TCS HR

1) **About TCS**

Tata Consultancy Services (TCS) is a global IT services, consulting, and business solutions company headquartered in Mumbai, India. It is part of the Tata Group and is one of the largest IT services companies in the world. TCS offers services including software development, enterprise solutions, cloud computing, and IT infrastructure management.

2) **Where is the headquarters of TCS?**

TCS is headquartered in Mumbai, Maharashtra, India.

Its registered office is in Bombay House, the HQ of Tata Group.

3) **Chairman of TCS**

As of 2025, **Natarajan Chandrasekaran** is the Chairman of Tata Sons, the holding company of Tata Group, which includes TCS. He has a significant influence over TCS as part of the Tata Group.

4) **Current CEO of TCS**

The current CEO and Managing Director (MD) of Tata Consultancy Services (TCS) is **K. Krithivasan.** He took over in June 2023 after **Rajesh Gopinathan** stepped down.

5) **1st CEO of TCS**

The first CEO of Tata Consultancy Services (TCS) was **Faqir Chand Kohli**. He is also known as the "Father of the Indian Software Industry".

6) **Founder of TCS**

TCS was founded in 1968 by the Tata Group under the leadership of **J.R.D. Tata and with F.C. Kohli** as the first CEO who played a pivotal role in building the company.

7) **Values of TCS**

The values of TCS, such as “Leading Change, Integrity, Respect for the Individual, Excellence, and Learning and Sharing,” reflect the beliefs and culture I admire.

8) **Parent Company**

Tata sons (principal investment holding company and promoter of Tata companies)/Tata Group

9) **What is TCS’s revenue?**

For Financial Year (FY) 2024–25, TCS earned over $29 billion USD in revenue, making it India’s largest IT services company.

10) **Locations**

As of 2024, Tata Consultancy Services (TCS) operates from over 50 countries and more than 500 offices globally.

In India, TCS has multiple delivery centers in major cities, including Mumbai, Bangalore, Chennai, and Hyderabad.

North America, Latin America, Europe, UK & Ireland, India, Asia Pacific, Australia & New Zealand, Middle East & Africa, Japan…5 continents. 150 nationalities. 612,000+ associates.

11) **What is the global presence of TCS?**

TCS operates in 46+ countries with 285+ offices and 147+ delivery centers worldwide.

12) What training and development programs does TCS offer?

TCS is known for:

- **Initial Learning Program (ILP)** for freshers
- **Digital Learning Hub** for self-paced learning
- **TCS Elevate** for structured upskilling

It promotes **continuous learning** and encourages certifications.

13) Industries TCS Working in?

TCS provides services to diverse industries including:

- Banking and Financial Services
- Insurance
- Retail and Consumer Goods
- Telecom
- Manufacturing
- Healthcare and Life Sciences
- Energy and Utilities
- Travel, Transportation, and Hospitality
- Technology and Media

14) How many employees does TCS have?

TCS has over 600,000 employees (as of 2025), representing 150+ nationalities across 50+ countries.

15) What awards and recognitions has TCS received?

TCS has been featured in:

- Top Employers Global Certification
- Forbes Global 2000 list
- Brand Finance's Most Valuable IT Services Brand

16) Awards and achievements of TCS.

- TCS's most recent accomplishment is surpassing $30 billion in annual revenue and securing its position as the world's second-largest IT services brand. This milestone was achieved in the financial year 2025, as indicated by their financial results.
- In 2024, TCS was recognized as a **Leader in Gartner's Magic Quadrant for Application Services**.
- TCS was named a Top Employer globally by the Top Employers Institute for multiple consecutive years.

17) Do you know about ongoing project in TCS?

TCS is involved in numerous projects across various industries, including e-governance, manufacturing, and banking. Some notable recent projects include the development of AI-powered solutions with Microsoft for the manufacturing sector, digitization of HR operations for Nokia, and AI-driven aisle optimization for Woolworths. They are also heavily involved in India's e-governance initiatives, such as managing passport issuance systems and digitalizing India Post.

18) Why TCS? /

Why do you want to work at TCS? /

what motivates to choose this company?

