**Q1: What is software program?**

A1: A software program is commonly defined as a set of instructions, or a set of modules or procedures, that allow for a certain type of computer operation. The term is also often used interchangeably with terms like “software application” and “software product.

At the [lowest programming level](https://en.wikipedia.org/wiki/Low_level_language), [executable code](https://en.wikipedia.org/wiki/Executable_code) consists of [machine language](https://en.wikipedia.org/wiki/Machine_code) instructions supported by an individual [processor](https://en.wikipedia.org/wiki/Microprocessor)—typically a [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU) or a [graphics processing unit](https://en.wikipedia.org/wiki/Graphics_processing_unit) (GPU). Machine language consists of groups of [binary values](https://en.wikipedia.org/wiki/Binary_number) signifying [processor instructions](https://en.wikipedia.org/wiki/Instruction_set_architecture) that change the state of the computer from its preceding state. For example, an instruction may change the value stored in a particular storage location in the computer—an effect that is not directly observable to the user. An instruction [may also invoke](https://en.wikipedia.org/wiki/System_call) one of many [input or output operations](https://en.wikipedia.org/wiki/Input/output), for example displaying some text on a computer screen; causing state changes which should be visible to the [user](https://en.wikipedia.org/wiki/User_(computing)). The majority of software is written in [high-level programming languages](https://en.wikipedia.org/wiki/High-level_programming_language). They are easier and more efficient for programmers because they are closer to [natural languages](https://en.wikipedia.org/wiki/Natural_language) than machine languages.[[3]](https://en.wikipedia.org/wiki/Software#cite_note-3) High-level languages are translated into machine language using a [compiler](https://en.wikipedia.org/wiki/Compiler) or an [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)) or a combination of the two. Software may also be written in a low-level [assembly language](https://en.wikipedia.org/wiki/Assembly_language), which has a strong correspondence to the computer's machine language instructions and is translated into machine language using an [assembler](https://en.wikipedia.org/wiki/Assembly_language).

**Q2: Differentiate between Complied and Interpreted language?**

A2: Compiled languages are converted directly into machine code that the processor can execute. As a result, they tend to be faster and more efficient to execute than interpreted languages. They also give the developer more control over hardware aspects, like memory management and CPU usage.

Compiled languages need a “build” step – they need to be manually compiled first. You need to “rebuild” the program every time you need to make a change. In our hummus example, the entire translation is written before it gets to you. If the original author decides that he wants to use a different kind of olive oil, the entire recipe would need to be translated again and resent to you. Interpreters run through a program line by line and execute each command. Here, if the author decides he wants to use a different kind of olive oil, he could scratch the old one out and add the new one. Your translator friend can then convey that change to you as it happens.

Interpreted languages were once significantly slower than compiled languages. But, with the development of, that gap is shrinking.

**Q3: Differentiate between Psudocode and algorithm?**

A3: An algorithm is an unambiguous specification of how to solve a problem. Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.

**Usage**

An algorithm helps to simplify and understand the problem. On the other hand, pseudocode is a method of developing an algorithm.

**Conclusion**

An algorithm is an arrangement of steps to solve a problem. A pseudo-code uses natural language or compact mathematical notation to write algorithms. The main difference between algorithm and pseudocode is that an algorithm is a step by step procedure to solve a given problem while a pseudocode is a method of writing an algorithm.

**Q4: What the mathematical operator used in a programming langauage?**

A4: The Java programming language supports various arithmetic operators for all floating-point and integer numbers. These operators are + (addition), - (subtraction), \* (multiplication), / (division), and % (modulo). The following table summarizes the binary arithmetic operations in the Java programming language.

|  |  |  |
| --- | --- | --- |
| **Binary Arithmetic Operators** | | |
| **Operator** | **Use** | **Description** |
| **+** | op1 + op2 | Adds op1 and op2; also used to concatenate strings |
| **-** | op1 - op2 | Subtracts op2 from op1 |
| **\*** | op1 \* op2 | Multiplies op1 by op2 |
| **/** | op1 / op2 | Divides op1 by op2 |
| **%** | op1 % op2 | Computes the remainder of dividing op1 by op2 |

Here's an example program, Arithmetic Demo that defines two integers and two double-precision floating-point numbers and uses the five arithmetic operators to perform different arithmetic operations. This program also uses + to concatenate strings

**Q5: Mention any 5 relational operator?**

A5: A conditional statement uses conditional logic to determine what programming statements to execute. By using some conditional logic (a process of checking conditions), we determine if a certain condition is true or false. If the condition is true, we execute some part of the program. Otherwise, if the condition is false, we use some other part of the program.

|  |  |  |
| --- | --- | --- |
| **Relational operators** | | |
| **Operator Name** | **Symbol** | **Description** |
| Equal | = | Returns true if both sides are **equal**. |
| Greater than | > | Returns true if the variable on the left is **greater** than the variable on the right. |
| Less than | < | Returns true if the variable on the left is **less** than the variable on the right. |
| Greater than or equal to | >= | Returns true if the variable on the left is **greater than or equal to** the value of the variable on the right |
| Less than or equal to | <= | Returns true if the variable on the left is **less than or equal to** the value of the variable on the right |
| Not equal to | <> | Returns true if both sides are **not equal**. |

