Basic Concepts

What is Engineering?

Engineering is the application of science, math, and practical knowledge to design, build, and improve things like buildings, machines, and systems. Engineers solve real-world problems by applying principles from subjects like physics, chemistry, and biology to create useful and efficient solutions.

What is Software Engineering?

Software engineering is the process of designing, developing, testing, and maintaining software applications and systems. It involves using engineering principles and techniques to create software that meets user needs, works efficiently, and is reliable. Software engineering also involves key elements that help in managing and improving the software development process:

1. **Quality** – This focuses on improving the processes used to develop software, ensuring that the end product is of high quality and meets standards.
2. **Process** – A set framework is followed to guide the effective delivery of software. This includes making sure the software is developed on time and ensuring everything runs smoothly throughout the development lifecycle.
3. **Methods** – These are the specific techniques and strategies used to build the software. It’s about figuring out the "how-to" steps for creating the software, such as coding practices and development approaches.
4. **Tools** – These are automated or semi-automated tools that help with the implementation of processes and methods. For example, software development tools can assist in coding, testing, or managing projects, making the entire process more efficient.

What is the Software Crisis?

Software crisis is referred to as the inability to write efficient software programs within the required timeframe.
Some of the examples of software crisis are given below-

1. Software projects are not being delivered within the specified time.
2. Projects are over budget and provide unreliable software that is expensive to maintain.
3. Software applications are becoming obsolete.
4. Software becomes more complex when trying to scale them.

Why is Software Engineering required?

1. **Managing Complexity**: As software systems grow larger and more complicated, it can be hard to handle them. Software engineering provides methods, processes, and tools to make it easier to build, maintain, and scale these systems effectively.
2. **Ensuring Quality**: Software engineering helps ensure that software is reliable, secure, and works as expected. Through testing and quality practices, it reduces the risk of bugs, errors, and security issues.
3. **Meeting User Needs**: It ensures that software is designed to meet the needs of users, whether for consumer apps or business solutions, making sure the software is functional and user-friendly.

4. **Reducing Costs and Time**: With a structured approach, software engineering helps reduce unnecessary work and makes the development process more efficient. It helps deliver software on time and within budget, avoiding costly mistakes and delays.
5. **Handling Changes in Requirements**: Business needs and environments often change quickly. Software engineering allows for flexibility, providing systems that can adapt to changing requirements and enabling developers to respond effectively to these changes.
6. **Scalability**: Software engineering helps create systems that can grow and handle increasing demands. This ensures that software can continue to function well as more users or data are added.
7. **Adaptability**: It also helps in designing software that can adjust to new or changing needs over time, making it easier to keep the software relevant and efficient.
8. **Reducing Development Time**: By using established processes and tools, software engineering ensures that software is developed within set timelines, reducing delays and speeding up the overall development process.

What are the Attributes of Software Engineering?

1. **Functionality**: How well the software does what it's supposed to do. It should meet the needs of the users and work as expected.
2. **Reliability**: The software should work consistently without errors or crashes, even when used over time or in different conditions.
3. **Usability**: The software should be easy and intuitive for users to understand and use.
4. **Efficiency**: The software should use system resources like memory and processing power wisely, ensuring it runs smoothly and quickly.
5. **Maintainability**: The software should be easy to update and fix if something goes wrong. It should be simple for developers to make changes or improvements.
6. **Portability**: The software should be able to work on different devices, operating systems, or platforms without needing a lot of changes.
7. **Scalability**: The software should be able to handle more users, data, or tasks as needed, without slowing down or breaking.
8. **Security**: The software should protect user data and be safe from attacks or unauthorized access.
9. **Testability**: The software should be easy to test to make sure it works correctly and is free of errors.
10. **Reusability**: The software should have parts that can be used in other projects, saving time and effort.
11. **Flexibility**: The software should be able to adjust to changes in requirements or needs without needing to be completely rewritten.

What is Software?

Software is a set of instructions that tells a computer or device how to do certain tasks. Unlike hardware, which is the physical part of a computer, software is the part that runs on the computer and makes it work.
There are two main types of software:

1. **System Software**: This includes programs that help the computer run, like the operating system (e.g., Windows or macOS), which manages everything on the computer.

2. **Application Software**: These are programs that help you do specific things, like word processors (e.g., Microsoft Word), web browsers (e.g., Google Chrome), or games.

What is the Difference between Program and Software?

Program	Software
A single set of instructions to perform a specific task.	A collection of programs, tools, and data that work together.
Focused on a single function or task.	It covers a wider range of tasks and often includes multiple programs.
Relatively simple, performing one function.	More complex, made up of multiple programs and components.
It lacks proper documentation.	It is well documented.
Smaller in size, typically one application.	Larger in size, often includes many programs and tools.

Types of software:

1. **Product:** Software's developed based on market requirements. Developed product-based companies like Microsoft
2. **Project:** Developed for a specific customer. Developed by service-based companies

   These are further classified as:
   System Software: System software is designed to manage and control the computer hardware so that other software can run.
   Application Software: Application software is designed to perform specific tasks for the user, such as writing, browsing, or playing games.
   Development Software: Development software helps developers create, test, and maintain other software.
   Middleware: Middleware acts as a bridge between different software applications or between software and hardware.
   Firmware: Firmware is software that is permanently programmed into hardware devices and is used to control hardware.
   Enterprise Software: Enterprise software is designed to help businesses and organizations manage and automate their processes.