- I want to join TCS because of its reputation as a leading technology company and its global presence.
- The company's commitment to continuous learning and employee growth aligns with my career goals.
- The opportunity to work on **diverse projects across multiple industries** offers broad exposure.
- TCS is one of the top IT companies globally, known for its innovation, stability, and strong global presence. I'm motivated to join TCS because it offers diverse project exposure and continuous learning opportunities.

19) What does TCS do?

Tata Consultancy Services (TCS) is an IT services, consulting, and business solutions company. They help businesses transform their operations by providing various IT solutions, including application development and management, digital transformation, AI, data and cloud services, and engineering services.

TCS partners with clients across multiple industries and countries to deliver customized solutions that meet their unique needs.

20) Why should we hire you?

I bring fresh energy, a strong work ethic, and a willingness to learn and adapt, which will help me add value to your team. I am a quick learner with strong problem-solving skills. I'm motivated to contribute to innovative projects and grow with TCS.

21) Tell me 2 reasons why we should not hire you.

22) Describe a time when you worked in a team.

23) Tell me about a challenging project you worked on

24) Tell me something that is not mentioned in your resume.

25) Difference between hard work and Smart work.

Hard work involves dedicating significant effort, time, and energy to achieve goals, often through persistence and dedication., while smart work means using strategies to achieve better results efficiently. Smart work saves time by focusing on priorities.

26) Tell us the financial condition of India.

India's economy is growing steadily with strong sectors like IT and manufacturing. However, challenges like inflation and unemployment remain areas of focus.

27) Why not opting for higher studies?

I want to gain practical industry experience first to apply my knowledge. I plan to pursue higher studies after building a solid professional foundation.

28) Who inspired you to become an engineer?

My uncle, who is a software engineer working in the US, has been a big inspiration for me. Seeing his journey motivates me to pursue a career in software engineering and aim for global opportunities.

29) What is your strength and weakness?

Strength: I'm highly motivated, a quick learner, and I work well in teams, which helps me adapt to new challenges effectively. Or I am a quick learner and a great team player.

Weakness: I sometimes overthink details, but I'm working on balancing perfection with efficiency. Or I can concentrate on one thing at a time

30) Extra- curricular activities

I've been involved in extra-curricular activities like volunteering during college events and participating in cultural programs. These helped me build confidence and improve my communication and teamwork skills.

31) co-curricular activities?

I actively participate in extra-curricular activities such as

- *being a member of the ISTE/IETE student chapter,*
- *taking part in technical workshops and coding competitions,*
- *and volunteering in college events.*

These activities have helped me improve my teamwork, leadership, and communication skills outside of academics.

32) **Are you a team player?**

Yes, I believe teamwork leads to better solutions. I communicate openly and support my team to achieve common goals.

33) Tell me about the recent trends in IT industry.

AI, cloud computing, and cybersecurity are rapidly evolving. There's also a big push towards automation and edge computing.

34) What is your aim?

My aim is to become a skilled software engineer and contribute to impactful projects. I want to keep learning and grow professionally.

35) What were you doing for past 6 months?

Over the past 6 months, I've been actively preparing for placement opportunities. I've focused on improving my technical skills, practicing aptitude and communication, and working on mini-projects to strengthen my resume.

36) Tell me about your dream job.

My dream job is one where I can solve challenging problems using technology and continuously learn new skills. I want to contribute to projects that make a real impact.

37) Where is your native place and what is it famous for?

My native place is Tirunelveli, in Tamil Nadu. It is well known for its rich cultural heritage, scenic spots like the Courtallam waterfalls, and especially for the famous Tirunelveli halwa, a traditional sweet.

38) What is your long and short term goal.

Short term, I want to develop strong technical skills and contribute effectively to my team. Long term, I aim to lead projects and innovate in my field.

39) Are you ok with relocation?

Yes, I am open to relocation as it offers new experiences and growth opportunities. I'm adaptable and excited to work anywhere.

40) Can you work at any location in india?

Yes, I am willing to work at any location in India as per the company's requirements. I believe it will broaden my exposure and skills.

41) Ready for night shift?

Yes, I am flexible and willing to work night shifts if required. I understand it may be necessary for project deadlines or client needs.

42) How do you handle tight deadlines?

I stay organized by breaking tasks into smaller steps and focusing on priorities. I also communicate proactively to manage expectations and seek help if needed.

43) How do you prioritize tasks?

