**DAY 1**

**SQL Statements Overview**

SQL (Structured Query Language) is used to interact with databases. It consists of various types of statements categorized based on their functionality. Here's a concise breakdown:

**1. Data Definition Language (DDL)**

These statements define and modify the structure of a database.

* **CREATE**: Creates a new database object (e.g., table, view).

CREATE TABLE Employees (ID INT, Name VARCHAR(50), Age INT);

* **ALTER**: Modifies an existing database object.

ALTER TABLE Employees ADD Salary INT;

* **DROP**: Deletes a database object.

DROP TABLE Employees;

**TRUNCATE:** Removes all records from a table

TRUNCATE TABLE Employees;

**2. Data Manipulation Language (DML)**

These statements handle data within tables.

* **INSERT**: Adds new records to a table.

INSERT INTO Employees (ID, Name, Age) VALUES (1, 'John Doe', 30);

* **UPDATE**: Modifies existing records.

UPDATE Employees SET Age = 31 WHERE ID = 1;

* **DELETE**: Removes records from a table.

DELETE FROM Employees WHERE ID = 1;

**3. Data Query Language (DQL)**

Used to retrieve data from the database.

* **SELECT**: Fetches data from tables.

SELECT \* FROM Employees;

**4. Data Control Language (DCL)**

Manages access and permissions.

* **GRANT**: Provides access to users.

GRANT SELECT ON Employees TO User1;

* **REVOKE**: Removes access from users.

REVOKE SELECT ON Employees FROM User1;

**5. Transaction Control Language (TCL)**

Manages transactions in the database.

* **COMMIT**: Saves changes permanently.

COMMIT;

* **ROLLBACK**: Reverts changes to the last save point.

ROLLBACK;

* **SAVEPOINT**: Sets a point to roll back to.

SAVEPOINT Save1;

**DAY 2**

**PL/SQL**

|  |  |  |
| --- | --- | --- |
| **Feature** | **SQL** | **PL/SQL** |
| **Full Form** | Structured Query Language | Procedural Language/SQL |
| **Type** | Declarative language | Procedural programming language |
| **Purpose** | Used to query and manipulate data | Used to write logic, control flow, and manage transactions |
| **Execution** | Executes one statement at a time | Executes a block of statements together |
| **Variables** | Not supported | Fully supports variables, constants, and data types |
| **Control Structures** | Not available (no loops, conditions) | Supports loops, conditions (IF, FOR, WHILE, etc.) |
| **Error Handling** | Limited (mostly syntax errors) | Robust exception handling (EXCEPTION block) |
| **Use Case** | Data retrieval, insertion, update, deletion | Writing procedures, functions, triggers, and packages |

**Features of PL/SQL**

PL/SQL brings the benefits of procedural programming to the relational database world. Some of the most important features of PL/SQL include:

* **Block Structure:**PL/SQL can execute several queries in one block using single command.
* **Procedural Constructs:** One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
* **Error Handling:**PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
* **Reusable Code**: Create stored procedures, functions, triggers, and packages, which can be executed repeatedly.
* **Performance**: Reduces network traffic by executing multiple SQL statements within a single block

**Structure of PL/SQL Block**

***DECLARE* (optional)** *declaration statements;****BEGIN*** *executable statements****EXCEPTIONS* (optional)** *exception handling statements****END;***

(DECLARE

v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary FROM Employees WHERE EmpID = 101;

IF v\_salary > 50000 THEN

DBMS\_OUTPUT.PUT\_LINE('High salary');

END IF;

END; )

**PL/SQL code is written in blocks, which consist of three main sections:**

* Declare section starts with **DECLARE** keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.
* Execution section starts with **BEGIN** and ends with **END** keyword. This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL\*PLUS built-in functions as well.
* Exception section starts with **EXCEPTION** keyword. This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

ANONYMOUS BLOCK:

In PL/SQL, That's blocks which is not have header are known as anonymous blocks. These blocks do not form the body of a function or triggers or procedure. Example: Here a code example of find greatest number with Anonymous blocks.

NAMED BLOCK :

That's PL/SQL blocks which having header or labels are known as Named blocks. These blocks can either be subprograms like functions, procedures, packages or Triggers. Example: Here a code example of find greatest number with Named blocks means using function.

DECLARE

School\_name constant varchar2(20) := “DPS”

BEGIN

Dbms\_output.put\_line(‘ I study in ‘|| school\_name);

* **CHAR**: It Handles variable-length binary data and fixed-length character strings. If the string is less than the defined length, then the rest is filled up with spaces. For instance, CHAR(10) would store string as CHAR data type that has a length of 10 characters regardless of the actual string length.
* **VARCHAR2**: To store character strings of varying lengths. **VARCHAR2** is slightly different because it only allocates the required amount of space required to store the string. For instance,**VARCHAR2(10)** data type can accommodate a string with as many as 10 characters.
* **LONG**: It Can store variable-length character strings of up to 2 gigabytes. However, it is deprecated and it should be replaced with **CLOB**to 2 GB. However, it is deprecated and should be avoided in favor of CLOB.

DECLARE

school\_name constant varchar2(20) := “DPS”

BEGIN

dbms\_output.put\_line(‘ I study in ‘|| school\_name);

END;

**SCALAR DATA TYPES**

|  |  |
| --- | --- |
| 1 | **Numeric**  Numeric values on which arithmetic operations are performed. |
| 2 | **Character**  Alphanumeric values that represent single characters or strings of characters. |
| 3 | **Boolean**  Logical values on which logical operations are performed. |
| 4 | **Datetime**  Dates and times. |

1.

DECLARE

a int;

b int;

BEGIN

a := :a;

b := :b;

if (a>b) THEN

dbms\_output.put\_line(' a is greater than b');

else if (b>a) THEN

dbms\_output.put\_line('b is greater than a');

else

dbms\_output.put\_line('Both a and b are equal');

end if;

END;

2.

DECLARE

a int;

b int;

BEGIN

a := :a;

b := mod(a,2);

CASE b

when 0 then dbms\_output.put\_line('Even Number');

when 1 then dbms\_output.put\_line('Odd Number');

else dbms\_output.put\_line('User has not given any input value to check');

END CASE;

END;

3.

DECLARE

marks NUMBER := 71;

result VARCHAR2(10);

BEGIN

CASE

WHEN marks >= 50 THEN

result := 'Pass';

ELSE

result := 'Fail';

END CASE;

dbms\_output.put\_line('Result: ' || result);

END;

4. BASIC LOOP

DECLARE

i int;

BEGIN

i := 3; starting no.

LOOP

IF i>10 then till 10

exit;

end if;

dbms\_output.put\_line(i);

i := i+1;

END LOOP;

END;

5. table

DECLARE

i int;

BEGIN

FOR i in 1..10 ///range

For range only 2 dots n no 3 or more

LOOP

dbms\_output.put\_line( '2 x ' || i || ' = ' ||(2 \* i));

END LOOP;

END;

6. WHILE LOOP

DECLARE

num int :=1;

BEGIN

while(num <= 10)

LOOP

dbms\_output.put\_line(''|| num);

num := num+2;

END LOOP;

END;

7.

DECLARE

i number(2);

BEGIN

FOR i IN 1..10

LOOP

dbms\_output.put\_line(i);

END LOOP;

END;

8.

DECLARE

a1 number;

b1 number;

c1 number;

FUNCTION findMax(x IN number, y IN number)

RETURN number

IS

z number;

BEGIN

IF x>y THEN

z:= x;

ELSE

z:=y;

END IF;

RETURN z;

END;

BEGIN

a1 := 10;

b1 := 100;

c1 := findMax(a1, b1);

dbms\_output.put\_line('Maximum number in 10 and 100 is:' || c1);

END;

**9. creation of a table**

create table user5(id number(10) primary key, name varchar2(100));

**ANOTHER**

create or replace procedure "INSERTUSER3"

(id IN NUMBER,

name IN VARCHAR2)

is

begin

insert into user5 values(id, name);

end;

10. create a procedure in which create a table, insert value and update the value

CREATE OR REPLACE PROCEDURE ManageTestTable IS

BEGIN

-- Step 1: Create the table

EXECUTE IMMEDIATE '

CREATE TABLE TestTable (

ID NUMBER PRIMARY KEY,

Name VARCHAR2(50)

)

';

-- Step 2: Insert a value

INSERT INTO TestTable (ID, Name)

VALUES (1, 'Initial Name');

-- Step 3: Update the value

UPDATE TestTable

SET Name = 'Updated Name'

WHERE ID = 1;

-- Optional: Show result

DBMS\_OUTPUT.PUT\_LINE('Table created, value inserted, and updated.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

CREATE OR REPLACE PROCEDURE ManageTestTable IS

BEGIN

EXECUTE IMMEDIATE '

CREATE TABLE TestTable (

ID NUMBER PRIMARY KEY,

Name VARCHAR2(50)

)

';

INSERT INTO "INSERTUSER3 (ID, Name)

VALUES (1, 'Initial Name');

UPDATE INSERTUSER5

SET Name = 'Updated Name'

WHERE ID = 1;

DBMS\_OUTPUT.PUT\_LINE('Table created, value inserted, and updated.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

11.

Here assigned value of roll no to sno

Both dt is same

CREATE TABLE Students (

RollNo INT PRIMARY KEY,

StudentName VARCHAR(100),

Course VARCHAR(100)

);

INSERT INTO Students(RollNo, StudentName, Course)

VALUES (1, 'VIDISHYA', 'JAVA');

DECLARE

CURSOR showRec(sno student.RollNo%type) IS

SELECT StudentName, Course FROM Students WHERE RollNo=sno;

x Students.StudentName%type;

y Students.Course%type;

z Students.RollNo%type;

BEGIN

z := :RollNo;

OPEN showRec(z);

IF showRec%Isopen = FALSE then

dbms\_output.put\_line('Cannot open Cursor');

ELSE LOOP

FETCH showRec into x,y;

EXIT WHEN showRec%NOTFOUND;

dbms\_output.put\_line(x|| '' || y);

END LOOP;

END IF;

CLOSE showRec;

END;

DECLARE

v\_emp\_salary employees.salary%TYPE;

v\_emp\_id employee.employee\_id%TYPE :=101;

v\_bonus Number;

BEGIN

SELECT salary INTO v\_emp\_salary

FROM employees

WHERE employee\_id = v\_emp\_id;

v\_bonus := v\_emp\_salary / 0 ;

DBMS\_OUTPUT.PUT\_LINE('Employee Salary:' || v\_emp\_salary);

DBMS\_OUTPUT.PUT\_LINE('Calculated Bonus:' || v\_bonus);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No employee found with ID' || v\_emp\_ID);

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Multiple employees found with ID' || v\_emp\_ID);

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Division by zero occured in bonus calculation');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected Error:' || SQLERRM);

END;

KEYS, NORMALIZATION, INDEXING, VIEW TO READ AND PRACTICE

**Keys:**

|  |  |
| --- | --- |
| **Type of Key** | **Description** |
| **Primary Key** | Uniquely identifies each record in a table. Cannot be NULL. |
| **Foreign Key** | Refers to the primary key in another table to establish relationships. |
| **Candidate Key** | A column (or set of columns) that can qualify as a primary key. |
| **Composite Key** | A primary key made up of two or more columns. |
| **Unique Key** | Ensures all values in a column are unique, but allows one NULL. |

**Normalization:**

Normalization is the process of organizing data to **reduce redundancy** and improve data integrity.

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| **1NF** | Eliminate repeating groups; ensure atomicity of data. |
| **2NF** | Remove partial dependencies (non-key attributes depend on part of a composite key). |
| **3NF** | Remove transitive dependencies (non-key attributes depend on other non-key attributes). |
| **BCNF** | A stricter version of 3NF; every determinant must be a candidate key. |

**Indexing**

Indexing improves the speed of data retrieval operations on a database table.

|  |  |
| --- | --- |
| **Type of Index** | **Description** |
| **Single-column** | Index on one column. |
| **Composite** | Index on multiple columns. |
| **Unique Index** | Ensures indexed values are unique. |
| **Full-text Index** | Used for searching text-based data efficiently. |
| **Clustered Index** | Sorts and stores data rows in the table based on the index. |
| **Non-clustered Index** | Maintains a separate structure from the data rows. |

**Views**

A **view** is a virtual table based on the result of a SQL query.

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Virtual Table** | Doesn't store data physically, just the query result. |
| **Simplifies Queries** | Can hide complexity and join multiple tables. |
| **Security** | Can restrict access to specific columns or rows. |
| **Updatable Views** | Some views can be updated if they meet certain criteria. |

**Collection**

**PL/SQL Collections**

In PL/SQL, a **collection** is an ordered group of elements, all of the same data type. Collections are used to store and manipulate data in memory, making them useful for handling arrays, lists, or tables within PL/SQL programs. Each element in a collection is identified by a unique subscript (index).

**Types of Collections in PL/SQL**

1. **Associative Arrays (Index-By Tables)**:
   * Key-value pairs where the key can be a number or string.
   * Dynamically sized and sparse (**non-contiguous** indexes are allowed).
   * Example:

DECLARE

TYPE AssocArray IS TABLE OF VARCHAR2(50) INDEX BY PLS\_INTEGER;

my\_array AssocArray;

BEGIN

my\_array(1) := 'Apple';

my\_array(2) := 'Banana';

DBMS\_OUTPUT.PUT\_LINE(my\_array(1)); -- Output: Apple

END;

Library or dmart

1. **Nested Tables**:
   * Similar to a database table, but stored in memory.
   * Can grow dynamically and allow non-contiguous elements.
   * Example:
   * DECLARE
   * TYPE NestedTable IS TABLE OF NUMBER;
   * my\_table NestedTable := NestedTable(10, 20, 30);
   * BEGIN
   * my\_table.EXTEND; -- Add a new element
   * my\_table(4) := 40;
   * DBMS\_OUTPUT.PUT\_LINE(my\_table(4)); -- Output: 40
   * END;
2. **Varrays (Variable-Size Arrays)**:
   * Fixed maximum size, but elements are stored in contiguous memory.
   * Useful when the number of elements is known and limited.
   * Example:
   * DECLARE
   * TYPE VarrayType IS VARRAY(5) OF NUMBER;
   * my\_varray VarrayType := VarrayType(1, 2, 3);
   * BEGIN
   * DBMS\_OUTPUT.PUT\_LINE(my\_varray(2)); -- Output: 2
   * END;

**Key Operations on Collections**

* **Initialization**: Use constructors for Nested Tables and Varrays.
* **Manipulation**: Use methods like EXTEND, DELETE, TRIM, etc.
* **Iteration**: Use loops to iterate through elements.

