Why Do We Need Programming Languages

Most computers rely on a CPU, and a CPU can understand a specific set of instructions in the form of ones and zeros.

Therefore, we could theoretically write code that tells the CPU what to do by stringing together long sequences of ones and zeros in a form the CPU understands. Instructions written in binary form like this are called **machine code**.

Sounds horrible to work with, doesn't it?

A **programming language** provides a set of human-readable keywords, statements, and syntax rules that are much simpler for people to learn, debug, and work with.

Programming languages provide a means of bridging the gap between the way our human brains understand the world and the way computer brains (CPUs) understand the world.

Ultimately, the code that we write needs to be translated into the binary instructions (machine code) that the CPU understands.

Depending on the language you choose, we say that your code is either **compiled** or **interpreted** into machine code capable of being executed by your CPU. Most programming languages include a program called a **compiler** or an **interpreter** which performs this translation step.

COMPILER VS INTERPRETER

A compiler translates the entire source code in a single run. An interpreter translates the entire source code line by line. It consumes less time i.e., it is faster than an interpreter. It consumes much more time than the compiler i.e., it is slower than the compiler.