I list tasks based on urgency and impact, tackling high-priority items first. I review and adjust priorities regularly as new tasks come up.

44) Describe a time you received constructive critics.

During my internship, my mentor suggested I improve my code documentation. I took the feedback positively and started focusing on writing clearer, more maintainable code. Similarly, during my third-year project, I was advised to improve my report writing. I worked on making it more structured and clear, which really helped me present my work better.

45) Have you got any other placement than this?

No, I haven't received any other placement offers yet. I've been very focused on preparing for opportunities that align with my career goals, and I see TCS as an excellent place to start my professional journey due to its global presence, structured learning environment, and diverse projects.

I've participated in a few processes, and each has been a learning experience. I've used the feedback to improve my skills continuously. I see TCS as a great opportunity, and I feel confident in my preparation now."

46) What motivates you?

I'm motivated by challenges that help me grow and learn new skills. Delivering successful projects and making an impact keeps me driven.

47) If your thoughts conflict with your co-worker, how would you handle the situation.

I would listen carefully to their perspective and communicate mine respectfully. Finding common ground or compromising is key to teamwork.

48) Describe your ability to work under pressure.

I remain calm and organized during pressure by prioritizing tasks and managing time efficiently. Staying focused helps me meet deadlines without compromising quality.

49) About family. / Tell me about your family background.

I come from a supportive family that values education and hard work. They have always encouraged me to pursue my goals and grow professionally.

50) If you're offered a lower role/profile, will you still join?

Yes, I believe every role offers learning opportunities. I am ready to start wherever needed and prove my capabilities.

51) What specific role you want to work in TCS?

I want to work as a software developer where I can apply my coding skills and contribute to innovative projects.

52) Why do you want this job?

This job aligns with my skills and career goals. I'm excited to work in a reputed company like TCS and grow with challenging assignments.

53) Where do you see yourself in 5 years?

I see myself growing into a more skilled professional, taking on greater responsibilities, and contributing meaningfully to the company.

In 5 years, I see myself as a skilled professional contributing to impactful projects and growing into a leadership role.

54) Describe yourself in 3 words.

Dedicated, quick-learner, team-player.

55) If you are rejected today, then what will you do.

I will seek feedback to improve and keep enhancing my skills. I believe persistence is key to success.

56) Do you have any Questions for me?

What are the skills needs to be focused before joining the organization?

Yes, could you please tell me about the growth opportunities in this role? Also, what's the team culture like?

57) As an ECE student, why choose software?

I chose software because I enjoy programming and finding solutions to problems. Also, the tech keeps evolving, so there's always something interesting to work on and learn.

58) "Why do you want to join TCS?"

"I want to join TCS because it's one of the leading IT services companies globally, known for its strong values like integrity, continuous learning, and innovation. I admire TCS's inclusive and growth-oriented work culture, and its large-scale projects across various domains and geographies.

TCS also provides excellent training programs like Elevate and access to platforms like Digital Learning Hub, which align perfectly with my goal to build a strong foundation in technology and keep upgrading my skills. I'm also drawn to the opportunity for global exposure and career mobility that TCS offers.

Overall, I see TCS as a place where I can grow both personally and professionally, while contributing to meaningful digital transformation projects."

MR

Situation,project,puzzle

1) **What's your response when you are unable to meet a deadline or submit quality work in the given timeframe?**

   "If I realize I may miss a deadline or quality is at risk, I inform my mentor or team early. I try to prioritize tasks, seek help if needed, and do my best to deliver what's possible within time. Clear communication and accountability are key."

2) **How will you react to last-minute changes to a technical assignment you spent weeks completing?**

   "I would stay calm and flexible. I understand that project requirements can change. I'll analyze the new changes quickly, adjust my work accordingly, and coordinate with the team to meet the new expectations on time."

3) **How do you overcome challenges?**

   "I break the challenge into smaller tasks, understand what's causing the issue, and approach it step by step. I also take feedback, ask for guidance if needed, and stay positive until I find a solution."

4) **You are a leader of a team. How would you motivate your team to work?**

   "I would keep communication open, recognize each member's strengths, and assign tasks accordingly. I'd set clear goals, appreciate good work, and support the team when they face difficulties to keep morale high."

