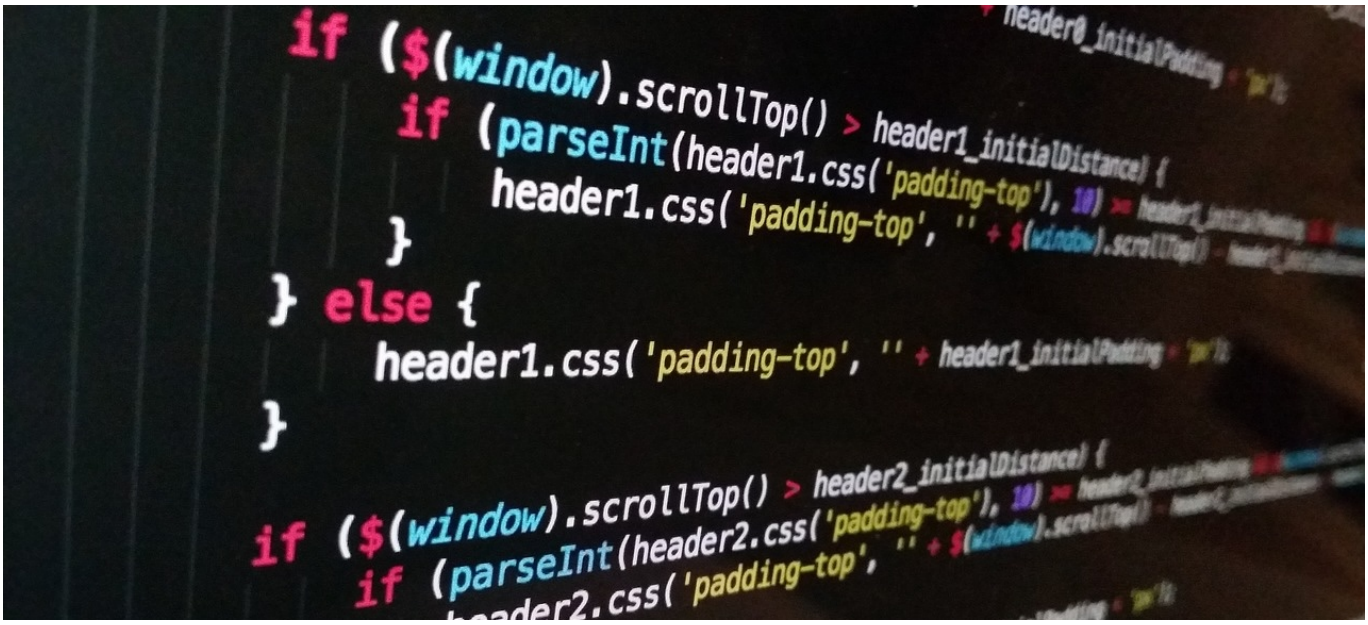
1. Programming in High-Level Languages



Introduction to Programming

Programming is the process of giving instructions to a computer to perform specific tasks. A computer understands only machine language, which consists of binary numbers (0s and 1s). However, humans write programs in high-level programming languages, which are easier to read and understand. These languages are translated into machine language by compilers or interpreters.

High-Level Languages

High-level programming languages are designed to be easy for humans to read and write. Some examples of high-level languages are Python, Java, and C++. These languages use English-like words and mathematical symbols to express instructions, making them more accessible than low-level languages like assembly or machine code.

- **Python**: A popular language known for its simple syntax and wide usage in fields like web development, data science, and artificial intelligence.
- **Java**: A widely-used language for building large, complex applications, especially for Android development.
- **C++**: A powerful language often used in game development and systems programming.

Types of Programming Languages

There are different types of programming languages based on their use and the way they interact with the computer:

1. **Procedural Programming Languages**: These languages focus on the sequence of actions or steps the computer must take to accomplish a task. The most common procedural language is C.
2. **Object-Oriented Programming (OOP) Languages**: These languages focus on objects, which are collections of data and methods that operate on that data. Java and Python are examples of OOP languages.
3. **Functional Programming Languages**: These languages treat computation as the evaluation of mathematical functions and avoid changing-state and mutable data. Haskell is an example of a functional language.

Syntax, Semantics, and Code Execution

- **Syntax** refers to the rules that define the structure of valid code in a programming language. It is like grammar in a language. For example, in Python, the syntax for defining a variable is:

  ```
  x = 5
  ```

  If you forget to write the correct syntax, the program will produce an error.

- **Semantics** refers to the meaning of the code. It is the logic that explains what each part of the code does. For example, in the above line of Python code, the syntax is correct, and the semantics tell us that the variable `x` is being assigned the value 5.

A good programmer needs to understand both syntax and semantics to write code that works as expected.

Compilers and Interpreters

High-level programming languages need to be translated into machine language so that a computer can understand and execute them. This translation is done using either a **compiler** or an **interpreter**:

- **Compiler**: A compiler translates the entire source code into machine code before execution. This results in a standalone executable file that can run on a computer. Examples of compiled languages include C, C++, and Go.
- **Interpreter**: An interpreter translates and executes code line by line. This allows for immediate execution but may be slower than compiled programs. Examples of interpreted languages include Python, JavaScript, and Ruby.

Some languages, like Java, use a combination of both: Java code is first compiled into an intermediate form called bytecode, which is then executed by an interpreter called the Java Virtual Machine (JVM).

"Hello, World!" in Different Programming Languages

The following examples show how to print "Hello, World!" in different programming languages. While most of these are high-level languages, assembly (or assembler) is a low-level language that is closer to machine code and directly interacts with the computer's hardware.

The "Hello, World!" program is a simple example used to demonstrate the syntax of a programming language. Below are examples of how to write "Hello, World!" in different languages:

- **Python**:

```
print("Hello, World!")
```

- **C**:

```
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

- **C++**:

```
#include <iostream>
using namespace std;
```

```cpp
int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

- **C#**:

```csharp
using System;
class Program {
    static void Main() {
        Console.WriteLine("Hello, World!");
    }
}
```

- **Java**:

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

- **JavaScript**:

```javascript
console.log("Hello, World!");
```

- **Go**:

```go
package main
import "fmt"
func main() {
    fmt.Println("Hello, World!")
}
```

- **Swift**:

```swift
print("Hello, World!")
```

- **Ruby**:

```ruby
puts "Hello, World!"
```

- **Assembly (x86 NASM syntax)**:

```
section .data
    hello db "Hello, World!",0

section .text
    global _start

_start:
    mov edx, 13      ; Message length
    mov ecx, hello  ; Message to print
    mov ebx, 1       ; File descriptor (stdout)
    mov eax, 4       ; System call number (sys_write)
    int 0x80         ; Call kernel

    mov eax, 1       ; System call number (sys_exit)
    xor ebx, ebx     ; Exit status 0
    int 0x80         ; Call kernel
```