development tools

A programming language is a set of instructions and syntax used to create software programs. Some of the key features of programming languages include:

1. **Syntax**: The specific rules and structure used to write code in a programming language.
2. **Data Types**: The type of values that can be stored in a program, such as numbers, strings, and booleans.
3. **Variables**: Named memory locations that can store values.
4. **Operators**: Symbols used to perform operations on values, such as addition, subtraction, and comparison.
5. **Control Structures**: Statements used to control the flow of a program, such as if-else statements, loops, and function calls.
6. **Libraries** and Frameworks: Collections of pre-written code that can be used to perform common tasks and speed up development.
7. **Paradigms**: The programming style or philosophy used in the language, such as procedural, object-oriented, or functional.

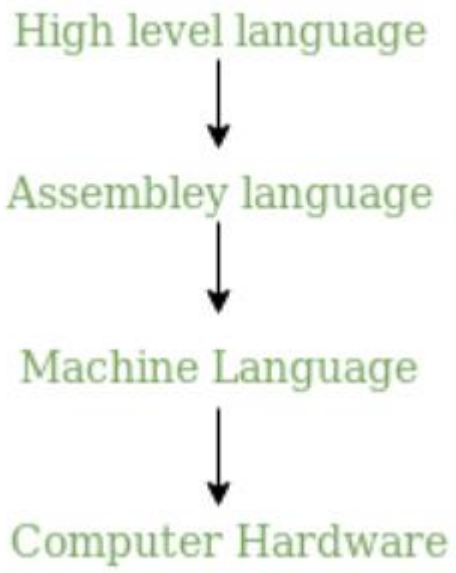
Examples of popular programming languages include Python, Java, C++, JavaScript, and Ruby. Each language has its own strengths and weaknesses and is suited for different types of projects.

Examples of popular programming languages include Python, Java, C++, JavaScript, and Ruby. Each language has its own strengths and weaknesses and is suited for different types of projects.

A programming language is a formal language that specifies a set of instructions for a computer to perform specific tasks. It's used to write software programs and applications, and to control and manipulate computer systems. There are many different programming languages, each with its own syntax, structure, and set of commands. Some of the most commonly used programming languages include Java, Python, C++, JavaScript, and C#. The choice of programming language depends on the specific requirements of a project, including the platform being used, the intended audience, and the desired outcome. Programming languages continue to evolve and change over time, with new languages being developed and older ones being updated to meet changing needs.

Hierarchy of Computer language

Between high-level language and machine language, there are assembly languages also called symbolic machine code. Assembly languages are particularly computer architecture specific. Utility program (**Assembler**) is used to convert assembly code into executable machine code. High Level Programming Language is portable but requires Interpretation or compiling to convert it into a machine language that is computer understood.



Characteristics of a programming Language –

- A programming language must be simple, easy to learn and use, have good readability, and be human recognizable.
- Abstraction is a must-have Characteristics for a programming language in which the ability to define the complex structure and then its degree of usability comes.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and its execution consumes little space in memory.
- A programming language should be well structured and documented so that it is suitable for application development.
- Necessary tools for the development, debugging, testing, maintenance of a program must be provided by a programming language.
- A programming language should provide a single environment known as Integrated Development Environment(IDE).

- A programming language must be consistent in terms of syntax and semantics.

Basic Terminologies in Programming Languages

- **Algorithm**: A step-by-step procedure for solving a problem or performing a task.
- **Variable**: A named storage location in memory that holds a value or data.
- **Data Type**: A classification that specifies what type of data a variable can hold, such as integer, string, or boolean.
- **Function**: A self-contained block of code that performs a specific task and can be called from other parts of the program.
- **Control Flow**: The order in which statements are executed in a program, including loops and conditional statements.
- **Syntax**: The set of rules that govern the structure and format of a programming language.
- **Comment**: A piece of text in a program that is ignored by the compiler or interpreter, used to add notes or explanations to the code.
- **Debugging**: The process of finding and fixing errors or bugs in a program.
- **IDE**: Integrated Development Environment, a software application that provides a comprehensive development environment for coding, debugging, and testing.
- **Operator**: A symbol or keyword that represents an action or operation to be performed on one or more values or variables, such as + (addition), – (subtraction), * (multiplication), and / (division).
- **Statement**: A single line or instruction in a program that performs a specific action or operation.

Software Development Tools

Two primary activities in computing are program development and the use of application software. Language processors and operating systems play an obvious role in these activities. A less obvious but vital role is played by programs that help in using or developing other programs. These programs perform various housekeeping tasks involved in program development and application usage. Their use provides convenience and improves a user's productivity.

- A software tool is a system program that suitably interfaces a program with other

- programs or human users in its environment. The tasks performed by software tools cover a wide spectrum from mundane tasks of interfacing to sophisticated tasks that improve the effectiveness of a user's activity.
- Examples of software tools at the two ends of this spectrum are a tool that reformats data so that they can be used readily by an existing program, and a debug monitor that provides "intelligent' assistance for debugging a program, respectively.