5) **If you cut a cake into 3 parts, but you have to give 8 pieces out of it.**

   2 vertical cut,1 horizontal cut.8 pieces in 2 layers

TECH HR

1) Newtons Laws

First Law (Law of Inertia):
An object remains at rest or in uniform motion unless acted upon by an external force.

Second Law:
The force acting on an object is equal to the mass of the object multiplied by its acceleration.
Formula: `F = m × a`

Third Law:
For every action, there is an equal and opposite reaction.

2) Normalization & types

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

Types:

- **1NF (First Normal Form):** Removes duplicate columns; ensures atomic values.
- **2NF (Second Normal Form):** Removes partial dependency; applicable only if 1NF is met.
- **3NF (Third Normal Form):** Removes transitive dependency.
- **BCNF (Boyce-Codd Normal Form):** A stricter version of 3NF.

1. First Normal Form (1NF): Eliminating Duplicate Records

A table is in 1NF if it satisfies the following conditions:

- All columns contain atomic values (i.e., indivisible values).
- Each row is unique (i.e., no duplicate rows).
- Each column has a unique name.
- The order in which data is stored does not matter.

Example of 1NF Violation: If a table has a column "Phone Numbers" that stores multiple phone numbers in a single cell, it violates 1NF. To bring it into 1NF, you need to separate phone numbers into individual rows.

2. Second Normal Form (2NF): Eliminating Partial Dependency

A relation is in 2NF if it satisfies the conditions of 1NF and additionally. No partial dependency exists, meaning every non-prime attribute (non-key attribute) must depend on the entire primary key, not just a part of it.

Example: For a composite key (StudentID, CourseID), if the StudentName depends only on StudentID and not on the entire key, it violates 2NF. To normalize, move StudentName into a separate table where it depends only on StudentID.

3. Third Normal Form (3NF): Eliminating Transitive Dependency

A relation is in 3NF if it satisfies 2NF and additionally, there are no transitive dependencies. In simpler terms, non-prime attributes should not depend on other non-prime attributes.

Example: Consider a table with (StudentID, CourseID, Instructor). If Instructor depends on CourseID, and CourseID depends on StudentID, then Instructor indirectly depends on StudentID, which violates 3NF. To resolve this, place Instructor in a separate table linked by CourseID.

4. Boyce-Codd Normal Form (BCNF): The Strongest Form of 3NF

BCNF is a stricter version of 3NF where for every non-trivial functional dependency (X → Y), X must be a superkey (a unique identifier for a record in the table).

Example: If a table has a dependency (StudentID, CourseID) → Instructor, but neither StudentID nor CourseID is a superkey, then it violates BCNF. To bring it into BCNF, decompose the table so that each determinant is a candidate key.

5. Fourth Normal Form (4NF): Removing Multi-Valued Dependencies

A table is in 4NF if it is in BCNF and has no multi-valued dependencies. A multi-valued dependency occurs when one attribute determines another, and both attributes are independent of all other attributes in the table.

Example: Consider a table where (StudentID, Language, Hobby) are attributes. If a student can have multiple hobbies and languages, a multi-valued dependency exists. To resolve this, split the table into separate tables for Languages and Hobbies.

6. Fifth Normal Form (5NF): Eliminating Join Dependency

5NF is achieved when a table is in 4NF and all join dependencies are removed. This form ensures that every table is fully decomposed into smaller tables that are logically connected without losing information.

Example: If a table contains (StudentID, Course, Instructor) and there is a dependency where all combinations of these columns are needed for a specific relationship, you would split them into smaller tables to remove redundancy.

3) Explain BCNF

Rule 1: The table should be in the 3rd Normal Form.

Rule 2: X should be a super-key for every functional dependency (FD) X−>Y in a given relation.