**Q6: What are the loops?**

A6: In computer programming, a loop is a sequence of [instruction](https://whatis.techtarget.com/definition/instruction) s that is continually repeated until a certain condition is reached. Typically, a certain process is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number. If it hasn't, the next instruction in the sequence is an instruction to return to the first instruction in the sequence and repeat the sequence. If the condition has been reached, the next instruction "falls through" to the next sequential instruction or branches outside the loop. A loop is a fundamental programming idea that is commonly used in writing programs. An [infinite loop](https://whatis.techtarget.com/definition/infinite-loop-endless-loop) is one that lacks a functioning exit routine . The result is that the loop repeats continually until the operating system senses it and terminates the program with an error or until some other event occurs (such as having the program automatically terminate after a certain duration of time). The structure of a loop can be virtually divided into two parts, namely the control statement, and the body. The control statement of a loop comprises the conditions that have to be met for the execution of the body of the loop. For every iteration of the loop, the conditions in the control statement have to be true. The body of a loop comprises the block of code or the sequence of logical statements that are to be executed multiple times. There are two types of loops in Python, namely, For loop, and While loop. When a loop is written within another loop, the control structure is termed as a nested loop.

**Q7: Write the syntax of while loop in Javascript?**

A7: **Looping Statements in C** execute the sequence of statements many times until the stated condition becomes false. A loop in C consists of two parts, a body of a loop and a control statement. The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the C loop is to repeat the same code a number of times.

Depending upon the position of a control statement in a program, looping statement in C is classified into two types:

1. Entry controlled loop

2. Exit controlled loop

In an **entry control loop in C,** a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.

In an **exit controlled loop**, a condition is checked after executing the body of a loop. It is also called as a post-checking loop.

**Q8: Write the syntax of for loop in Javascript?**

A8: The 'for' loop is the most compact form of looping. It includes the following three important parts −

The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.

The test statement which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.

The iteration statement where you can increase or decrease your counter.

Q9: What is the software development life cycle?

A9: Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

* SDLC is the acronym of Software Development Life Cycle.
* It is also called as Software Development Process.
* SDLC is a framework defining tasks performed at each step in the software development process.
* ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.
* SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

**Q10: Explain the waterfall software development life cycle model?**

A10: The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development.The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

**System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

**Implementation**− With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

**Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

**Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**Q11: Explain the v-shaped software development life cycle model?**

A11: The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.

**Verification:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

**Design Phase:**

**Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.

**System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.

**Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

**Module Design:** In this phase the system breaks dowm into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

**Testing Phases:**

**Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.

**Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.

**System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional

**Requirements of the developed application**.

**User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user’s requirement and system is ready for use in real world.

**Industrial Challange:** As the industry has evolved, the technologies have become more complex, increasingly faster, and forever changing, however, there remains a set of basic principles and concepts that are as applicable today as when IT was in its infancy.

Accurately define and refine user requirements.

Design and build an application according to the authorized user requirements.

Validate that the application they had built adhered to the authorized business requirements.

**Q12: Explain the Agile software development life cycle model?**

A12: The Agile software development life cycle is the structured series of stages that a product goes through as it moves from beginning to end. It contains six phases: concept, inception, iteration, release, maintenance, and retirement.

The Agile life cycle will vary slightly depending on the [project management methodology](https://www.wrike.com/project-management-guide/methodologies/) chosen by a team. For example, [Scrum](https://www.wrike.com/project-management-guide/faq/what-is-scrum-in-project-management/) teams work in short time periods known as [sprints](https://www.wrike.com/project-management-guide/faq/what-is-a-sprint-in-agile/), which are similar to iterations. They also have clearly defined roles, such as [Scrum master](https://www.wrike.com/project-management-guide/faq/what-is-an-agile-scrum-master/). On the other hand, [Kanban](https://www.wrike.com/features/kanban-boards/) teams have more of a continuous flow with no required roles. Another example is [Extreme Programming](https://www.wrike.com/project-management-guide/faq/what-are-the-different-types-of-agile-methodologies/), where teams tend to work in shorter iterations and place an extra focus on engineering practices.

However, the goal of all software development teams is the same to deliver working software to users on time.

Here, a product owner will determine [the scope of their project](https://www.wrike.com/project-management-guide/faq/what-is-scope-in-project-management/). If there are numerous projects, they will prioritize the most important ones. The product owner will discuss key requirements with a client and prepare documentation to outline them, including what features will be supported and the proposed end results. It is advisable to keep the requirements to a minimum as they can be added to in later stages. In the concept stage, the product owner will also estimate the time and cost of potential projects. This detailed analysis will help them to decide whether or not a project is feasible before commencing work