**Common Use Cases**

* Temporary storage of data during program execution.
* Passing arrays of data between PL/SQL blocks or procedures.
* Bulk operations for performance optimization.

Collections are a powerful feature in PL/SQL, enabling efficient data handling and manipulation in various scenarios.

**DAY 3**

[**12 Principles of Agile Methodology**](https://www.bing.com/ck/a?!&&p=7011d287dc9eb022bbe1bda04de2978848f64f87797c9fdd07fb6f460493b096JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=12+principles+of+agile+methodology&u=a1aHR0cHM6Ly93d3cuYWdpbGVhbGxpYW5jZS5vcmcvYWdpbGUxMDEvMTItcHJpbmNpcGxlcy1iZWhpbmQtdGhlLWFnaWxlLW1hbmlmZXN0by8&ntb=1)

The **12 principles of Agile methodology**, derived from the Agile Manifesto, serve as a foundation for flexible, efficient, and customer-focused software development. These principles emphasize collaboration, adaptability, and continuous improvement to deliver value effectively.

1. **Customer Satisfaction**: The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. **Embrace Change**: Agile welcomes changing requirements, even late in development, to provide a competitive advantage for the customer.
3. **Frequent Delivery**: Deliver working software frequently, ranging from a couple of weeks to a couple of months, with a preference for shorter timescales.
4. **Collaboration**: Business stakeholders and developers must work together daily throughout the project to ensure alignment and success.
5. **Motivated Teams**: Build projects around motivated individuals, providing them with the environment and support they need while trusting them to get the job done.
6. **Face-to-Face Communication**: The most efficient and effective method of conveying information within a development team is face-to-face conversation.
7. **Working Software as a Measure**: Progress is primarily measured by the delivery of working software.
8. **Sustainable Development**: Agile promotes sustainable development, enabling sponsors, developers, and users to maintain a constant pace indefinitely.
9. **Technical Excellence**: Continuous attention to technical excellence and good design enhances agility.
10. **Simplicity**: Focus on maximizing the amount of work not done, as simplicity is essential for efficiency.
11. **Self-Organizing Teams**: The best architectures, requirements, and designs emerge from self-organizing teams.
12. **Continuous Improvement**: At regular intervals, the team reflects on how to become more effective and adjusts its behavior accordingly.

These principles guide Agile teams in delivering high-quality software while adapting to changing needs and fostering collaboration

**Scrum master**

The role of the Scrum Master has a lot of layers and facets to it. While building awareness around Scrum and enabling greater agility, Scrum Masters also need soft skills that are needed to coach and mentor members of the Scrum Team and others in the organization. Scrum Masters are accountable for helping their teams succeed, and that often means offering them assistance in groups or on a one-on-one basis. They may facilitate exercises, give guidance or help people come to conclusions on their own. Not everyone has the skills necessary to be a Scrum Master, and that is important to keep in mind when considering this career path.  
  
In the end, the Scrum Master is accountable for the Scrum Team’s effectiveness as they help the Scrum Team to improve how the team works together to create value on an ongoing basis.

What does a Scrum Master do?

Scrum Masters utilize their unique skillset to do a lot of critical work that helps the Scrum Team and the organization as listed below.

**The Scrum Master:**

helps the Scrum Team:

* By coaching the team members in self-management and cross-functionality
* Focus on creating high-value Increments that meet the Definition of Done
* Influence the removal of impediments to the Scrum Team’s progress
* Ensure that all Scrum events take place and are positive, productive, and kept within the timebox.

helps the Product Owner:

* Find techniques for effective Product Goal definition and Product Backlog management
* Provide ways for the Scrum Team to understand the need for clear and concise Product Backlog items
* Establish empirical product planning for a complex environment
* Facilitate stakeholder collaboration as requested or needed

supports the Organization:

* By Leading, training and coaching them in their Scrum adoption
* By helping employees and stakeholders understand and instill an empirical approach for complex work
* Remove barriers between stakeholders and Scrum Teams

**Product Owner**

**A Product Owner is a key role in Agile methodologies, particularly in Scrum, responsible for maximizing the value of the product and managing the product backlog.**

**Definition and Responsibilities**

The Product Owner (PO) is accountable for ensuring that the development team delivers value to the business and its customers. This role involves several key responsibilities:

1. [**Maximizing Product Value**: The Product Owner is responsible for maximizing the value of the product resulting from the work of the Scrum Team. This includes understanding customer needs and aligning the product with business objectives.](https://www.bing.com/ck/a?!&&p=5462ead0ed80b26036fe4d488dcae146b0789010d1f2b4656e68236824ee0f08JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cuc2NydW0ub3JnL3Jlc291cmNlcy93aGF0LWlzLWEtcHJvZHVjdC1vd25lcg&ntb=1)
2. [**Managing the Product Backlog**: The PO creates, maintains, and prioritizes the product backlog, which is a list of features, user stories, and tasks that need to be completed. This backlog is continuously refined based on feedback and changing requirements.](https://www.bing.com/ck/a?!&&p=5462ead0ed80b26036fe4d488dcae146b0789010d1f2b4656e68236824ee0f08JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cuc2NydW0ub3JnL3Jlc291cmNlcy93aGF0LWlzLWEtcHJvZHVjdC1vd25lcg&ntb=1)
3. [**Defining User Stories**: The Product Owner translates customer requirements into user stories that the development team can work on. These stories help clarify what needs to be built and why.](https://www.bing.com/ck/a?!&&p=c56c6af2414882c45b80e4e2cf9b8d957cb48ccef6f50435e6e94e67585f90a0JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cuZ2Vla3Nmb3JnZWVrcy5vcmcvYnVzaW5lc3Mtc3R1ZGllcy9wcm9kdWN0LW93bmVyLWRlZmluaXRpb24tcm9sZXMtYW5kLXJlc3BvbnNpYmlsaXRpZXMv&ntb=1)
4. [**Stakeholder Engagement**: The PO acts as the primary point of contact for stakeholders, gathering feedback and ensuring that the development team understands the priorities and needs of the business and its customers.](https://www.bing.com/ck/a?!&&p=bcd61164f2d9a1d1446317ad96ed2fe8694655f7674e247e1092bb7a483fa6c2JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cucG1pLm9yZy9kaXNjaXBsaW5lZC1hZ2lsZS9wcm9kdWN0LW93bmVy&ntb=1)
5. [**Participating in Scrum Events**: The Product Owner is actively involved in Scrum ceremonies, such as Sprint Planning and Sprint Reviews, to provide direction and feedback to the team.](https://www.bing.com/ck/a?!&&p=c56c6af2414882c45b80e4e2cf9b8d957cb48ccef6f50435e6e94e67585f90a0JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cuZ2Vla3Nmb3JnZWVrcy5vcmcvYnVzaW5lc3Mtc3R1ZGllcy9wcm9kdWN0LW93bmVyLWRlZmluaXRpb24tcm9sZXMtYW5kLXJlc3BvbnNpYmlsaXRpZXMv&ntb=1)

**Skills and Qualities**

To be effective, a Product Owner should possess several key skills and qualities, including:

* **Strong Communication Skills**: The ability to clearly articulate the product vision and requirements to both the development team and stakeholders.
* **Decision-Making Ability**: The PO must make informed decisions about priorities and trade-offs to maximize product value.
* [**Customer Focus**: Understanding customer needs and ensuring that the product meets those needs is crucial for success.](https://www.bing.com/ck/a?!&&p=5462ead0ed80b26036fe4d488dcae146b0789010d1f2b4656e68236824ee0f08JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cuc2NydW0ub3JnL3Jlc291cmNlcy93aGF0LWlzLWEtcHJvZHVjdC1vd25lcg&ntb=1)

**Importance in Agile Development**

[The Product Owner plays a critical role in Agile development by bridging the gap between the business and the development team. They ensure that the team is focused on delivering the most valuable features first, which helps in achieving business goals and improving customer satisfaction.](https://www.bing.com/ck/a?!&&p=bcd61164f2d9a1d1446317ad96ed2fe8694655f7674e247e1092bb7a483fa6c2JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&psq=product+owner&u=a1aHR0cHM6Ly93d3cucG1pLm9yZy9kaXNjaXBsaW5lZC1hZ2lsZS9wcm9kdWN0LW93bmVy&ntb=1)

In summary, the Product Owner is a vital part of the Scrum team, responsible for guiding the product's development and ensuring that it delivers maximum value to users and stakeholders alike.

**User requirement**

**When the requirement is completed**

* Development testing done
* All acceptance tests are running and passed
* No defects found
* Product owner has accepted the requirement
* Software product is delivered to the end user

**What is Software Acceptance Criteria?**

**Acceptance Criteria** are the **conditions that a software product must meet** to be accepted by the user, customer, or stakeholders.

They define:

* **What is expected** from a feature or user story
* **How it will be tested**
* **When it is considered “done”**

**Example:**

For a login feature, acceptance criteria might be:

* User can log in with valid credentials
* Error message is shown for wrong password
* Login button is disabled until both fields are filled

**Purpose:**

* Ensures **clarity** between developers, testers, and stakeholders
* Helps in **test case creation**
* Prevents **scope creep**
* Acts as a **checklist** for completion

**Advantages of Agile Methodology**

Agile is a **flexible, iterative approach** to software development. It focuses on **collaboration, customer feedback, and small, rapid releases**.

**Key Advantages:**

|  |  |
| --- | --- |
| **Benefit** | **Description** |
| **Faster Delivery** | Work is done in short cycles (sprints), so features are delivered quickly |
| **Customer Involvement** | Regular feedback ensures the product meets real needs |
| **Flexibility** | Easy to adapt to changes in requirements |
| **Improved Quality** | Continuous testing and integration help catch bugs early |
| **Better Collaboration** | Daily stand-ups and team involvement improve communication |
| **Transparency** | Everyone knows what’s being worked on and what’s next |
| **Risk Reduction** | Problems are identified early, reducing chances of failure |

**Disadvantages of agile methodology**

* Not useful for small developments projects
* Lack of intensity on necessary designing and doc
* Requires expert member to take crucial decision
* Cost is slightly more
* Project can quickly go out of track if pm not clear about requirement and outcome

**CI/CD**

 (Continuous Integration and Continuous Delivery/Deployment)

In the past, software delivery was slow and full of manual work. Developers used to create separate feature branches and worked on them for weeks or even months.

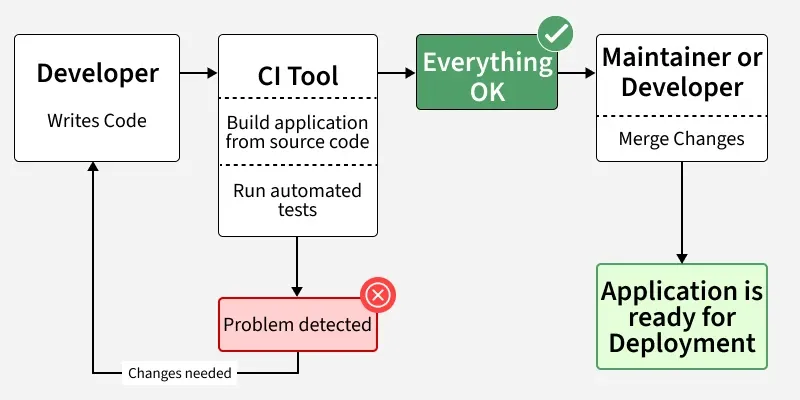
* All these branches were merged at the end, which often caused big conflicts and broken builds. Testing and building the code were also manual, usually done at the final stage, so bugs were discovered very late and were costly to fix.
* Deployment was a long and risky process, taking days or sometimes weeks, because everything was released together in one big update.
* Teams were also divided: developers wrote code, testers tested it, and operations managed servers.

**After CI/CD**

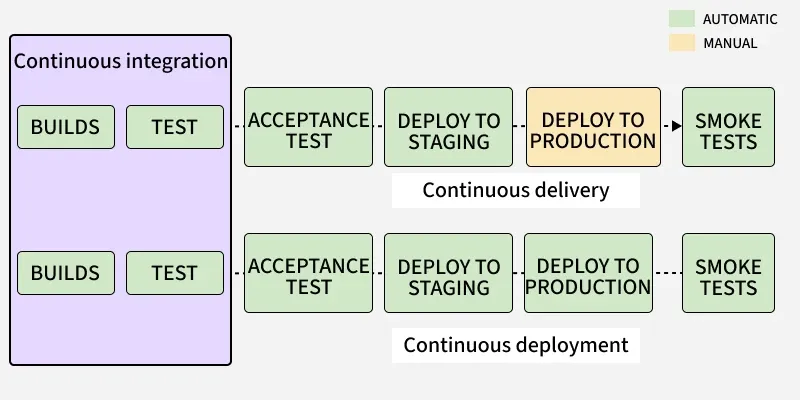
With CI/CD, the whole process became faster, automated, and more reliable. Developers now commit their code frequently to a shared main branch, which prevents conflicts and ensures smooth integration.