Example:

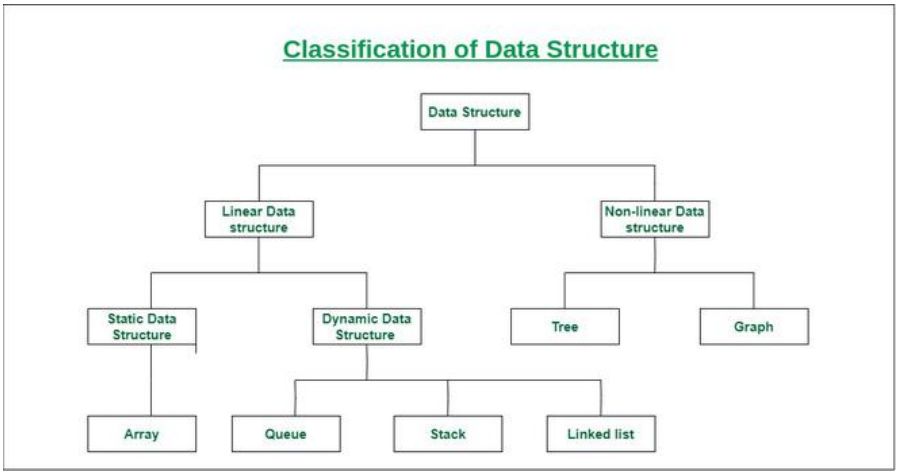
If a table has:

- Emp_ID → Dept_ID
- Dept_ID → Manager

Then if Dept_ID is not a super key, it violates BCNF. We decompose the table to remove this violation.

4) What is Data Structure?

A Data Structure is a way of organizing and storing data so that operations like insertion, deletion, searching, and updating can be performed efficiently.



5) What is OOPs?

OOPs is a programming paradigm based on the concept of objects, which can contain data (attributes) and methods (functions).

Key Concepts:

- Class: Blueprint for an object
- Object: Instance of a class
- Encapsulation: Binding data and code together
- Inheritance: Reuse features of existing classes
- Polymorphism: Same method behaves differently
- Abstraction: Hiding complex details and showing only essentials

6) Inheritance, Method Overloading

Inheritance:

It allows a class to acquire properties and behavior (methods) of another class.

Example (Python):

python Copy code

```python
class Animal:
    def sound(self):
        print("Animal sound")

class Dog(Animal):
    def sound(self):
        print("Bark")
```

Method Overloading:

Having multiple methods in a class with the **same name but different parameters.**

Note: Python does not support traditional method overloading but can be handled using default arguments or `*args`.

Example (Java-style logic):

python Copy code

```python
class Math:
    def add(self, a=None, b=None, c=None):
        if c is not None:
            return a + b + c
        elif b is not None:
            return a + b
        return a
```

7) Abstraction vs Interface

Abstraction is a general concept that involves hiding complex implementation details and exposing only essential information to the user. It simplifies complex systems by focusing on what an object does rather than how it does it.

An interface defines a contract that specifies methods a class must implement. It focuses on what a class should be able to do, without providing any implementation details. In Python, interfaces are typically implemented using abstract classes with only abstract methods, effectively functioning as an interface.

8) HTTP vs SMTP

Feature	HTTP	SMTP
Stands for	HyperText Transfer Protocol	Simple Mail Transfer Protocol
Used for	Web browsing (fetching web pages)	Sending emails
Port	80 (HTTP), 443 (HTTPS)	25, 587 (for secure)
Type	Pull protocol (client fetches)	Push protocol (server sends)

9) Operator Overloading

Operator Overloading allows us to define custom behavior for operators (`+`, `-`, `*`, etc.) for user-defined classes.

Example (Python):

