# 1 WHAT IS A SOFTWARE?

*Software is capable of performing many tasks, as opposed to hardware which can only perform mechanical tasks that they are designed for.*
*The software provides the means for accomplishing many different tasks with the same basic hardware.*

**Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries, and documentation. Software, when made for a specific requirement is called **software product**.

***Classes of Software***

- **System software** – helps run the computer hardware and computer system itself. System software includes operating systems, device drivers, diagnostic tools, and more. System software is almost always pre-installed on your computer.

- **Application software** – allows users to accomplish one or more tasks. It includes word processing, web browsing, and almost any other task for which you might install the software. (Some application software is pre-installed on most computer systems.)

- **Programming software** – a set of tools to aid developers in writing programs. The various tools available are compilers, linkers, debuggers, interpreters, and text editors.

***Basic Principles***

1. Software, commonly known as programs or apps, consists of all the instructions that tell the hardware how to perform a task.
2. These instructions come from a software developer in the form that will be accepted by the platform (operating system + CPU) that they are based on.
3. For example, a program that is designed for the Windows operating system will only work for that specific operating system. The compatibility of the software will vary as the design of the software and the operating system differ. Software that is designed for Windows 10 may experience a compatibility issue when running under Windows 11.
4. Software, in its most general sense, is a set of instructions or programs instructing a computer to do specific tasks. Software is a generic term used to describe computer programs, Scripts, applications, programs and a set of instructions are all terms often used to describe software.

## ***Software Evolution***

The process of developing a software product using software engineering principles and methods is referred to as software evolution. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.
Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished. Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and going one-on-one with requirements is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

## ***Laws in Software Evolution***

**Eight laws for software evolution**

- **Continuing change** – a software system must continue to adapt to the real world changes, else it becomes progressively less useful.

- **Increasing complexity** – a software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.

- **Conversation of familiarity** – the familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system.

- **Continuing growth** –for a system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.

- **Reducing quality** – a software system declines in quality unless rigorously maintained and adapted to a changing operational environment.

- **Feedback systems** – the software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.

- **Self-regulation** – a system evolution processes are self-regulating with the distribution of product and process measures close to normal.

- **Organizational stability** – the average effective global activity rate in an evolving system is invariant over the lifetime of the product.