* Automated pipelines run tests, builds, and checks as soon as code is pushed, so errors are caught early. Continuous Delivery makes sure that tested code is automatically packaged and ready for deployment
* While Continuous Deployment takes it a step further by directly releasing to production without manual steps. This means smaller, frequent updates instead of one big risky release.
* Tools like Git, Jenkins, Docker, and Kubernetes automate everything from building to deployment, making the workflow transparent and collaborative. Bugs are fixed quickly, rollbacks are easier, and overall software delivery happens in hours instead of days

**Workflow of CI/CD**

****

**CI and CD Workflow**

****

**Advantages of CI/CD (Continuous Integration / Continuous Deployment)**

* CI/CD is a DevOps practice that automates the process of building, testing, and deploying code.
* **Key Advantages:**

|  |  |
| --- | --- |
| * **Benefit** | * **Description** |
| * **Faster Releases** | * Code changes are automatically tested and deployed, reducing time to market. |
| * **Improved Code Quality** | * Automated tests catch bugs early. |
| * **Reduced Manual Work** | * Automation reduces human error and repetitive tasks. |
| * **Quick Feedback** | * Developers get instant feedback on their code. |
| * **Better Collaboration** | * Teams can work on features in parallel and merge changes easily. |
| * **Rollback Support** | * Easy to revert to a previous version if something breaks. |

* **What is Scrum?**
* **Scrum** is an **Agile framework** used to manage and complete complex projects, especially in software development.
* It breaks work into **small, manageable pieces** called **sprints** (usually 1–4 weeks long).
* **Uses of Scrum**

|  |  |
| --- | --- |
| * **Use Case** | * **Description** |
| * **Software Development** | * Most common use — helps teams deliver features quickly. |
| * **Product Management** | * Keeps product development aligned with customer needs. |
| * **Marketing Projects** | * Organizes campaigns and content creation. |
| * **Operations & Support** | * Manages tasks and improvements in IT teams. |

* **What is the Scrum Framework?**
* The **Scrum Framework** defines **roles, events, and artifacts** to help teams work efficiently.
* **🔹 Scrum Roles:**
* **Product Owner** – Defines the product vision and prioritizes work.
* **Scrum Master** – Facilitates the process and removes blockers.
* **Development Team** – Builds the product.
* **🔹 Scrum Events:**

|  |  |
| --- | --- |
| * **Event** | * **Purpose** |
| * **Sprint** | * A time-boxed period to complete work (1–4 weeks) |
| * **Sprint Planning** | * Decide what to work on in the sprint |
| * **Daily Scrum** | * 15-minute daily meeting to sync up |
| * **Sprint Review** | * Show what was built to stakeholders |
| * **Sprint Retrospective** | * Reflect and improve the process |

* **🔹 Scrum Artifacts:**

| * **Artifact** | * **Description** |
| --- | --- |
| * **Product Backlog** | * List of all features and tasks |
| * **Sprint Backlog** | * Tasks selected for the current sprint |
| * **Increment** | * The working product delivered at the end of the sprint |

* Sprint planning meeting-

Between scrum master, product owner, and team collaborate

‘what’ and ‘how’ discussed

PO shares features to be completed, tea determines tasks required

Team reviews workload if all tasks completed on time sprint, if no low priority features return

* Track progress
* Sprint review
* Release planning

**ADVANTAGES OF SCRUM**

Scrum is a popular Agile framework that organizes work into iterative cycles called Sprints. It is widely adopted across various industries due to its adaptability and efficiency. Here are some key advantages of using Scrum methodology:

* **Adaptable Approach**

Scrum is highly adaptable, making it suitable for projects with undefined goals or requirements at the beginning. It operates in Sprints, allowing changes to be implemented in subsequent Sprints without impacting the overall project.

* **Highly Efficient**

Scrum teams are usually small, which means each member takes full responsibility for their work. This results in better work quality and timely identification and resolution of bottlenecks.

* **Promotes Creativity**

Scrum encourages continuous monitoring and improvement of the project. This allows team members to brainstorm and come up with creative ideas to enhance the product.

* **Lowers Project Cost**

Scrum minimizes excessive documentation and focuses on actionable plans and user stories. This reduces overhead costs compared to conventional methods, allowing developers to concentrate on current tasks.

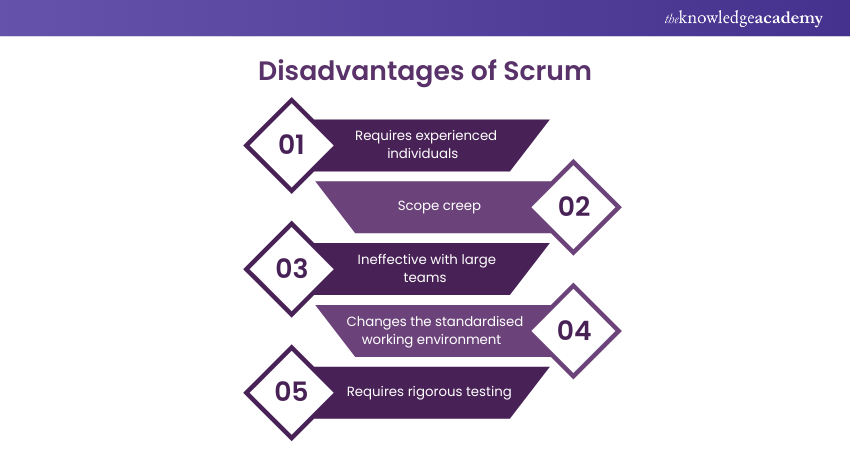
* **Provides Regular Feedback**

Daily Scrum meetings facilitate progress updates, identification of impediments, and team collaboration. This regular feedback helps maintain alignment with Sprint goals and ensures smooth project functioning[**1**](https://www.bing.com/ck/a?!&&p=41b22af1024b9dc3b6cea39f9e20add8b429aae43ba8e1b76ef2ea7dc74965f3JmltdHM9MTc1NjY4NDgwMA&ptn=3&ver=2&hsh=4&fclid=024f2945-db91-6a52-316d-3f1eda396b05&u=a1aHR0cHM6Ly93d3cudGhla25vd2xlZGdlYWNhZGVteS5jb20vYmxvZy9hZHZhbnRhZ2VzLWFuZC1kaXNhZHZhbnRhZ2VzLW9mLXNjcnVtLw&ntb=1).

* **Transparent**

Scrum promotes transparency by ensuring regular feedback and participation from all team members. Even small modifications in the project are visible to everyone, building trust and ensuring the delivery of high-quality products.

**DISADVANTAGES OF SCRUM**



* **Requires Experienced Individuals**: Scrum projects typically require highly skilled and experienced employees. Lack of experience can make it challenging to operate projects effectively.
* **Scope Creep**: While Scrum is flexible, it can still face issues with scope creep, especially if team members leave during the process, increasing the risk of project failure.
* **Ineffective with Large Teams**: Scrum is not designed for large projects as it requires extensive coordination and communication, which can be challenging with larger teams.
* **Changes the Standardized Working Environment**: Implementing Scrum may require significant changes in the organizational hierarchy, promoting collaboration over traditional hierarchical structures.
* **Requires Rigorous Testing**: Scrum focuses on delivering business value iteratively, which involves continuous inspection, adaptation, and rigorous testing.
* **Difficult to Scale**: Scaling Scrum for larger projects can be challenging and may require extensive training and precise coordination.

**DEVOPS TOOLS**

* 1. **DOCKER**

Allows building, ship and run distributed applications on multiple system.

Suitable for container management.

* Configures system more comfortable and faster
* Increases productivity
* Provides container used to run application in an isolated environment
* Routes incoming request for published ports on available nodes to an active container. Enables connection even if there is no task running.
* Allows saving secrets into swarm itself.
  1. **JENKINS**

For monitoring execution of repeated tasks.

Allows continuous integration

* Increases scale of automation
* Distribute task across multiple machines
* Supports CI/CD
* Offers 400 plugins to support building and testing any project virtually
* Requires little maintenance
* Built-in GUI tool
* Easily set up
  1. **GIT**

Handle minor to major projects

Work among programmers

Allows team members to work on same workspace

* Free open-source tool
* Supports pull request
* Enables faster release cycle
* Scalable
* Secure and fast

It is used for:

* Tracking code changes
* Tracking who made changes
* Coding collaboration
  1. **SELENIUM**

Portable software framework for web applications

Easy interface for developing automated tests

* Free open-source tool
* Supports multi-platforms for testing
* Easy to build key-driven framework for web-driver

SCRUM EVENT (ceremonies)

1.sprint

2. sprint planning

3. daily scrum

4.sprint review

5. sprint retrospective

**WHAT IS KANBAN**

The Kanban system is a visual workflow management method that increase teams productivity, streamline their processes and improve efficiency using Kanban System. The principles of Kanban, which have their roots in Toyota's production method, are to visualize work, reduce [work-in-progress (WIP),](https://www.geeksforgeeks.org/software-engineering/what-is-the-concept-of-work-in-progress-wip-limits-in-kanban/) and improve flow. It focuses on Continuous improvement and adaptation by enabling teams to pull work via a predetermined process and guaranteeing that tasks are finished successfully and efficiently, One such tool is the Kanban board, which facilitates transparency and teamwork by showing tasks as cards arranged in columns that correspond to the various workflow stages.

Workflow phases

* **To-Do:** The workflow is started with an empty clear board where all the tasks are meant to be listed. A To-Do list is formed in the first column of the board which consists of all the user stories. The individual cards must provide a brief description and necessary details of tasks like due date, functionalities, nature of task, etc.
* **In-Progress:** As the name suggests, the In-Progress column consists of all the tasks that are currently under progress and under development phase. It acts as an indication that a team is currently operating working on it.
* **Validation:** In the validation phase, the code is reviewed, and the model is tested against several sets of tests like unit testing, acceptance testing, system testing, etc. All the functionalities are reviewed and verified. Apart from this, all the bugs and inconsistencies found are rectified in the validation phase itself.
* **Done:** After all the functional and system tests are performed and quality assurance is reviewed, the task is shifted to Done state (4th column on the board). All the tasks move column to column towards right until they reach the last column i.e. Done phase. The done column reflects all the tasks that are finished and deployed without any errors.

**Tools for Kanban**

* **Trello**: User-friendly and great for small teams.
* **Jira**: Powerful tool for larger teams, especially in software development.
* **Asana**: Versatile for both project management and Kanban.
* **Kanbanize**: Specifically designed for Kanban and integrates analytics and reporting.

**Benefits**

1. Visual Workflow

* Kanban uses a board with columns (like "To Do", "In Progress", "Done") to show the status of tasks.
* This makes it easy to see what’s happening at a glance.

2. Limits Work in Progress (WIP)

* You can set limits on how many tasks can be in progress at once.
* This helps teams focus better and avoid multitasking, which improves quality and speed.

3. Improves Flow

* By tracking how tasks move through the board, teams can spot bottlenecks and fix them.
* This leads to a smoother and faster workflow.

4. Increases Flexibility

* Unlike Scrum, Kanban doesn’t require fixed-length sprints.
* You can add or change tasks anytime, which is great for dynamic environments.

5. Encourages Continuous Improvement

* Teams regularly review the board and performance.
* This helps them find ways to improve how they work over-time.

6. Better Collaboration

* Everyone can see the same board, so it’s easier to communicate and coordinate.

Characteristics

| **S. No.** | **Scrum** | **Kanban** |
| --- | --- | --- |
| 1. | It defines the role of each member of the [**Scrum team**](https://www.geeksforgeeks.org/software-engineering/scrum-team/). | There is no role assigned to individuals. |
| 2. | It follows the iterative method. | It does not follow the iterative approach |
| 3. | To solve a problem, it breaks it into small tasks and then processes it further. | It does not break a problem into sub-problems. |
| 4. | It is a highly prescriptive approach. | It is not much prescriptive as compared to **Scrum**. |
| 5. | There is no **visualization process** to perform tasks. | There is a **visualization process** to perform tasks. |
| 6. | There are sprints that keep track of the progress of any project. | They use task cards to keep track of the progress of any project. |
| 7. | It is processed in successive sprints to complete a task. | It is used to optimize the task to complete a project. |
| 8. | It is not preferred when resources are limited. | It is preferred when tasks and resources are limited. |
| 9. | **Scrum Master** is the problem solver in case of a problem. | All the members are allowed to pick a problem and solve it. |
| 10. | The process does not get disturbed if a team member leaves in between a sprint. | The flow of work gets disturbed if a team member leaves in between. |
| 11. | The velocity of the sprint is used to measure the production. | The time taken to finish the project is the measure of production. |
| 12. | Estimation is crucial to **Scrum** because it places a strong emphasis on planning. | Estimation is not as important in **Kanban** as in **scrum**. |
| 13. | In s**crum**, **cross-functional teams** are important to deal with the issues that may occur during software development. | In **Kanban**, specialized teams are important. |
| 14. | Only one team owns a sprint backlog. | The sharing among multiple teams is possible with the **Kanba**n board. |
| 15. | The **scrum**methodology is centered on the backlog. | The **Kanban** methodology is centered on the process dashboard. |
| 16. | It is suitable for projects that have changing priorities. | It is suitable for projects that have stable priorities i.e. unlikely to change over time. |
| 17. | "Velocity" through sprints is a production measurement metric. | "Cycle time" is a production measurement metric. |
| 18. | One to four weeks make up a sprint cycle. | The delivery cycle is continuous. |
| 19. | **Some of the Tools-**   * Jira Software * Axosoft * VivifyScrum and more. | **Some of the Tools-**   * Jira Software * Kanbanize * SwiftKanban * Asana and more. |

**PHASE OF DevOps LIFECYCLE**

1. Plan

Requirement gathering

Backlog creation

Sprint planning

1. Code(development)

Code reviews

1. Build
2. Test
3. Release
4. Deploy
5. Operate
6. Monitor

**DAY 4**

**What is AI (Artificial Intelligence)?**

AI is the **science of making machines smart**. It allows computers to:

* **Think** (like solving problems)
* **Learn** (from data and experience)
* **Act** (make decisions or predictions)

AI mimics human intelligence but works faster and with more data.

**How AI Impacts Our Day-to-Day Life**

AI is already part of our daily routine, often without us realizing it:

|  |  |  |
| --- | --- | --- |
| **Area** | **Example** | **How AI Helps** |
| **Phones** | Google Assistant, Siri | Understands voice commands |
| **Shopping** | Amazon, Flipkart | Recommends products |
| **Entertainment** | Netflix, YouTube | Suggests shows/movies |
| **Travel** | Google Maps, Uber | Finds best routes, estimates time |
| **Banking** | Fraud detection | Alerts suspicious activity |
| **Healthcare** | Diagnosis tools | Detects diseases early |

**How AI Impacts the Software Industry**

AI is changing how software is **developed, tested, and used**:

**In Development:**

* **Code Assistants**: Tools like GitHub Copilot suggest code.
* **Bug Prediction**: AI finds errors before testing.

**In Testing:**

* **Automated Testing**: AI writes and runs test cases.
* **Visual Testing**: Detects UI issues using image recognition.

**In Operations:**

* **Monitoring**: AI watches for crashes or slowdowns.
* **Chatbots**: Handle customer support 24/7.

**Why AI?**

AI is used because it:

* **Saves time** by automating tasks
* **Reduces errors** compared to humans
* **Works 24/7** without breaks
* **Learns from data** to improve over time
* **Handles big data** better than humans

**What is Generative AI (Gen AI)?**

Generative AI is a **type of AI that creates new content**. It can:

* Write text (like this explanation!)
* Generate images, music, or videos
* Create code, designs, and more

**Examples:**

* **ChatGPT**: Writes emails, answers questions
* **DALL·E**: Creates images from text
* **Bard, Claude, Copilot**: Other Gen AI tools

**Advantages of AI**

|  |  |
| --- | --- |
| **Advantage** | **Description** |
| **Speed** | Processes data faster than humans |
| **Accuracy** | Reduces human errors |
| **Efficiency** | Automates repetitive tasks |
| **Personalization** | Customizes user experience |
| **Scalability** | Can handle large-scale operations |

**Disadvantages of AI**

|  |  |
| --- | --- |
| **Disadvantage** | **Description** |
| **Job Loss** | May replace some human jobs |
| **Bias** | Can reflect bias in training data |
| **Cost** | Expensive to build and maintain |
| **Privacy** | Can misuse personal data |
| **Lack of Emotion** | Can’t understand human feelings deeply |

**Types of AI**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| **Narrow AI** | Does one task well | Siri, Google Translate |
| **General AI** | Thinks like a human (still in research) | Not yet available |
| **Super AI** | Smarter than humans (theoretical) | Sci-fi concept |

**What is Machine Learning (ML)?**

Machine Learning is a **subset of AI**. It means:

Teaching computers to learn from data and improve over time **without being explicitly programmed**.