What is Software Tool?

A software tool is a system program that interfaces a program with the entity generating its input data or interfaces the results of a program with the entity consuming them.

- It is a set of computer programs used by developers to create, maintain, debug, or support other applications and programs.
- It can be code editors, language libraries, debuggers, performance analysis tools, GUI designers, etc.
- There are some factors that need to be considered before selecting a software tool like the usefulness of the tool, company criteria, integration of one tool with another, etc.
- Software tools assist in all the activities of the software lifecycle.
- These are used to accomplish and investigate the business processes, document the development process, and optimize all the processes.
- These help developers easily maintain the workflow of the project.

Evolution of Software Development Tools

The evolution of software development tools has been driven by the need to improve efficiency, accuracy, and collaboration in the software development process. As software development has become more complex and specialized, tools have been developed to address specific needs and challenges.

Software development tools have evolved significantly over the past several decades, as advances in technology have led to new capabilities and improved performance. Here are some key milestones in the evolution of software development tools:

1. **The 1960s:** The first computer programming languages, such as FORTRAN and COBOL, are developed, along with early text editors and compilers.
2. **The 1970s:** The rise of personal computers and the development of new programming languages, such as C and Pascal, lead to the creation of more advanced software development tools, including integrated development environments (IDEs) and debuggers.
3. **The 1980s:** The introduction of graphical user interfaces (GUIs) and the widespread adoption of personal computers leads to the development of tools that are easier to use and more visually appealing.The
4. **The 1990s:** The emergence of the World Wide Web and the development of new programming languages, such as Java and Python, led to the creation of tools for web development, including web browsers and HTML editors.
5. **The 2000s:** The rise of mobile devices and the development of new platforms, such as iOS and Android, lead to the creation of tools for mobile app development, including IDEs and emulators.
6. **The 2010s:** The proliferation of cloud computing and the emergence of new programming languages, such as Go and Swift, lead to the development of tools for cloud development, including cloud IDEs and containerization tools.
7. **The 2020s:** The rise of artificial intelligence and machine learning leads to the development of tools for developing and deploying machine learning models, such as notebooks and model serving platforms.

Common Software Development Tools Types

Software tools are programs or applications that assist in the development, testing, and maintenance of software. Some common software development tools include:

1. **Integrated Development Environments (IDEs):** IDEs are specialized software tools that provide a complete environment for software development, including a text editor, compiler, debugger, and other tools.
2. **Source Code Management (SCM) tools:** SCM tools allow developers to manage and track changes to their source code, as well as collaborate with other team members.
3. **Debugging tools:** Debugging tools allow developers to identify and fix errors in their code.

4. **Test automation tools:** Test automation tools allow developers to automate the testing process, making it more efficient and accurate.
5. **Project management tools:** Project management tools help developers to plan, track, and manage their software development projects.
6. **Collaboration tools:** Collaboration tools allow developers to communicate and work with other team members remotely.
7. **Static code analysis tools:** Static code analysis tools analyze source code for potential issues, such as security vulnerabilities or coding errors.
8. **Performance analysis tools:** Performance analysis tools help developers to identify and optimize the performance of their software.
9. **Documentation tools:** Documentation tools help developers to create and maintain documentation for their software projects.
10. **Configuration management tools:** Configuration management tools help developers manage and track software configuration changes.

Why do Software Development Tools Matter?

Below are some of the reasons for using software development tools:

- **Increases productivity:** Software development tools matter because they help developers to be more productive, efficient, and accurate. They also help to ensure the quality and reliability of the software being developed.
- **Helps to create quality software efficiently:** Software development tools are important because they help developers create high-quality software more efficiently and effectively. These tools can help with tasks such as writing and organizing code, debugging and testing code, and collaborating with other team members.
- **Saves time and effort:** Good software development tools can save time and effort by automating repetitive tasks and providing helpful features such as code completion, refactoring, and debugging tools. This can allow developers to focus on more important tasks, such as solving complex problems and implementing new features.
- **Helps to write reliable and maintainable code:** Software development tools can help developers write more reliable and maintainable code by providing features such as static analysis, unit testing, and integration testing. This can help prevent bugs and other

issues from being introduced into the codebase and can make it easier to catch and fix problems when they do occur.

- **Supports team collaboration:** Many software development tools include features that support team collaboration, such as version control systems, code review tools, and project management tools. These tools can help teams work together more effectively, by allowing them to track changes to the codebase, coordinate their work, and share ideas and feedback.
- **Range of tools provides variety:** Different software development tools are better suited to different types of projects, programming languages, and platforms. Having a range of tools available can allow developers to choose the best tool for the job, and can make it easier to adapt to changing needs or requirements.

Where are Development Tools found?

Software development tools can be found in a variety of places, including online marketplaces, software development platform websites, and as standalone applications that can be downloaded and installed on a local machine.