python Copy code

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

p1 = Point(1, 2)
p2 = Point(3, 4)
p3 = p1 + p2
print(p3.x, p3.y)  # Output: 4 6
```

10) DBMS

A Database Management System (DBMS) is a software application that allows users to manage and organize data efficiently. The four types of DBMS are

- Relational DBMS (RDBMS)
- Hierarchical DBMS
- Network DBMS
- Object-Oriented DBMS (OODBMS)

11) RDBMS

An RDBMS is a database management system based on the relational model.

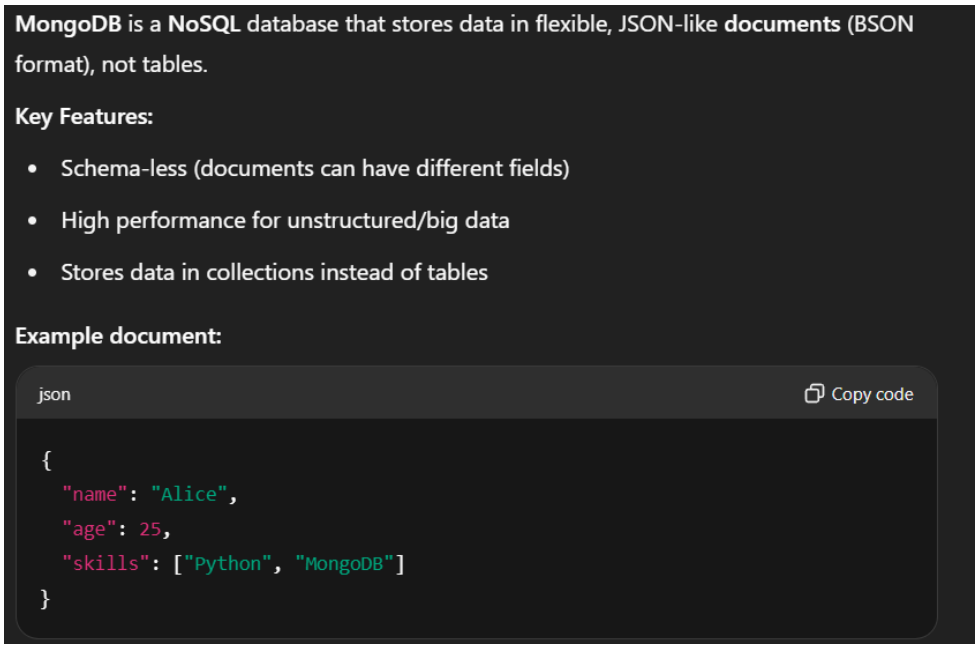
Data is stored in tables (rows and columns), and relationships between tables are maintained using keys (Primary key, Foreign key).

Examples: MySQL, PostgreSQL, Oracle, SQL Server

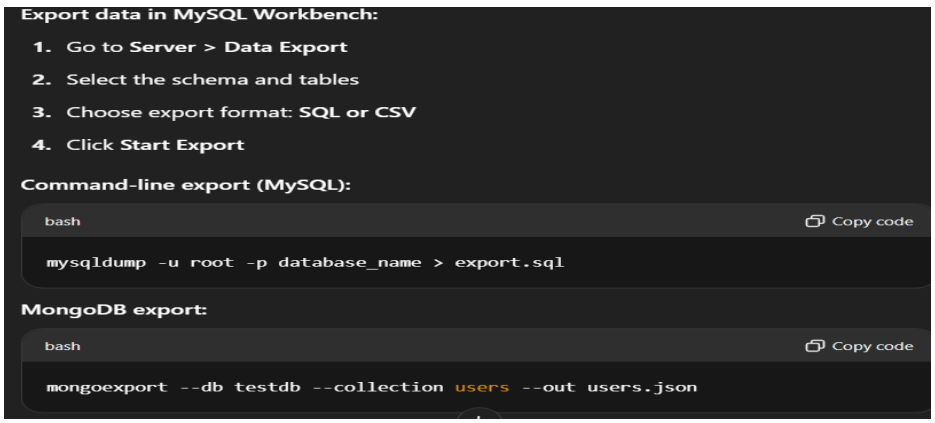
Key Features:

- Structured data with schemas
- ACID compliance (Atomicity, Consistency, Isolation, Durability)
- Supports SQL queries

12) MongoDB



13) What DBMS tool you know. How to export data from it?



14) Write SQL using SubQuery.

Example: Find employees who earn more than the average salary

sql Copy code

