1. What is software? What is software engineering?

Answer:- Software is a set of computer programs and associated documentation and data. This is in contrast to hardware, from which the system is built and which actually performs the work.

Software engineering is an engineering-based approach to software development. A software engineer is a person who applies the engineering design process to design, develop, maintain, test, and evaluate computer software. The term programmer is sometimes used as a synonym, but may also refer more to implementation rather than design and can also lack connotations of engineering education or skills.

1. Explain type of software.

Answer:- Among the various categories of software, the most common types include the following:

* **Application software.**The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user. Examples of [modern applications](https://www.techtarget.com/searchcio/feature/The-rise-of-modern-applications-Why-you-need-them) include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.
* **System software.** These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the [firmware](https://www.techtarget.com/whatis/definition/firmware), computer language translators and system [utilities](https://www.techtarget.com/whatis/definition/utility).
* **Driver software.**Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.
* **Middleware.**The term *middleware* describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work with legacy ones.
* **Programming software.** Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and [debug](https://www.techtarget.com/searchsoftwarequality/definition/debugging) other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.

1. What is SDLC? Explain each phase of SDLC.

Answer:- The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure.

The following phase of SDLC(Software development cycle)

1. Requirement gathering and analysis: Collecting and defining the needs and specifications of the software.
2. Feasibility study: Evaluating the technical, economic, and operational feasibility of the software.
3. Design: Creating the architecture and design of the software components and interfaces.
4. Coding: Writing the source code of the software using programming languages and tools.
5. Testing: Verifying and validating the functionality and quality of the software.
6. Installation/Deployment: Deploying the software to the target environment and making it available to the users.
7. Maintenance: Providing support and updates to the software to fix bugs and enhance features.