- **Online marketplaces:** Some popular online marketplaces for software development tools include the Apple App Store, Google Play Store, and Microsoft Store. These marketplaces offer a wide range of tools, including IDEs, text editors, version control systems, and more.
- **Software development platform websites:** Software development platform websites, such as GitHub, GitLab, and Bitbucket, also offer a range of tools that are specifically designed for use with their platform. These tools may include version control systems, collaboration tools, and integrations with other software development tools.
- **Free or open-source tool:** Standalone software development tools can also be downloaded and installed on a local machine. These tools may be available as free or open-source software, or they may require a purchase or subscription. Some popular standalone software development tools include IDEs, text editors, and version control systems..

10 Best Software Development Tools

Below are some of the software development tools:

1. Git: A version control system that allows developers to track changes to their code and collaborate with other team members. Features include branching, merging, and merging conflicts.

2. IntelliJ IDEA: A popular integrated development environment (IDE) for Java development, with support for a wide range of programming languages and frameworks. Features include code completion, refactoring, and debugging tools.

3. Eclipse: Another popular IDE, with a focus on Java development but support for a range of other languages as well. Features include code completion, refactoring, and debugging tools.

4. Visual Studio: A comprehensive IDE from Microsoft, with support for a wide range of programming languages and platforms. Features include code completion, refactoring, debugging, and integration with other Microsoft tools.

5. PyCharm: An IDE specifically designed for Python development, with support for scientific computing and data science tools. Features include code completion, debugging, and integration with version control systems.

6. Xcode: The IDE for Apple's platforms, including macOS, iOS, iPadOS, watchOS, and tvOS. Features include code completion, debugging, and integration with Apple's development tools.

7. Android Studio: The official IDE for Android development, with support for building, testing, and debugging Android apps. Features include a visual layout editor, emulator, and integration with Google's developer tools.

8. Node.js: A JavaScript runtime built on Chrome's V8 JavaScript engine, allowing developers to run JavaScript on the server side. Features include support for asynchronous programming and a large ecosystem of open-source libraries.

9. npm: The default package manager for the Node.js runtime, allowing developers to easily install and manage third-party libraries and dependencies. Features include version management and automatic dependency resolution.

10. Gradle: A build automation tool for Java and other programming languages, allowing developers to automate the process of building, testing, and deploying their code. Features include support for multiple languages and platforms and integration with a range of tools.

Coding Standards and Guidelines

Different modules specified in the design document are coded in the Coding phase according to the module specification. The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test this code.

Good software development organizations want their programmers to maintain to some well-defined and standard style of coding called coding standards. They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop. It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

Purpose of Having Coding Standards:

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

1. **Limited use of globals:**
   These rules tell about which types of data that can be declared global and the data that can't be.

2. **Standard headers for different modules:**
   For better understanding and maintenance of the code, the header of different modules should follow some standard format and information. The header format must contain below things that is being used in various companies:
   - Name of the module
   - Date of module creation
   - Author of the module
   - Modification history
   - Synopsis of the module about what the module does
   - Different functions supported in the module along with their input output parameters
   - Global variables accessed or modified by the module

3. **Naming conventions for local variables, global variables, constants and functions:**
   Some of the naming conventions are given below:
   - Meaningful and understandable variables name helps anyone to understand the reason of using it.
   - Local variables should be named using camel case lettering starting with small letter (e.g. **localData**) whereas Global variables names should start with a capital letter

(e.g. **GlobalData**). Constant names should be formed using capital letters only (e.g. **CONSDATA**).

- It is better to avoid the use of digits in variable names.
- The names of the function should be written in camel case starting with small letters.
- The name of the function must describe the reason of using the function clearly and briefly.

4. **Indentation:**
   Proper indentation is very important to increase the readability of the code. For making the code readable, programmers should use White spaces properly. Some of the spacing conventions are given below:
   - There must be a space after giving a comma between two function arguments.
   - Each nested block should be properly indented and spaced.
   - Proper Indentation should be there at the beginning and at the end of each block in the program.
   - All braces should start from a new line and the code following the end of braces also start from a new line.

5. Avoid using a coding style that is too difficult to understand:
Code should be easily understandable. The complex code makes maintenance and debugging difficult and expensive.

6. Avoid using an identifier for multiple purposes:
Each variable should be given a descriptive and meaningful name indicating the reason behind using it. This is not possible if an identifier is used for multiple purposes and thus it can lead to confusion to the reader. Moreover, it leads to more difficulty during future enhancements.

7. Code should be well documented:
The code should be properly commented for understanding easily. Comments regarding the statements increase the understandability of the code.

8. Length of functions should not be very large:
Lengthy functions are very difficult to understand. That's why functions should be small enough to carry out small work and lengthy functions should be broken into small ones for completing small tasks.

9. Try not to use GOTO statement:
GOTO statement makes the program unstructured, thus it reduces the understandability of the program and also debugging becomes difficult.