**Example:**

You show a computer 1000 pictures of cats and dogs. It learns the difference and can then identify new images correctly.

**Types of ML:**

1. **Supervised Learning** – Learns from labeled data (e.g., spam vs. not spam)
2. **Unsupervised Learning** – Finds patterns in unlabeled data (e.g., customer segmentation)
3. **Reinforcement Learning** – Learns by trial and error (e.g., game-playing AI)

**LLM**

An **LLM** stands for **Large Language Model**. It’s a type of **artificial intelligence model** designed to understand and generate human-like text based on vast amounts of data. LLMs are a key part of modern **Natural Language Processing (NLP)** systems.

**What Makes a Model “Large”?**

* **Scale**: LLMs are trained on billions (or even trillions) of words.
* **Parameters**: These models have **millions to hundreds of billions of parameters**—the internal settings that the model learns during training.
* **Data**: They are trained on diverse datasets including books, websites, code, conversations, and more.

**How Do LLMs Work?**

LLMs use a type of neural network architecture called a **Transformer**, introduced in the paper *“Attention is All You Need”* (2017). Key concepts include:

* **Tokenization**: Breaking text into smaller units (words or subwords).
* **Embedding**: Converting tokens into numerical vectors.
* **Attention Mechanism**: Helps the model focus on relevant parts of the input when generating output.
* **Training**: Predicting the next word in a sentence, over and over, using massive datasets.

**DEEP LEARNING**

**What is Deep Learning?**

**Deep Learning** is a **subset of Machine Learning**, which is itself a subset of **Artificial Intelligence (AI)**.

It uses **artificial neural networks** — systems inspired by the human brain — to learn from **large amounts of data**.

**How It Works (Simple Explanation)**

Imagine how a child learns to recognize a dog:

1. They see many pictures of dogs.
2. They notice patterns (fur, ears, tail, bark).
3. Eventually, they can recognize a dog in a new picture.

Deep Learning works similarly:

* It takes **input data** (like images, text, or sound),
* Passes it through **layers of neurons** (like brain cells),
* And **learns patterns** to make decisions or predictions.

**Structure of a Deep Learning Model**

A **neural network** has:

* **Input Layer**: Takes in the data (e.g., an image).
* **Hidden Layers**: Multiple layers that process the data and learn features.
* **Output Layer**: Gives the result (e.g., “This is a cat”).

The more hidden layers it has, the **“deeper”** the network — hence the name **Deep Learning**.

**PROMPT ENGINEER**

Prompt engineering is the process of creating effective prompts that enable [**AI models**](https://www.geeksforgeeks.org/artificial-intelligence/what-is-ai-model/) to generate responses based on given inputs. Prompt engineering essentially means writing prompts intelligently for text-based Artificial Intelligence tasks, more specifically, [**Natural Language Processing**](https://www.geeksforgeeks.org/nlp/natural-language-processing-overview/) **(NLP)** tasks. In the case of such text-based tasks, these prompts help the user and the model generate a particular output as per the requirement. These requirements are efficiently added in the form of prompts and hence the name Prompt Engineering.

**What are Prompts?**

Prompts are short pieces of text that are used to provide context and guidance to [**machine learning models**](https://www.geeksforgeeks.org/machine-learning/machine-learning/). When talking about the specific text **AI tasks**, also called **NLP tasks**, these prompts are useful in generating relevant outputs which are as close to the expected output itself. Precisely, these prompts help in generating accurate responses by:

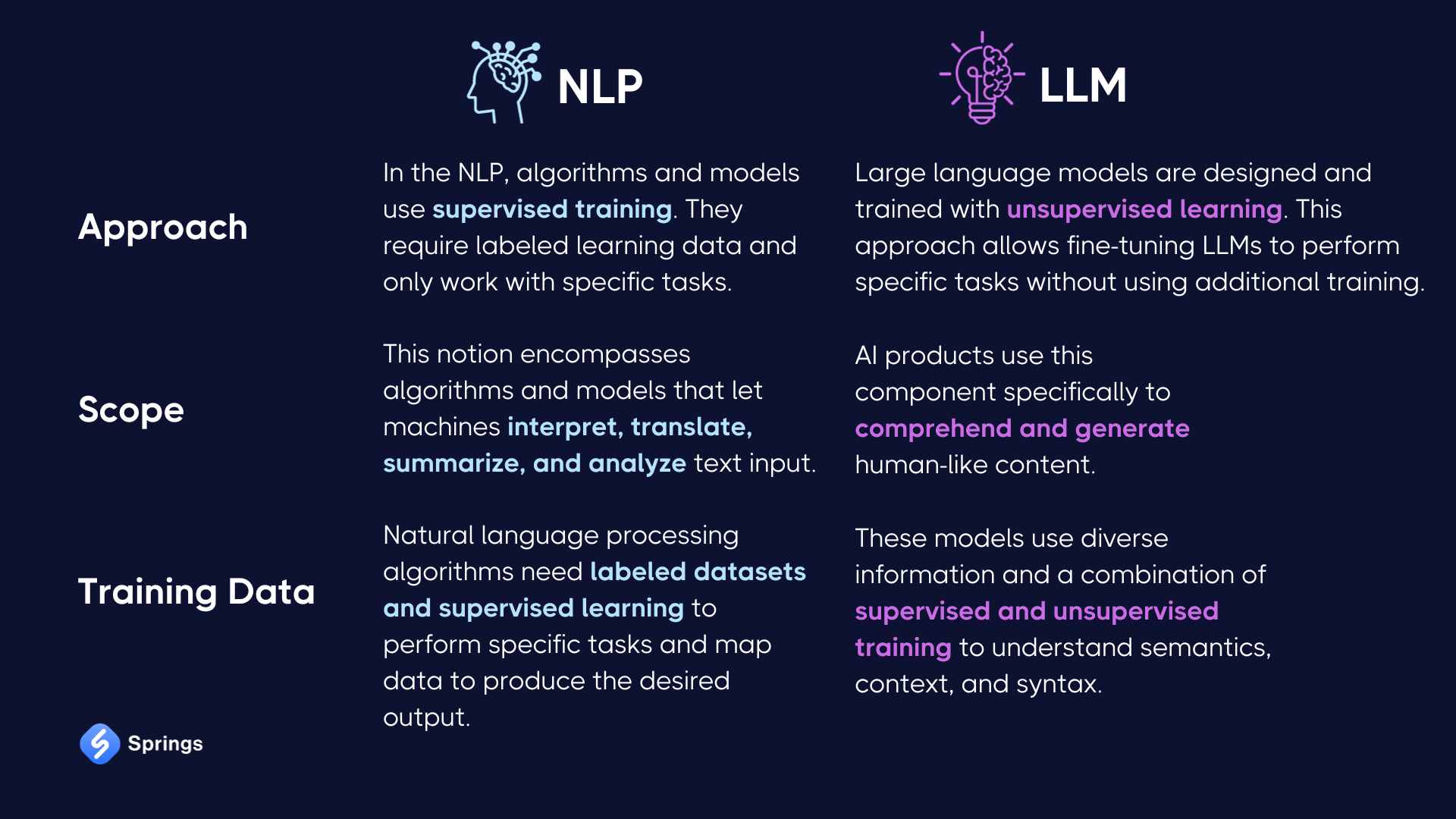
* Adding on some additional guidance for the model.
* Not generalizing a prompt too much.
* Make sure the information added is not too much as that can confuse the model.

Making the user intent and purpose clear for the model to generate content in the relevant context only.

**NLP**

[Natural language processing (NLP)](https://www.deeplearning.ai/the-batch/issue-44/) is the discipline of building machines that can manipulate human language — or data that resembles human language — in the way that it is written, spoken, and organized. It evolved from computational linguistics, which uses computer science to understand the principles of language, but rather than developing theoretical frameworks, NLP is an engineering discipline that seeks to build technology to accomplish useful tasks. NLP can be divided into two overlapping subfields: natural language understanding (NLU), which focuses on semantic analysis or determining the intended meaning of text, and natural language generation (NLG), which focuses on text generation by a machine. NLP is separate from — but often used in conjunction with — speech recognition, which seeks to parse spoken language into words, turning sound into text and vice versa.

**DIFFERENCE BETWEEN LLM AND NLP**



**DAY 5**

**JAVA**

All communication through objects(Identity,behaviour,function) only(OOP)C++

Class- data mem and data function

**JDK JVM JRE difference**

* [JDK](https://www.geeksforgeeks.org/java/jdk-in-java/): JDK stands for Java Development Kit. It is a set of development tools and libraries used to create Java programs. It works together with the JVM and JRE to run and build Java applications..
* [JRE](https://www.geeksforgeeks.org/java/jre-in-java/): JRE stands for **Java Runtime Environment,** and it provides an environment to run Java programs on the system. The environment includes Standard Libraries and JVM.
* [JVM](https://www.geeksforgeeks.org/java/how-jvm-works-jvm-architecture/): JVM stands for**Java Virtual Machine.**It's responsible for executing the Java program.

| * **Aspect** | **JDK** | **JRE** | **JVM** |
| --- | --- | --- | --- |
| **Purpose** | Used to develop Java applications | Used to run Java applications | Executes Java bytecode |
| **Platform Dependency** | Platform-dependent (OS specific) | Platform-dependent (OS specific) | JVM is OS-specific, but bytecode is platform-independent |
| **Includes** | JRE + Development tools (javac, debugger, etc.) | JVM + Libraries (e.g., rt.jar) | ClassLoader, JIT Compiler, Garbage Collector |
| **Use Case** | Writing and compiling Java code | Running a Java application on a system | Convert bytecode into native machine code |

**Note:** The JVM is platform-independent in the sense that the bytecode can run on any machine with a JVM, but the actual JVM implementation is platform-dependent. Different operating systems (e.g., Windows, Linux, macOS) require different JVM implementations that interact with the specific OS and hardware.



CODE 1:

public class DemoHi{

public static void main(String ar[]){

System.out.println("Welcome to java");

}

}

CODE 2:

import java.util.\*;

public class DemoHi{

public static void main(String ar[]){

Scanner sc=new Scanner(System.in);

System.out.println(“Enter 1st number");

int n1=sc.nextInt();

System.out.println("Enter 2nd number");

int n2=sc.nextInt();

int res=n1+n2;

System.out.println("**'Addition** of "+n1+" and "+n2+" = "+res);

}

}

CODE 3:

import java.util.\*;

public class DemoHi{

public static void main(String ar[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter 1st number");

int n1=sc.nextInt();

System.out.println("Enter 2nd number");

int n2=sc.nextInt();

int res=n1/n2;

System.out.println("**'Division** of "+n1+" and "+n2+" = "+res);

}

}

CODE 4:

import java.util.\*;

public class DemoHi{

public static void main(String ar[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter student name");

String s1=sc.next();

System.out.println("Enter 1st subject name");

int n1=sc.nextInt();

System.out.println("Enter 2nd subject name");

int n2=sc.nextInt();

System.out.println("Enter 3rd subject name");

int n3=sc.nextInt();

int res=n1+n2+n3;

System.out.println("Total marks"+res);

float percent=(float)((res\*100)/300);

System.out.println("percent marks = "+**percent**);

}

}

CODE 5:

import java.util.\*;

public class DemoHi{

public static void main(String ar[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter any number");

int n1=sc.nextInt();

if(n1%2==0){

System.out.println("Entered number"+n1+" is **even**");

}

else{

System.out.println("Entered number"+n1+" is **odd**");

}

}

}

CODE 6:

import java.util.\*;

public class DemoHi {

public static void main(String[] ar) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a year: ");

int year = sc.nextInt();

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

System.out.println(year + " is a leap year.");

} else {

System.out.println(year + " is not a **leap year**.");

}

}

}

CODE 7:

import java.util.\*;

public class DemoHi {

public static void main(String ar[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number: ");

int number = sc.nextInt();

if (number > 0) {

System.out.println("The number is **positive**.");

} else if (number < 0) {

System.out.println("The number is **negative**.");

} else {

System.out.println("The number is **zero**.");

}

}

}

CODE 8:

For loop

import java.util.\*;

public class DemoHi{

public static void main(String ar[]){

System.out.println("enter a number")

int n1=sc.nextInt();

}

}NOT DONE

CODE 9:

SWITCH CASE

import java.util.\*;

public class DemoHi {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n2 = 45;

int n3 = 57;

System.out.println("1 for addition, 2 for subtraction, 3 for division");

System.out.print("Enter choice: ");

int n1 = sc.nextInt();

switch (n1) {

case 1:

int res1 = n2 + n3;

System.out.println("Addition = " + res1);

break;

case 2:

int res2 = n2 - n3;

System.out.println("Subtraction = " + res2);

break;

case 3:

int res3 = n2 / n3;

System.out.println("Division = " + res3);

break;

default:

System.out.println("Wrong choice");

}

}

}

CODE 10:

import java.util.\*;

class abc{

public void add(int a, int b){

int res=a+b;

System.out.println("Addition =" +res);

}

}

public class DemoHi{

public static void main(String ar[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter 1st number");

int s1=sc.nextInt();

System.out.println("Enter 2nd number");

int s2=sc.nextInt();

abc ab=new abc();

ab.add(s1,s2);

}

}

OTHER(BOTH NOT DONE)

import java.util.\*;

class abc {

public void add(int a, int b) {

int res = a + b;

System.out.println("Addition = " + res);

}

}

public class DemoHi {

public static void main(String ar[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter 1st number");

int s1 = sc.nextInt();

System.out.println("Enter 2nd number");

int s2 = sc.nextInt();

abc ab = new abc();

ab.add(s1, s2);

}

}

While loop – entry level

Do while loop – exit level

**JAVA DATA TYPES**

**A diagram of data types

AI-generated content may be incorrect.**

| **Type** | **Description** | **Default** | **Size** | **Example Literals** | **Range of values** |
| --- | --- | --- | --- | --- | --- |
| **boolean** | true or false | false | JVM-dependent (typically 1 byte) | true, false | true, false |
| **byte** | 8-bit signed integer | 0 | 1 byte | (none) | -128 to 127 |
| **char** | Unicode character(16 bit) | \u0000 | 2 bytes | 'a', '\u0041', '\101', '\\', '\', '\n', 'β' | 0 to 65,535 (unsigned) |
| **short** | 16-bit signed integer | 0 | 2 bytes | (none) | -32,768 to 32,767 |
| **int** | 32-bit signed integer | 0 | 4 bytes | -2,0,1 | -2,147,483,648  to  2,147,483,647 |
| **long** | 64-bit signed integer | 0L | 8 bytes | -2L,0L,1L | -9,223,372,036,854,775,808  to  9,223,372,036,854,775,807 |
| **float** | 32-bit IEEE 754 floating-point | 0.0f | 4 bytes | 3.14f, -1.23e-10f | ~6-7 significant decimal digits |
| **double** | 64-bit IEEE 754 floating-point | 0.0d | 8 bytes | 3.1415d, 1.23e100d | ~15-16 significant decimal digits |