```sql
SELECT name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
);
```

SubQuery: `(SELECT AVG(salary) FROM employees)` returns the average salary, which is then used in the outer query.

15) Linux Commands

Command	Description
`ls`	List files and directories
`cd`	Change directory
`pwd`	Print current working directory
`mkdir dirname`	Create a new directory
`rm filename`	Delete a file
`rmdir dirname`	Remove an empty directory
`cp source dest`	Copy files or directories
`mv old new`	Rename or move files
`cat file.txt`	Display file contents
`grep "text" file`	Search for text in a file
`chmod 755 file`	Change file permissions
`top` or `htop`	View running processes
`df -h`	Show disk usage

16) Difference between Python 2 and Python 3 versions

Feature	Python 2	Python 3
Print Statement	`print "Hello"`	`print("Hello")`
Division	Integer division by default	True division by default
Unicode Support	Needs explicit declaration	Unicode is default
End of Life	Ended on Jan 1, 2020	Actively maintained
Compatibility	Older libraries	Modern libraries, better support

Conclusion: Use **Python 3** for all new projects.

17) Difference between Stack and Queue.

Feature	Stack	Queue
Structure	Last In, First Out (LIFO)	First In, First Out (FIFO)
Insertion	`push()`	`enqueue()`
Deletion	`pop()`	`dequeue()`
Example	Undo feature in editors	Print job management

Feature	Stack	Queue
Definition	A linear data structure that follows the **Last In First Out (LIFO)** principle.	A linear data structure that follows the **First In First Out (FIFO)** principle.
Primary Operations	Push (add an item), Pop (remove an item), Peek (view the top item)	Enqueue (add an item), Dequeue (remove an item), Front (view the first item), Rear (view the last item)
Insertion/Removal	Elements are added and removed from the same end (the top).	Elements are added at the rear and removed from the front.
Use Cases	Function call management (call stack), expression evaluation and syntax parsing, undo mechanisms in text editors.	Scheduling processes in operating systems, managing requests in a printer queue, breadth-first search in graphs.

18) Different types of JOINS in SQL.

Joins combine rows from two or more tables based on a related column.

🔗 **Types of Joins:**

Join Type	Description
INNER JOIN	Returns only matching rows from both tables
LEFT JOIN	All rows from the left table + matched right rows
RIGHT JOIN	All rows from the right table + matched left rows
FULL JOIN	All rows from both tables, matched or not
CROSS JOIN	Cartesian product (all combinations)
SELF JOIN	Table joined with itself

```sql
SELECT e.name, d.dept_name
FROM employees e
INNER JOIN departments d ON e.dept_id = d.id;
```

19) Explain recursion with example.

Recursion is when a function calls **itself** to solve smaller instances of a problem.

🔁 **Example: Factorial of a number** `n! = n × (n-1)!`

```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # Output: 120
```

- Base case: when `n == 0 or 1`, return 1
- Recursive case: keep calling `factorial(n - 1)`

20) What is purpose of indexes in database?

Index in a database is like an index in a book — it helps **find data faster**.

📌 **Benefits:**

- Speeds up **search** and **query performance**
- Reduces the time complexity from linear to logarithmic for lookups

Example:

```sql
CREATE INDEX idx_name ON employees(name);
```

🔍 Without index: full table scan
⚡ With index: direct access to matching rows

21) What is ddl,dml,dcl with examples?

Type	Full Form	Purpose	Example
DDL	Data Definition Language	Defines structure of database	`CREATE TABLE`, `ALTER`, `DROP`
DML	Data Manipulation Language	Manages data in tables	`INSERT`, `UPDATE`, `DELETE`
DCL	Data Control Language	Controls access to data	`GRANT`, `REVOKE`

```sql
-- DDL
CREATE TABLE students (id INT, name VARCHAR(50));

-- DML
INSERT INTO students VALUES (1, 'John');

-- DCL
GRANT SELECT ON students TO user1;
```

22) Difference between Delete and Truncate in SQL

Feature	`DELETE`	`TRUNCATE`
Removes	Specific rows (with condition)	All rows from table
WHERE clause	Yes	No
Rollback	Yes (if in transaction)	No (mostly irreversible)
Speed	Slower	Faster
Affects	Logs each row	Minimal logging

Examples:

sql Copy

```sql
DELETE FROM students WHERE id = 5;
TRUNCATE TABLE students;
```

23) What is lambda function?

A **lambda function** is a small **anonymous function** defined using the `lambda` keyword in Python.

🔧 **Syntax:**

python Copy code

