1.Fundamentals of Programming

Understanding the Basics of Computers and Programming

Most of us have just started our journey in the programming world and are absolute beginners. While learning, many fail to understand programming because they directly start to learn and jump into the code, finding it hard to maintain a consistent approach. The major reason for leaving halfway is often the lack of understanding of the basics and fundamentals.

Strong fundamentals and basics are crucial to learning and grasping any programming language quickly. So, we'll start from the very basics and then move on to the fundamentals of programming.

Basics of Computers

What is a computer? What are its components?

Casually, we might say that a computer consists of a CPU or processor, a monitor, a mouse, RAM, a hard disk, and a keyboard.

In essence, your mobile phone is a computer, a laptop is a computer, and a desktop is also a computer. The main and major component of a computer is the processor, also called a microprocessor, referred to as the CPU (Central Processing Unit).

Whenever we perform any task on a computer, like playing a video file, listening to music, writing a blog, or preparing sheets for annual records, it is directly linked to the processor, which is responsible for executing these tasks.

The Processor

What is a processor?

- The processor is made up of a technology called **semiconductor technology**, which is known for its fast execution capabilities.
- Semiconductor technology involves the use of transistors and capacitors in its manufacturing process.

Types of Transistors:

1. NPN
2. PNP

These transistors operate with two voltage levels: low voltage (0V) and high voltage (5V). In the software world, this low and high voltage can be represented in binary form as 0s and 1s, known as binary language.

Understanding the Processor:

- The processor is essentially a "dumb" device that can't perform any tasks without instructions. We need to assign tasks and instruct them through communication.
- Just as we communicate in our native language (e.g., English), we need to communicate with the machine in its own language, known as **machine-level language (MLL)**.

Our main objective is to write or communicate in a way that the machine understands, whether in binary or another form it comprehends. These instructions, when grouped together, form a program.

Using MLL, the processor understands our instructions and provides the desired output. However, writing and understanding binary code can be challenging for humans. For small operations, it might be manageable, but for larger programs, it becomes impractical.

Assembly Level Language (ALL):

- To simplify the process, mnemonics were introduced for operations like ADD (add) and SUB (subtract).
- This usage of mnemonics is called **Assembly Level Language (ALL)**. It is somewhat human-friendly but still requires conversion to binary language for the processor to understand.
- An **assembler** is system software that converts ALL into MLL.

High-Level Language (HLL):

- In the 1960s, researchers developed another approach using symbols (+, -, /, x) and English terms (print, scan, if, else, etc.).
- This combination of symbols and English terms is called **High-Level Language (HLL)**.
- HLL is preferred and more human-friendly compared to MLL and ALL. However, the processor still only understands binary code.

Compiler:

- To bridge this gap, a system software called the **compiler** was developed in the 1960s.
- The compiler converts HLL into MLL, making it understandable for the processor.
- Since then, compilers have been used for popular languages like C, C++, Java, and many more.

So, moving on to the next to understand the working and how these programs are stored behind the scenes, we will learn in the next chapter.

Till then Stay curious and happy learning.

☺