**Four pillar**

1.Encapsulation

2. Inheritance

3.Polymorphism

import java.util.\*;

class Payment{

void pay(){

System.out.println("Generic payment");

}

}

class CreditCardPayment extends Payment{

@Override

void pay(){

System.out.println("Payment using Credit Card");

}

}

class UpiPayment extends Payment{

@**Override**

void pay(){

System.out.println("Payment using UPI");

}

}

public class main{

public static void main(String[] args){

Payment p1 = new CreditCardPayment();

Payment p2 = new UpiPayment();

p1.pay();

p2.pay();

}

}

overloading

4.Abstraction

Abstract class contains atleast one abstract method.

import java.util.\*;

abstract class PaymentGateway{

abstract void pay(int amount);

}

class PayPal extends PaymentGateway{

void pay(int amount){

System.out.println("Paid" + amount + "using PayPal");

}

}

class Stripe extends PaymentGateway{

void pay(int amount){

System.out.println("Paid" + amount + "using Stripe");

}

}

public class Main{

public static void main(String[] args)

{

PaymentGateway payment = new PayPal();

payment.pay(500);

}

}

**Exception in java(recoverable)**

**Java Checked vs Unchecked Exceptions**

|  |  |
| --- | --- |
| **Error/Exception** | **Description** |
| ArithmeticError | Occurs when a numeric calculation goes wrong |
| ArrayIndexOutOfBoundsException | Occurs when trying to access an index number that does not exist in an array |
| ClassFormatError | Occurs when a class file cannot be accessed |
| ClassNotFoundException | Occurs when trying to access a class that does not exist |
| ConcurrentModificationException | Occurs when an element is added or removed from iterables |
| FileNotFoundException | Occurs when a file cannot be accessed |
| IncompatibleClassChangeError | Occurs when there's been a change in a base class after a child class has already been initialized |
| InputMismatchException | Occurs when entering wrong input (e.g. text in a numerical input) |
| InterruptedException | Occurs when a Thread is interrupted while waiting/sleeping |
| InvalidClassException | Occurs when the Serialization runtime observes a problem with a class |
| IOException | Occurs when an input or output operation fails |
| NegativeArraySizeException | Occurs when trying to create an array with negative size |
| NoClassDefFoundError | Occurs when the class is not found at runtime |
| NoSuchFieldException | Occurs when trying to access a class field/variable that does not exist |
| NoSuchMethodException | Occurs when trying to access a class method that does not exist |
| NullPointerException | Occurs when trying to access an object referece that is null |
| NumberFormatException | Occurs when it is not possible to convert a specified string to a numeric type |
| RuntimeException | Occurs when an exception occurs at runtime |
| StringIndexOutOfBoundsException | Occurs when trying to access a character in a String that does not exist |
| TypeNotPresentException | Occurs when a type cannot be found |
| IllegalArgumentException | Occurs when when an illegal argument is passed to a method |
| IllegalStateException | Occurs when when a method is called at an illegal time |

**collection frameworks in java**

Generic<string>

import java.util.\*;

public class ShoppingCart{

public static void main(String[] args){

List<String> cart = new ArrayList<>();

cart.add("Laptop");

cart.add("Phone");

cart.add("Phone");

cart.add("Headphones");

System.out.println("Shopping Cart Items:");

for (String item:cart){

System.out.println(item);

}

}

}

**difference between array list and vector**

|  |  |
| --- | --- |
| ArrayList performs faster for most operations, like adding, searching, or deleting elements. Since it does not have built-in synchronization, there's no delay from locking, which means better speed. | Vector is slower in comparison. Because it locks the whole structure during operations, other threads have to wait. The thread-safety comes at the cost of performance. |

Stack

In Java, a Stack is a linear data structure that follows the Last In First Out (LIFO) principle and is defined in the java.util package. Internally, it extends the Vector class.

**Key Characteristics of Vector**

* Maintains insertion order
* Allows duplicate and null values
* Grows dynamically when capacity is exceeded
* Implements List, RandomAccess, Cloneable and Serializable interfaces
* Inherits from the AbstractList class

**1. List**

A **list** is a collection of items that are **ordered** and **changeable**. You can add, remove, or access items by their position (index).

**Example (Python):**

fruits = ["apple", "banana", "cherry"]

print(fruits[1]) # Output: banana

**2. Stack**

A **stack** is a collection that follows **LIFO** (Last In, First Out). The last item added is the first one removed.

**Example:**

* Add: Push
* Remove: Pop

stack = []

stack.append("book1")

stack.append("book2")

print(stack.pop()) # Output: book2

**Use case**: Undo feature in apps, browser back button.

**3. Queue**

A **queue** follows **FIFO** (First In, First Out). The first item added is the first one removed.

**Example:**

* Add: Enqueue
* Remove: Dequeue

from collections import deque

queue = deque()

queue.append("person1")

queue.append("person2")

print(queue.popleft()) # Output: person1

**Use case**: Print queue, customer service line.

**4. Vector**

A **vector** is like a dynamic array that can grow or shrink in size. In C++ and Java, vectors are used instead of arrays when size is not fixed.

**Example (C++):**

#include &lt;vector&gt;

#include &lt;iostream&gt;

using namespace std;

int main() {

vector&lt;int&gt; numbers = {1, 2, 3};

numbers.push\_back(4);

cout &lt;&lt; numbers[2]; // Output: 3

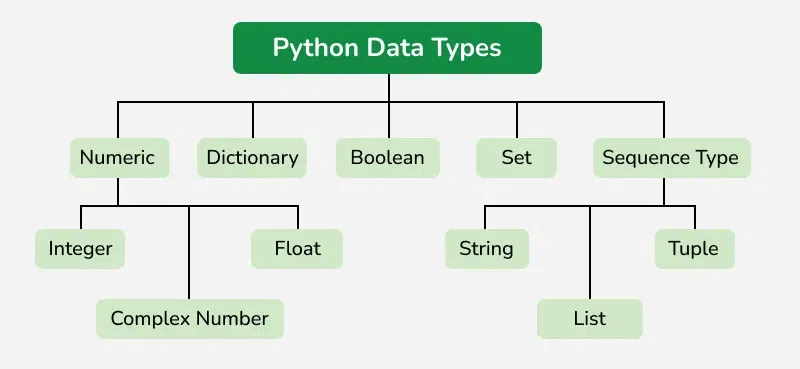
}

**Use case**: When you need a resizable array.

**DAY 6**

**PYTHON**

**Data types**



1. **Numeric -**[int](https://www.geeksforgeeks.org/python/python-int-function/), [float](https://www.geeksforgeeks.org/python/python-float-type-and-its-methods/), [complex](https://www.geeksforgeeks.org/python/python-complex-function/)
2. **Sequence Type -**[string](https://www.geeksforgeeks.org/python/python-string/), [list](https://www.geeksforgeeks.org/python/python-lists/), [tuple](https://www.geeksforgeeks.org/python/python-tuples/)
3. **Mapping Type -**[dict](https://www.geeksforgeeks.org/python/python-dictionary/" \t "_blank)
4. **Boolean -**[bool](https://www.geeksforgeeks.org/python/boolean-data-type-in-python/)
5. **Set Type -**[set](https://www.geeksforgeeks.org/python/python-sets/), [frozenset](https://www.geeksforgeeks.org/python/frozenset-in-python/" \t "_blank)
6. **Binary Types -**[bytes](https://www.geeksforgeeks.org/python/python-bytes-method/), [bytearray](https://www.geeksforgeeks.org/python/python-bytearray-function/), [memoryview](https://www.geeksforgeeks.org/python/memoryview-in-python/" \t "_blank)
7. **Numeric data type**

* **Integers** - value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). There is no limit to how long an integer value can be.
* **Float**- value is represented by float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, character e or E followed by a positive or negative integer may be appended to specify scientific notation.
* **Complex Numbers** - It is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example - 2+3j

1. **Sequence data type**

* **String**

Python Strings are arrays of bytes representing Unicode characters. In Python, there is no character data type, a character is a string of length one. It is represented by str class. String in Python can be created using single quotes, double quotes or even triple quotes. We can access individual characters of a String using index.

* **List**

Lists are just like arrays, declared in other languages which is an ordered collection of data. It is very flexible as items in a list do not need to be of the same type.

* **Tuple Data Type**

Tuple is an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable. Tuples cannot be modified after it is created.

1. **Boolean data type**

[Python Boolean](https://www.geeksforgeeks.org/python/boolean-data-type-in-python/) Data type is one of the two built-in values, **True or False**. Boolean objects that are equal to True are truthy (true) and those equal to False are falsy (false). However non-Boolean objects can be evaluated in a Boolean context as well and determined to be true or false. It is denoted by class bool.

1. **Set Data Type**

In Python Data Types, Set is an **unordered** collection of data types that is iterable, mutable, and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

1. **Dictionary Data Type**

A dictionary in Python is a collection of data values, used to store data values like a map, unlike other Python Data Types, a Dictionary holds a key: value pair. Key-value is provided in dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon:) whereas each key is separated by a ‘comma’.

**High level language and low-level language**

**A person sitting in front of a computer

AI-generated content may be incorrect.**

**High-Level Language**

A **high-level language (HLL)** is a **human-readable** programming language that simplifies coding by hiding complex hardware details, letting developers focus on **logic** and functionality.

* Designed to make writing code simpler and faster.
* Allow developers to build large programs more easily.
* Easier to find and fix mistakes.
* Can work on different computers with minimal adjustments.
* Usually slower in performance compared to machine-oriented languages.
* Provide many ready-made features to speed up coding.
* Good for beginners and widely used for everyday software.
* Examples include JavaScript, Ruby, Swift, and PHP.

**Low-Level Language**

A **low-level language** is a machine-oriented programming language that provides **minimal abstraction from hardware**, offering direct control over memory and system resources for maximum performance and efficiency.

* Provide direct access to the computer’s hardware.
* Require detailed knowledge of how a computer works.
* Less user-friendly, making programming more challenging.
* Harder to find and solve errors in code.
* Not easily adapted for use on different hardware.
* Used mostly for specific tasks needing high performance or precise control.
* Often faster and use fewer system resources.
* Examples include Binary code and Assembly languages like MIPS or ARM.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **High-Level Language** | **Low-Level Language** |
| **Abstraction Level** | High abstraction, closer to human language | Low abstraction, closer to machine code |
| **Ease of Use** | Easier to learn and use | More complex and harder to learn |
| **Portability** | Highly portable across different systems | Less portable, often system-specific |
| **Development Speed** | Faster development time | Slower development time |
| **Examples** | Python, Java, C++, JavaScript | Assembly language, Machine code(binary or hexadecimal instructions that are directly executed by the CPU. It is difficult for humans to read and write.) |
| **Memory Management** | Automatic memory management | Manual memory management |
| **Error Handling** | Built-in error handling features | Limited error handling, requires manual checks |
| **Performance** | Generally slower execution | Generally faster execution |
| **Use Cases** | Application development, scripting, web development | System programming, embedded systems, device drivers |

**Keywords**

(Predefined words which can be used for )

Keywords in Python are special reserved words that are part of the language itself. They define the rules and structure of Python programs, which means you **cannot use them as names for your variables, functions, classes, or any other identifiers.**

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',   
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',   
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

**List:**

In Python, a list is a built-in data structure that can hold an ordered collection of items. Unlike arrays in some languages, Python lists are very flexible:

* Can contain duplicate items
* **Mutable:** items can be modified, replaced, or removed
* **Ordered:** elements maintain the order in which they were added
* **Index-based:** items are accessed using their position (starting from 0)
* Can store mixed data types (integers, strings, booleans, even other lists)

Python lists are mutable data structures.

**Tuple :**

A tuple in Python is an immutable ordered collection of elements.

* Tuples are similar to lists, but unlike lists, they cannot be changed after their creation (i.e., they are immutable).
* Tuples can hold elements of different data types.
* The main characteristics of tuples are being ordered , heterogeneous and immutable.

**Set:**

Python set is an **unordered collection**of multiple items having different datatypes. In Python, sets are **mutable**, **unindexed**and do not contain duplicates. The order of elements in a set is not preserved and can change.

Python Sets can be created by using the built-in [set() function](https://www.geeksforgeeks.org/python/python-set-function/) with an iterable object or a sequence by placing the sequence inside curly braces, separated by a 'comma'.

**Dictonary:**

Python dictionary is a data structure that stores the value in **key: value pairs**. Values in a dictionary can be of any data type and can be **duplicated, whereas keys can't be repeated and must be immutable.**

**Code 1:**

Here’s a simple example demonstrating arithmetic operations in Python:

# Define two numbers

a = 12

b = 4

# Perform arithmetic operations

addition = a + b # Addition

subtraction = a - b # Subtraction

multiplication = a \* b # Multiplication

division = a / b # Division (float result)

# Print the results

print("Addition:", addition)

print("Subtraction:", subtraction)

print("Multiplication:", multiplication)

print("Division:", division)

**Output:**

Addition: 16

Subtraction: 8

Multiplication: 48

Division: 3.0

**Code 2:**

num = int(input("Enter a number: "))

if num % 2 == 0:

print(f"{num} is Even")

else:

print(f"{num} is Odd")

if (num % 4 == 0 and num % 100 != 0) or (num % 400 == 0):

print(f"{num} is a Leap Year")

else:

print(f"{num} is not a Leap Year")

**code 3:**

a = int(input("Enter first number: "))  
b = int(input("Enter second number: "))  
c = int(input("Enter third number: "))

print("Maximum number:", max(a, b, c))

**constructor**

* With a **constructor**, the name is set **automatically** when the object is created. Here, we can call inside the object only.
* Without a constructor, you have to **call a method manually** to set the data.

Here, we have to separately define the class we have created.

def \_\_init\_\_ (self, name)

* def → This starts a function definition.
* \_\_init\_\_ → This is a **special method** in Python. It’s called **automatically** when you create an object.
* self → Refers to the current object being created.
* name → A parameter you pass when creating the object. You can pass any value here.

def \_\_init\_\_ (self):

* def → This starts a function definition.
* greet → This is the name of the method. You can name it anything you like.
* self → Refers to the current object calling the method. It lets the method access the object’s data.

**four pillars**

**Multilevel Inheritance in Python**

Multilevel inheritance is a type of inheritance in Python where a class is derived from another class, which itself is derived from yet another class. This forms a chain-like hierarchy, where properties and methods are passed down through multiple levels.

**Key Features**

1. **Hierarchy**: A child class inherits from a parent class, and this child class can act as a parent for another class.
2. **Reusability**: Code from multiple levels can be reused, reducing redundancy.
3. **Access**: The last child class can access properties and methods of all its ancestor classes.

**Syntax Example**

class Grandparent:

def show\_grandparent(self):

print("This is the Grandparent class.")

class Parent(Grandparent):

def show\_parent(self):

print("This is the Parent class.")

class Child(Parent):

def show\_child(self):

print("This is the Child class.")

child\_obj = Child()

child\_obj.show\_grandparent()

child\_obj.show\_parent()

child\_obj.show\_child()

**Output**

This is the Grandparent class.

This is the Parent class.

This is the Child class.

**Advantages**

* Promotes **code reusability** across multiple levels.
* Simplifies **hierarchical relationships** in complex systems.

**Considerations**

* Be cautious of **method resolution order (MRO)**, as Python resolves methods from left to right in the inheritance chain.
* Overuse of multilevel inheritance can make the code harder to debug and maintain.

This structure is particularly useful when modeling real-world hierarchies, such as family trees or organizational structures.

class Dog:

def speak(self):

return "Woof!"

class Cat:

def speak(self):

return "Meow!"

def animal\_sound(animal):

print(animal.speak())

dog = Dog()

cat = Cat()

animal\_sound(dog)

animal\_sound(cat)

**Polymorphism**

**Polymorphism** means **"many forms"**. In programming, it allows the same function or method to behave differently based on the object or data type.

**Example (Python):**

class Animal:

def sound(self):

print("Some sound")

class Dog(Animal):

def sound(self):

print("Bark")

class Cat(Animal):

def sound(self):

print("Meow")

# Polymorphism in action

for animal in [Dog(), Cat()]:

animal.sound()

**Output**:

Bark

Meow

**Abstraction**

**Abstraction** means **hiding the complex details** and showing only the essential features.

**Example:**

When you drive a car, you don’t need to know how the engine works — just how to use the steering, brake, and accelerator.

**Example (Python):**

from abc import ABC, abstractmethod

class Shape(ABC):

@abstractmethod

def area(self):

pass

class Circle(Shape):

def area(self):

return 3.14 \* 5 \* 5

**Encapsulation**

**Encapsulation** means **wrapping data and methods** into a single unit (like a class) and **restricting access** to some parts.

**Example (Python):**

class Person:

def \_\_init\_\_(self):

self.\_\_age = 0 # private variable

def set\_age(self, age):

if age &gt; 0:

self.\_\_age = age

def get\_age(self):

return self.\_\_age

**Loops**

Loops are used to **repeat a block of code** multiple times.

**Types:**

* **for loop** – used when you know how many times to repeat
* **while loop** – used when you don’t know how many times

**Example:**

# for loop

for i in range(3):

print(i)

# while loop

count = 0

while count &lt; 3:

print(count)

count += 1

**Exception Handling**

Used to **handle errors** in a program without crashing it.

**Example:**

try:

x = 10 / 0

except ZeroDivisionError:

print("You can't divide by zero!")

finally:

print("This always runs.")

**pass Keyword**

The pass keyword is used as a **placeholder** when you don’t want to write any code yet.

**Example:**

def my\_function():

pass # code will be added later

**DAY 7**

**Exception handling in python**

**CODE :**

try:

n = 0

res = 100 / n

except **ZeroDivisionError**:

print("You can't divide by zero!")

except ValueError:

print("Enter a valid number!")

else:

print("Result is", res)

finally:

print("Execution complete.")

**OUTPUT:**

You can't divide by zero!

Execution complete.

**EXPLANATION:**

**Try** will run the code which might gave an error.

**Except** catches the error and prints the message.

**Else** if any exception occurs then this block will be ignored.

**Finally** block will run regardless the condition.

balance = 5000  # User's account balance

try:

    amount = int(input("Enter amount to withdraw: "))

    if amount > balance:

        raise ValueError("Insufficient balance.")

    result = balance - amount

except ValueError as e:

    print("Error:", e)

except ZeroDivisionError:

    print("Unexpected math error.")

else:

    print("Withdrawal successful. Remaining balance:", result)

finally:

    print("Thank you for using the ATM.")

**Output1:**

Enter amount to withdraw: 7000

ERROR!

Error: Insufficient balance.

Thank you for using the ATM.

=== Code Execution Successful ===

**Output2:**

Enter amount to withdraw: 4000

Withdrawal successful. Remaining balance: 1000

Thank you for using the ATM.

=== Code Execution Successful ===

**File handling**

File handling refers to the process of performing operations on a file, such as creating, opening, reading, writing and closing it through a programming interface. It involves managing the data flow between the program and the file system on the storage device, ensuring that data is handled safely and efficiently.

**Why do we need File Handling**

* To store data permanently, even after the program ends.
* To access external files like .txt, .csv, .json, etc.
* To process large files efficiently without using much memory.
* To automate tasks like reading configs or saving outputs.
* To handle input/output in real-world applications and tools.

**Modes:**

* "r" - Read - Default value. Opens a file for reading, error if the file does not exist
* "a" - Append - Opens a file for appending, creates the file if it does not exist
* "w" - Write - Opens a file for writing, creates the file if it does not exist
* "x" - Create - Creates the specified file, returns an error if the file exists
* "t" - Text - Default value. Text mode
* "b" - Binary - Binary mode (e.g. images)

**Opening file:**

***file = open ('filename.txt', 'mode')***

file = open (‘abcd.txt’, ‘w’)

opens the file in writing mode

**Reading a File**

[Reading a file](https://www.geeksforgeeks.org/python/how-to-read-from-a-file-in-python/) can be achieved by **file.read()** which reads the entire content of the file. After reading, it’s good practice to close the file to free up system resources.

**Example:** Reading a File in Read Mode (r)

file = open("geek.txt", "r")

content = file.read()

print(content)

file.close()

**Output:**

*Hello world  
GeeksforGeeks  
123 456*

**Writing a File**

In Python, [writing to a file](https://www.geeksforgeeks.org/python/writing-to-file-in-python/) is done using the mode "w". This creates a new file if it doesn’t exist, or overwrites the existing file if it does. The write() method is used to add content. After writing, make sure to close the file.

**Example:** Writing to a file (overwrites if file exists)

with open("geek.txt", "w") as file:

file.write("Hello, Python!\n")

file.write("File handling is easy with Python.")

print("File written successfully")

**Output:**

*Hello, Python!  
File handling is easy with Python.*

**Explanation:**

* "w" mode opens the file for writing (overwrites existing content if the file already exists).
* write() method adds new text to the file.
* When using with, the file closes automatically at the end of the block.

**File method objects**

**Working with binary files**

**Deleting a file:**

import os

if os.path.exists(“example.txt”):

os.remove(“example.txt”)

else:

print(“file does not exists”)

**List comprehensions**

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

**Without List Comprehension (using a loop):**

**Python**

squares = []

for i in range(1, 6):

squares.append(i \* i)

print(squares)

**Output**:

[1, 4, 9, 16, 25]

**🔹 With List Comprehension:**

**Python**

squares = [i \* i for i in range(1, 6)]

print(squares)

**Output**:

[1, 4, 9, 16, 25]

**What's the difference?**

* **Without list comprehension**: More lines of code, less concise.
* **With list comprehension**: One-liner, cleaner and more Pythonic.

**Nested loops in list comprehension:**

Vowles

Flatten 2d list

Convert Celsius to frahrenheit

**Decorators**

Decorators in Python are a powerful and flexible way to modify or extend the behavior of functions or methods **without altering their actual code**. They are essentially functions that wrap another function, **adding functionality before or after the wrapped function** is executed.

**CODE:**

def repeat(n):

def decorator(func):

def wrapper(\*args, \*\*kwargs):

for \_ in range(n):

func(\*args, \*\*kwargs)

return wrapper

return decorator

@repeat(3)

def greet(name):

print(f"Hello, {name}!")

greet("Vidishya")

**OUTPUT:**

Hello, Vidishya!

Hello, Vidishya!

Hello, Vidishya!

=== Code Execution Successful ===

**CODE:**

import datetime

def log(func):

def wrapper(\*args, \*\*kwargs):

print(f"[{datetime.datetime.now()}] {func.\_\_name\_\_} called...")

return func(\*args, \*\*kwargs)

return wrapper

@log

def process\_data(data):

print(f"Processing {data}")

process\_data("User Records")

**OUTPUT:**

[2025-09-08 06:50:58.363311] process\_data called...

Processing User Records

=== Code Execution Successful ===

**Unit testing**

Unit testing is a software testing technique where individual components or functions of a program are tested to ensure they work as expected. Python provides a built-in module called unittest to facilitate this process.

**CODE:**

import unittest

import calculator

class TestCalculator(unittest.TestCase):

def test\_add(self):

self.assertEqual(calculator.add(2, 3), 5)

def test\_subtract(self):

self.assertEqual(calculator.subtract(10, 5), 5)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

WITH DECORATORS

import time

def measure\_time(func):

def wrapper(\*args,\*\*kwargs):

start = time.time()

result = func(\*args, \*\*kwargs)

end = time.time()

print(f"{func.\_\_name\_\_} took {end - start:.4f}seconds")

return result

return wrapper

app.py

**ETL (Extract, Transform, Load)**

ETL is a process used in **data engineering** and **data analysis** to:

1. **Extract** data from various sources
2. **Transform** it into a usable format
3. **Load** it into a database or data warehouse

**Common Python Libraries Used in ETL:**

|  |  |  |
| --- | --- | --- |
| **Step** | **Library** | **Purpose** |
| **Extract** | pandas, requests, sqlalchemy, pyodbc | Read data from CSV, Excel, APIs, or databases |
| **Transform** | pandas, numpy, re | Clean, filter, and modify data |
| **Load** | sqlalchemy, pymysql, psycopg2, boto3 | Save data to databases, cloud storage, or files |

**Example ETL Process (Simple)**

import pandas as pd

from sqlalchemy import create\_engine

# 1. Extract

data = pd.read\_csv("sales.csv")

# 2. Transform

data['Total'] = data['Quantity'] \* data['Price']

# 3. Load

engine = create\_engine('sqlite:///sales.db')

data.to\_sql('sales\_data', con=engine, if\_exists='replace', index=False)

**Other Useful Libraries:**

* **airflow** – For scheduling and managing ETL workflows
* **dask** – For handling large datasets
* **pyarrow / fastparquet** – For working with Parquet files
* **boto3** – For loading data to/from AWS S3
* **beautifulsoup4 / scrapy** – For extracting data from websites

**DAY 8**

**DAY 9**

**Version control system**

A Version Control System (VCS) is a tool used in software development and collaborative projects to track and manage changes to source code, documents, and other files.

A **Version Control System** is important because it helps manage changes to code, supports team collaboration, and ensures safe recovery from errors.

Initialize configure repository

**PRACTICAL**

**DAY 10**

**TDD**

**Test driven development-**

Test-Driven Development (TDD) is a method in software development where the focus is on writing an [**Automation Tests**](https://www.geeksforgeeks.org/software-testing/automation-testing-software-testing/) before writing the actual code for any feature of an application or product. This approach uses short development cycles that repeat to verify the quality and correctness.

TDD simply means a method of coding in which you first write a test, and it fails, then write the code to pass the test of development, and clean up the code.

**Process of Test Driven Development (TDD)**

It is the process in which [**Test Cases**](https://www.geeksforgeeks.org/software-testing/test-case/)are written before the code that validates those cases. It depends on the repetition of a concise development cycle. Test-driven Development is a technique in which automated Unit tests are used to drive the design and free decoupling of dependencies.

The process of Test-Driven Development (TDD) follows a repetitive cycle called **Red-Green-Refactor**.

**Red-green-refactor -**

**A diagram of a test driven development

AI-generated content may be incorrect.**

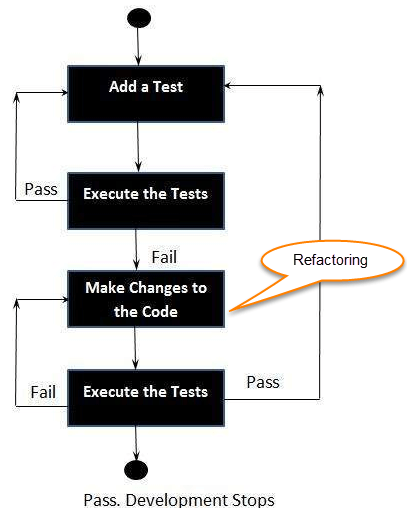
* **Red -** Create a test case and make it fail, Run the test cases
* **Green -** Make the test case pass by any means.
* **Refactor -** Change the code to remove duplicate/redundancy and Refactor code - This is done to remove duplication of code.

**Life cycle-**

**How to perform TDD Test-**

Following steps define how to perform TDD test,

1. Add a test.
2. Run all tests and see if any new test fails.
3. Write some code.
4. Run tests and Refactor code.
5. Repeat.

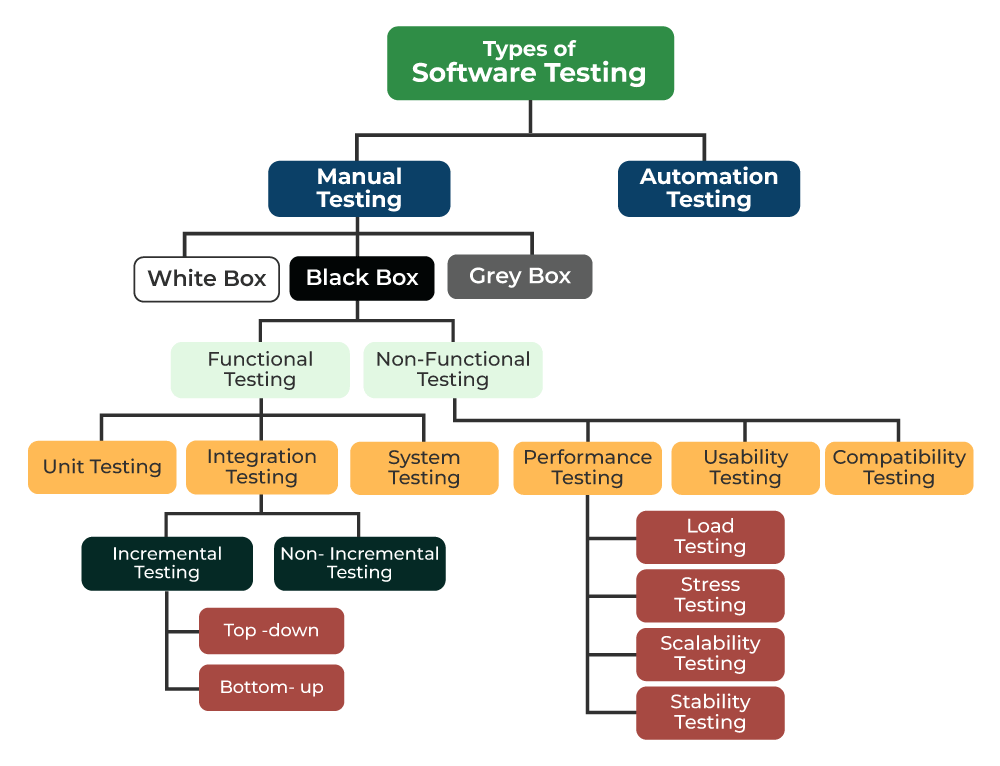


A diagram of a process

AI-generated content may be incorrect.

**Benefits-**

* **Improved Code Quality**: Writing tests first ensures that the code behaves as expected and adheres to requirements.
* **Early Bug Detection**: Bugs are identified and fixed early in the development process, reducing downstream issues.
* **Simpler Design**: TDD encourages writing small, focused units of code, making the system easier to maintain and extend.
* **Faster Feedback Loop**: Automated tests provide immediate feedback, accelerating development cycles.
* **Enhanced Confidence**: Developers can make changes without fear of breaking existing functionality, as tests validate the code continuously.



**What is Unit Testing?**

[**Unit Testing:**](https://www.geeksforgeeks.org/software-testing/unit-testing-software-testing/) refers to the practice of testing the smallest parts of an application, known as "units," in isolation from the rest of the system. In this context, a "unit" is typically a single method or function. Unit testing is essential because it helps detect bugs early in the development process, ensuring that each part of the code behaves as intended before it integrates with other components.

**What is JUnit?**

JUnit is a framework designed to facilitate unit testing in Java. It allows developers to write test cases for individual functionalities in their code and run these tests to check if the actual outcomes match the expected results. When tests fail, JUnit provides detailed feedback, making it easier for developers to debug their code.

**Why JUnit?**

* [**Automation**](https://www.geeksforgeeks.org/software-testing/automation-testing-software-testing/)**:**JUnit supports automated testing, allowing developers to quickly verify that recent changes haven’t disrupted existing functionality. JUnit tests can be integrated into build tools like Maven or Gradle and CI/CD pipelines.
* **Early Bug Detection**: Unit tests help catch bugs early in the development process, reducing the likelihood of introducing new errors as code evolves.
* **Maintainability**: JUnit ensures that changes, bug fixes, or new features don’t inadvertently break other parts of the system. This helps maintain the overall quality of the codebase.
* **Ease of Use**: With simple annotations and methods, JUnit is easy to set up and use, making it accessible even for developers who are new to testing.

**Use case of Junit?**

JUnit is used to do unit testing in Java. A JUnit test case is the set of codes which ensure that our program code works as expected or not. In Java, two types of unit testing are possible, manual testing and automated testing. Manual testing is a special type of testing in which test cases are executed without using any equipment. Unlike manual testing, automated testing is not possible without the support of any equipment. The org.junit package provides several classes and packages, which help us to ensure whether our code provides the expected output or not. These packages and classes are Assert, After, Before and Test, etc.

**ANNOTATIONS?**

**What is TDLC?**

The **Technology Development Life Cycle (TDLC)** is a comprehensive framework that governs the development, deployment, and management of technology solutions. It encompasses hardware, software, and system integration, providing a holistic perspective on technology evolution. The stages of TDLC include:

1. **Planning and Requirements Gathering**: Defining goals and technical requirements.
2. **Design**: Creating system architectures and technical specifications.
3. **Development**: Building and integrating hardware, software, and infrastructure components.
4. **Testing**: Verifying that all components function cohesively and meet quality standards.
5. **Implementation/Deployment**: Deploying the technology in an operational environment.
6. **Maintenance and Support**: Managing updates and resolving issues.
7. **Retirement**: Decommissioning outdated or obsolete technologies.

**What is SDLC?**

The **Software Development Life Cycle (SDLC)** focuses specifically on the structured development and delivery of software applications. Its processes ensure software quality, performance, and reliability. The key stages of SDLC are:

1. **Requirement Analysis**: Identifying user needs and defining system requirements.
2. **System Design**: Developing detailed designs for the software architecture.
3. **Implementation (Coding)**: Writing and compiling software code.
4. **Testing**: Ensuring the software meets requirements through various testing methodologies.
5. **Deployment**: Delivering the software to end-users or a production environment.
6. **Maintenance**: Managing bugs, updates, and new feature requests.

* **When to Use TDLC:** Ideal for projects requiring the development and integration of multiple technologies, such as enterprise-wide IT infrastructure or IoT solutions.
* **When to Use SDLC:** Best suited for software-focused projects, such as developing a mobile app or enterprise software platform.

[JUnit 5 User Guide](https://docs.junit.org/current/user-guide/)

| **Behavior Driven Development** | **Test Driven Development** |
| --- | --- |
| Behavior Driven Development is a development technique which focuses more on a software application's behavior. | Test Driven Development is a development technique which focuses more on the implementation of a feature of a software application/product. |
| In BDD the participants are Developers, Customer, QAs. | In TDD the participants are developers. |
| Mainly it creates an executable specification that fails because the respective feature doesn't exist, then writing the simplest code that can make the specification pass and as a result we get the required behavior implemented in the system. | Mainly it refers to write a test case that fails because the specified functionality doesn't exist and after that update the code that can make the test case pass and as a result we get the feature implemented in the system. |
| Its main focus is on system requirements. | Its main focus is on unit test. |
| In BDD the starting point is a scenario. | In TDD the starting point is a test case. |
| It is a team methodology. | It is a development practice. |
| Here language used to write behavior/scenarios is simple English language. | Here language is used is similar to the one used for feature development like programming language. |
| In BDD collaboration is required between all the stakeholders. | In TDD collaboration is required only between the developers. |
| It is a good approach for project development which are driven by user actions. | It is a good approach for projects which involve API and third-party tools. |

**DAY 11**

**WEBSERVICES**

**WHAT ARE WEBSERVICES, ITS NEED, ADV & DISADV-**

Web services are a powerful technology that enables different applications to communicate and exchange data over the internet. They rely on standardized protocols like SOAP, WSDL, and UDDI to create interoperable applications.

**WSDL v/s UDDI-**

**WSDL (Web Services Description Language)**

WSDL is an XML-based language used to describe web services and their operations, inputs, and outputs. It provides a way for developers to understand how to interact with a web service without having to access its source code.

**Example: WSDL Document**

A WSDL document includes the following components:

* **Definitions:** Define the data types, messages, and operations.
* **Types:** Define the data types used in the messages.
* **Messages:** Define the structure of the data exchanged.
* **Port Type:** Define the operations and their input/output messages.
* **Binding:** Specify the protocol and encoding style for the operations.
* **Service:** Define the network address and binding information.

**UDDI (Universal Description, Discovery, and Integration)**

* UDDI is a registry standard for web services, allowing businesses to publish and discover services over the internet. It provides a centralized directory where developers can search for and obtain information about available web services.

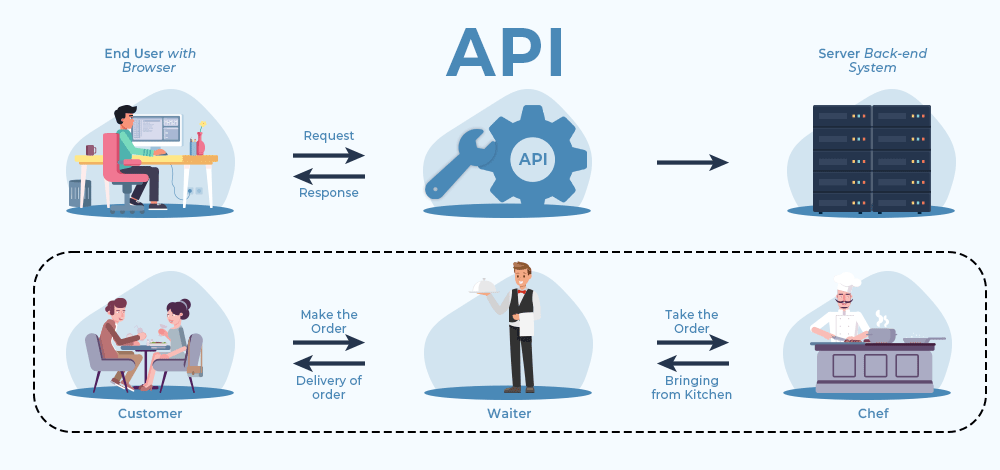
**SOAP v/s REST-**

|  |  |  |
| --- | --- | --- |
| SOAP | | REST |
| Stands for | | Simple Object Access Protocol | Representational State Transfer |
| What is it? | | SOAP is a protocol for communication between applications | REST is an architecture style for designing communication interfaces. |
| Design | | SOAP API exposes the operation. | REST API exposes the data. |
| Transport Protocol | | SOAP is independent and can work with any transport protocol. | REST works only with HTTPS. |
| Data format | | SOAP supports only XML data exchange. | REST supports XML, JSON, plain text, HTML. |
| Performance | | SOAP messages are larger, which makes communication slower. | REST has faster performance due to smaller messages and caching support. |
| Scalability | | SOAP is difficult to scale. The server maintains state by storing all previous messages exchanged with a client. | REST is easy to scale. It’s stateless, so every message is processed independently of previous messages. |
| Security | | SOAP supports encryption with additional overheads. | REST supports encryption without affecting performance. |
| Use case | | SOAP is useful in legacy applications and private APIs. | REST is useful in modern applications and public APIs. |

**API (Application Programming Interface)-**

An API is a set of rules that allow different software applications to communicate with each other. Think of it as a **bridge** that connects two systems—such as a client and a server—and enables them to work together seamlessly.

To understand it better, imagine you're at a restaurant: the **waiter** (API) takes your order (request), gives it to the **chef** (server), and then brings the prepared food (response) back to your table. Similarly, when you search for a course on a website, your request goes through an API, which then fetches the data from the database and sends it back as a response.



**Why Do We Need APIs?**

APIs help developers to create software programs more easily. Instead of writing complex code from scratch, they can call APIs that already provide the functions they need. For example, if a developer wants to display a weather report, they can use an API to get the data instead of creating the entire system to gather weather data themselves.

APIs are also crucial in building modern websites, where heavy data transfers happen between the client (user) and the server.

**How Do APIs Work?**

APIs work in a simple step-by-step process:

* **Request:** A client (user) sends a request through the API's URI (Uniform Resource Identifier).
* **Processing:** The API forwards the request to the server.
* **Response:** The server processes the request and sends the response back to the API.
* **Delivery:** The API returns the server's response to the client.

Think of this as a client-server architecture: the client sends a request, the server processes it, and the API acts as the messenger.curity threats. To provide additional security layers to the data, HTTP headers, query string parameters, or cookies are used.

**How to Create APIs?**

Creating an API is an easy task unless you are very well clear on the basic concepts. It's an iterative process (based on feedback) that just includes a few easy steps:

* Plan your goal and the intended users
* Design the API architecture
* Develop (Implement the code) and Test API
* Monitor its working and work on feedback

**Advantages of APIs**

* **Efficiency:**API produces efficient, quicker, and more reliable results than the outputs produced by human beings in an organization.
* **Flexible delivery of services:**API provides fast and flexible delivery of services according to developers' requirements.
* **Integration:**The best feature of API is that it allows the movement of data between various sites and thus enhances the integrated user experience.
* **Automation:**As API makes use of robotic computers rather than humans, it produces better and more automated results.
* **New functionality**: While using API the developers find new tools and functionality for API exchanges.

**Disadvantages of APIs**

* **Cost:**Developing and implementing API is costly at times and requires high maintenance and support from developers.
* **Security issues:**Using API adds another layer of surface which is then prone to attacks, and hence the security risk problem is common in APIs.

**(TYPES)**

**HTTP METHODS**

HTTP defines a set of request methods to indicate the purpose of the request and what is expected if the request is successful. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. Each request method has its own semantics, but some characteristics are shared across multiple methods, specifically request methods can be safe, idempotent, or cacheable.

[**GET**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/GET)

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should not contain a request [content](https://developer.mozilla.org/en-US/docs/Glossary/HTTP_Content).

[**HEAD**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/HEAD)

The HEAD method asks for a response identical to a GET request, but without a response body.

**POST**

The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

[**PUT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/PUT)

The PUT method replaces all current representations of the target resource with the request [content](https://developer.mozilla.org/en-US/docs/Glossary/HTTP_Content).

[**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/DELETE)

The DELETE method deletes the specified resource.

[**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/CONNECT)

The CONNECT method establishes a tunnel to the server identified by the target resource.

[**OPTIONS**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/OPTIONS)

The OPTIONS method describes the communication options for the target resource.

[**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/TRACE)

The TRACE method performs a message loop-back test along the path to the target resource.

[**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods/PATCH)

The PATCH method applies partial modifications to a resource.

**STATUS CODE-**

HTTP response status codes indicate whether a specific [HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP) request has been successfully completed. Responses are grouped in five classes:

1. [Informational responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#informational_responses) (100 – 199)
2. [Successful responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#successful_responses) (200 – 299)
3. [Redirection messages](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#redirection_messages) (300 – 399)
4. [Client error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#client_error_responses) (400 – 499)
5. [Server error responses](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#server_error_responses) (500 – 599)

* [100 Continue](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/100)
* [101 Switching Protocols](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/101)
* [102 Processing](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/102)
* [103 Early Hints](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/103)
* [200 OK](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/200)
* [201 Created](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/201)
* [202 Accepted](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/202)
* [203 Non-Authoritative Information](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/203)
* [204 No Content](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/204)
* [205 Reset Content](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/205)
* [206 Partial Content](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/206)
* [207 MultiStatus](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/207)
* [208 Already Reported](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/208)
* [226 IM Used](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/226)
* [300 Multiple Choices](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/300)
* [301 Moved Permanently](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/301)
* [302 Found](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/302)
* [303 See Other](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/303)
* [304 Not Modified](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/304)
* [307 Temporary Redirect](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/307)
* [308 Permanent Redirect](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/308)
* [400 Bad Request](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/400)
* [401 Unauthorized](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/401)
* [402 Payment Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/402)
* [**403 Forbidden**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/403)
* [**404 Not Found**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/404)
* [405 Method Not Allowed](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/405)
* [406 Not Acceptable](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/406)
* [407 Proxy Authentication Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/407)
* [408 Request Timeout](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/408)
* [409 Conflict](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/409)
* [410 Gone](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/410)
* [411 Length Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/411)
* [412 Precondition Failed](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/412)
* [413 Content Too Large](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/413)
* [414 URI Too Long](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/414)
* [415 Unsupported Media Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/415)
* [416 Range Not Satisfiable](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/416)
* [417 Expectation Failed](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/417)
* [418 I'm a teapot](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/418)
* [421 Misdirected Request](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/421)
* [422 Un-processable Content](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/422)
* [423 Locked](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/423)
* [424 Failed Dependency](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/424)
* [425 Too Early](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/425)
* [426 Upgrade Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/426)
* [428 Precondition Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/428)
* [429 Too Many Requests](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/429)
* [431 Request Header Fields Too Large](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/431)
* [451 Unavailable for Legal Reasons](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/451)
* [500 Internal Server Error](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/500)
* [501 Not Implemented](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/501)
* [**502 Bad Gateway**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/502)
* [**503 Service Unavailable**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/503)
* [**504 Gateway Timeout**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/504)
* [505 HTTP Version Not Supported](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/505)
* [506 Variant Also Negotiates](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/506)
* [507 Insufficient Storage](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/507)
* [508 Loop Detected](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/508)
* [510 Not Extended](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/510)
* [511 Network Authentication Required](https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status/511)

**PAYLOAD-**

**JSON-**

JSON has a straightforward syntax with **key-value pairs**, making it easy to read and write for humans. JSON is lightweight compared to XML, resulting in faster parsing and smaller data payloads.

*{"Geeks":[*

*{ "firstName":"Vivek", "lastName":"Kothari" },*

*{ "firstName":"Suraj", "lastName":"Kumar" },*

*{ "firstName":"John", "lastName":"Smith" },*

*{ "firstName":"Peter", "lastName":"Gregory" }*

*]}*

**XML-**

XML is highly extensible, allowing developers to define custom tags and structures suited to specific needs. Strong support for XML Schema Definition (XSD), enabling rigorous validation and data integrity checks.

*<Geeks>*

*<Geek>*

*<firstName>Vivek</firstName> <lastName>Kothari</lastName>*

*</Geek>*

*<Geek>*

*<firstName>Suraj</firstName> <lastName>Kumar</lastName>*

*</Geek>*

*<Geek>*

*<firstName>John</firstName> <lastName>Smith</lastName>*

*</Geek>*

*<Geek>*

*<firstName>Peter</firstName> <lastName>Gregory</lastName>*

*</Geek>*

*</Geeks>*

|  |  |
| --- | --- |
| **JSON** | **XML** |
| It is [JavaScript Object Notation](https://www.geeksforgeeks.org/javascript/javascript-json/) | It is [Extensible markup language](https://www.geeksforgeeks.org/html/xml-basics/) |
| It is based on JavaScript language. | It is derived from [SGML](https://www.geeksforgeeks.org/html/what-is-sgml/). |
| It is a way of representing objects. | It is a markup language and uses tag structure to represent data items. |
| It does not provides any support for namespaces. | It supports [namespaces](https://www.geeksforgeeks.org/javascript/javascript-namespace/). |
| It supports array. | It doesn't supports [array](https://www.geeksforgeeks.org/dsa/array-data-structure-guide/). |
| Its files are very easy to read as compared to XML. | Its documents are comparatively difficult to read and interpret. |
| It doesn't use end tag. | It has start and end tags. |
| It is less secured. | It is more secured than JSON. |
| It doesn't supports comments. | It supports comments. |
| It supports only [UTF-8 encoding](https://www.geeksforgeeks.org/dsa/understanding-character-encoding/). | It supports various encoding. |

**DAY 12**

**What is Kubernetes (K8s)?**

**Kubernetes** is an **open-source platform** used to **automate the deployment, scaling, and management of containerized applications**.

* Developed by Google, now maintained by the **Cloud Native Computing Foundation (CNCF)**.
* It helps manage **containers** (like Docker) across a **cluster of machines**.

**Why Use Kubernetes?**

|  |  |
| --- | --- |
| **Feature** | **Benefit** |
| **Scalability** | Automatically scale apps up/down based on demand |
| **Self-healing** | Restarts failed containers, replaces them |
| **Load balancing** | Distributes traffic evenly |
| **Rolling updates** | Deploy new versions without downtime |
| **Resource management** | Efficient use of CPU, memory, etc. |

**Key Concepts in Kubernetes**

**1. Cluster**

A group of machines (nodes) managed by Kubernetes.

* **Master Node**: Controls the cluster.
* **Worker Nodes**: Run the applications.

**2. Pod**

* The **smallest unit** in Kubernetes.
* A pod contains **one or more containers** that share storage and network.

**3. Deployment**

* Defines how to **create and manage pods**.
* Handles **updates**, **rollbacks**, and **replicas**.

**4. Service**

* Exposes your app to the network.
* Types: ClusterIP, NodePort, LoadBalancer, Ingress.

**5. Namespace**

* Used to **organize resources** in a cluster.
* Like folders for your Kubernetes objects.

**6. ConfigMap & Secret**

* **ConfigMap**: Stores non-sensitive configuration data.
* **Secret**: Stores sensitive data like passwords or API keys.

**7. Volume**

* Used to store **persistent data**.
* Unlike containers, volumes **don’t get deleted** when a pod restarts.

**DAY 13**

**What Is Kubernetes**

Kubernetes is a platform for managing [containers](https://www.redhat.com/en/topics/containers), which bundle the code, configuration, and dependencies of an application, allowing it to run as an isolated process with its own resources. Each application gets its own container or multiple containers, which are grouped into Kubernetes pods.

**Architecture-**

**Control Plane Components**

The **Control Plane** is responsible for maintaining the desired state of the cluster and managing the worker nodes.

1. **kube-apiserver**: The API server is the central hub of the Kubernetes cluster, exposing the Kubernetes API. It handles all API requests, manages authentication and authorization, and processes API requests for Kubernetes objects like pods and services.
2. **etcd**: A distributed key-value store that acts as the brain of the Kubernetes cluster. It stores all configurations, states, and metadata of Kubernetes objects. It ensures strong consistency and availability using the Raft consensus algorithm.
3. **kube-scheduler**: Responsible for scheduling pods on worker nodes based on resource requirements, affinity, taints, and other constraints. It uses filtering and scoring operations to select the best node for a pod.
4. **kube-controller-manager**: Runs various controllers that ensure the desired state of the cluster. Controllers include the deployment controller, replicaset controller, and node controller, among others.
5. **cloud-controller-manager**: Integrates Kubernetes with cloud provider APIs, managing cloud-specific resources like instances, load balancers, and storage volumes. It contains controllers for node, route, and service management.

**Worker Node Components**

Worker nodes are responsible for running containerized applications.

1. **kubelet**: An agent that runs on each node, ensuring that containers are running in a pod. It interacts with the container runtime to manage the lifecycle of containers and reports node and pod status to the API server.
2. **kube-proxy**: A network proxy that runs on each node, implementing the Kubernetes Service concept. It creates network rules to route traffic to the appropriate pods, supporting load balancing and service discovery.
3. **Container Runtime**: The software component required to run containers. Kubernetes supports multiple container runtimes like containerd, CRI-O, and Docker, which are compliant with the Container Runtime Interface (CRI).

**Addon Components**

Kubernetes clusters often require additional components to be fully operational.

1. **CNI Plugin**: Provides networking capabilities for pods, enabling communication between pods across nodes. Examples include Calico, Flannel, and Weave Net.
2. **CoreDNS**: Acts as a DNS server within the Kubernetes cluster, enabling DNS-based service discovery.
3. **Metrics Server**: Collects performance data and resource usage of nodes and pods in the cluster.
4. **Kubernetes Dashboard**: A web-based UI for managing and troubleshooting applications running in the cluster.

* **Advantages of Kubernetes**

**Resource Efficiency**

Kubernetes helps optimize the use of resources by scheduling containers to run on the most appropriate node based on their resource requirements.

**Self-Healing**

Kubernetes can detect and respond to failures in the application by automatically restarting containers, rolling back deployments, and rescheduling workloads.

**Service Discovery and Load Balancing**

Kubernetes provides a built-in service discovery and load balancing mechanism, which allows developers to easily expose their application services and manage traffic between them.

**Open-Source**

Kubernetes is open-source, meaning it is free to use and has a large community of developers contributing to its development.

**Fault Tolerance**

Kubernetes provides robust fault tolerance features, such as automatic failover and self-healing.

**Flexibility**

Kubernetes provides a high degree of flexibility in deploying and managing applications.

* **Disadvantages of Kubernetes-**

**Complexity**

Kubernetes can be complex to set up and manage, particularly for small teams or organizations with limited resources.

**Performance Overhead**

Kubernetes introduces some overhead in terms of CPU and memory usage, which can impact application performance.

**Security**

Kubernetes has a complex security model, and it is important to properly configure and secure the platform to ensure that applications are protected against potential security threats.

**Dependency on External Services**

Kubernetes relies on external services such as container registries, network storage providers, and load balancers.

**Infrastructure Requirements**

Kubernetes requires a robust and reliable infrastructure to run on**.**

**Complexity of Networking**

Kubernetes provides a highly flexible and configurable networking model, but this can also make networking more complex.

**What is a pod**

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

A Pod is a group of one or more [containers](https://kubernetes.io/docs/concepts/containers/), with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled and run in a shared context.

**DAY 14**

**What is selenium, adv and dis-adv**

Selenium is a free, open-source test execution automation platform for web applications. It implements itself using a browser-specific driver that accepts and transmits commands to the browser.

**Advantages:**

1. Free and Open Source

2. Supports a Wide Number of Systems, Languages, and Browsers

3. Integrated with CI/CD

4. Run Tests Concurrently

**Disadvantages:**

1. Supports Web Applications Only

2. Not Having an Image Comparison

3. Selenium cannot Create Reports

4. No Technical Support will be Provided

**What is selenium architecture**

**Architecture of Selenium 3 WebDriver**

In the architecture of Selenium 3 WebDriver, a Selenium test script uses a client library (such as Python, Java, or any other supported language) to send commands that interact with a web browser. These commands are converted into a JSON format by the client library. Once the commands are in JSON format, they are sent through an HTTP request to the browser driver.

The browser driver then decodes the JSON commands and performs the necessary actions on the web page, such as clicking buttons or entering text. The browser follows these commands as if a user were interacting with the page, allowing us to automate the testing process effectively.

Architecture of Selenium 3 WebDriver

**Components of Selenium 3 WebDriver:**

Selenium WebDriver is a tool that helps automate web applications. Its architecture is built to allow easy communication between different parts. Understanding the key components of Selenium 3 WebDriver makes it easier to see how automation scripts interact with browsers for smooth testing.

1. **Selenium Client Library**
2. **JSON Wire Protocol over HTTP**
3. **Browser Drivers**
4. **Real Browsers**

**Architecture of Selenium 4 WebDriver**

Selenium 4 brings significant improvements to the architecture, often with the introduction of the [W3C WebDriver Protocol](https://www.geeksforgeeks.org/computer-organization-architecture/w3c-full-form/). This protocol standardizes interactions between the purchaser and server, selling higher compatibility and consistency across one-of-a-kind implementations. Moreover, Selenium 4 affords a better guide for present-day net technology and progressed overall performance.

The architecture of Selenium 4 WebDriver has made a key change compared to Selenium 3 which is a communication protocol. Like Selenium 3, Selenium 4 offers client libraries for various programming languages, which help WebDriver interact with the browser. WebDriver W3C protocol is the major change in Selenium 4 as it completely replaces JSON Protocol which was in Selenium 3. The WebDriver W3C Protocol is defined by the World Wide Web Consortium (W3C) that ensure better compatibility and stability on different browsers and client libraries.

Selenium WebDriver 4 Architecture

**Components of Selenium 4 WebDriver**

Selenium 4 is an updated version of a tool used to automate web applications. Its design makes it easier and faster for automation scripts to communicate with browsers, improving the overall testing process.

1. **Selenium Client Library**
2. **WebDriver W3C Protocol**
3. **Browser Drivers**
4. **Real Browsers**

**What is web-driver**

**Selenium Webdriver** is mainly used to execute the scripts according to the browser we are using. Selenium is a powerful tool for controlling web browsers through programs.

WebDriver offers a programming interface to interact with web browsers, allowing developers and testers to automate browser actions seamlessly.

**DAY 15**

**WebElement**:

Represents an HTML element on a web page, allowing interaction as follows-  
1. The **sendKeys** method is used to send any keyboard key to an element on the webpage. This command is used to type something in a text box/input and also press any keys like enter. The command expects an argument of the keys to be sent and does not return anything.

2. **isSelected() Command**

This method tells if an element is selected or not. It returns a boolean value of "true" if element is selected and "false" if it is not selected. It can be used to check if some link or radio button or checkbox is selected or not.

3. **isEnabled() Command**

This method tells if the element is enabled or not and returns a boolean value depending on that. It returns "true" if the element is enabled.

4. **getText() Command**

This command gives us the text of the particular element visible on the webpage.

5. **click() Command**

The click command is used to click on the element. It can be used to click a link, button, or a checkbox.

6. **getSize() Command**

This command gives us the size of the specific element in terms of its height and width in pixels. It returns the size as the dictionary of these attributes.

**Polling period in fluent wait?**

In Selenium WebDriver, wait commands are essential to ensure that elements are present, visible, and interactable before performing actions. Different types of waits help to handle dynamic elements, such as those that load asynchronously.

There are three main wait strategies in Selenium are Implicit Wait, Explicit Wait, and Fluent Wait.

**1. Implicit Wait**

An Implicit Wait is a global timeout that Selenium will apply to all element searches. This means that Selenium will wait for the specified amount of time before throwing a *NoSuchElementException* if an element is not found in the DOM.

Once you set an implicit wait, it remains active for the entire session of the WebDriver instance.

**2. Explicit Wait**

An Explicit Wait allows you to wait for a specific condition to be met before proceeding with the next step in the test. Unlike implicit waits, you can use this wait for individual elements with specific conditions, such as the element being visible or clickable.

Explicit waits are more precise and efficient, as they only apply to the specific conditions you define.

**3. Fluent Wait**

A Fluent Wait is a more flexible version of the explicit wait. It allows you to configure how frequently Selenium checks the condition (polling interval), how long it waits before timing out (maximum wait time), and which exceptions it should ignore (like *NoSuchElementException*).

This is ideal for situations where elements load unpredictably or asynchronously.