```python
lambda arguments: expression
```

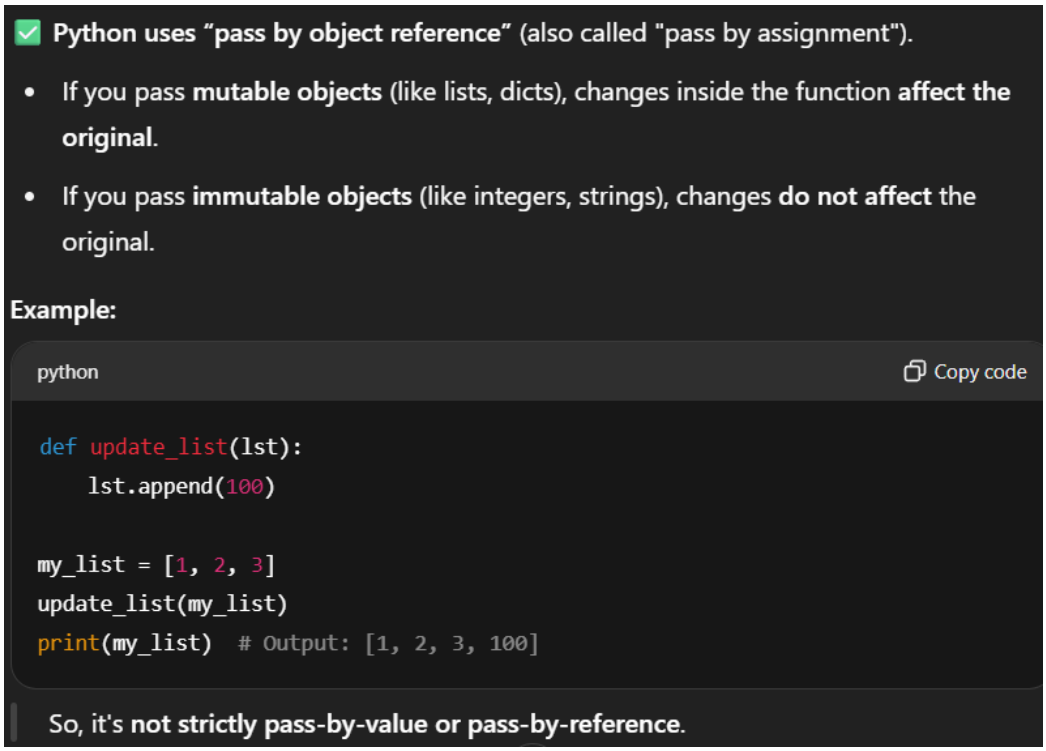
Example:

python Copy code

```python
square = lambda x: x * x
print(square(5))  # Output: 25
```

Used commonly with functions like `map()`, `filter()`, and `sorted()`.

24) Is python is pass by value or pass by reference?



25) What were the tables and columns in database?,

- **Tables**: A collection of related data organized in rows and columns (like Excel sheets).

- **Columns**: Fields or attributes (like "name", "age", "email") — each column stores a specific type of data.

- **Rows**: Each row is a record (or entry) in the table.

ID	Name	Age
1	Alice	20
2	Bob	22

26) DBMS architecture

DBMS Architecture refers to the design of how database systems are structured.

🔁 Types:

1. 1-Tier: User interacts directly with the database.
2. 2-Tier: Client ↔ Database (application is client).
3. 3-Tier (most common):
    - Presentation Layer (User Interface)
    - Application Layer (Business logic)
    - Database Layer (Data storage & access)

✅ 3-tier is preferred for security, modularity, and scalability.

27) why do you choose python language as your primary coding language.?

✅ **Reasons I chose Python:**

- **Easy syntax** – readable and beginner-friendly
- **Versatile** – used in web development, data science, automation, ML
- **Huge library support** – like NumPy, Pandas, OpenCV, Flask, etc.
- **Community and Resources** – lots of tutorials and support
- **Fast development time** – good for prototyping

> *"Python helps me turn ideas into working solutions quickly."*

28) Differences in programming paradigms (OOP vs procedural).

Feature	Procedural Programming	OOP (Object-Oriented Programming)
Approach	Step-by-step procedures	Objects and classes
Structure	Uses functions	Uses objects and methods
Data Access	Global or passed via functions	Encapsulated in objects
Example Language	C, Pascal	Java, Python, C++
Best For	Simple, linear tasks	Complex systems with real-world mapping

29) Are you open to learning new technologies?

"I believe in continuous learning. I enjoy exploring new technologies that can improve my skills, keep me industry-relevant, and contribute better to projects. I've already learned tools like OpenCV, Flask, and NS-3 outside my curriculum, and I'm always open to more."

30) What are some ways to improve page loading.

- optimize images by compressing them and choosing the right format
- minimize HTTP requests by combining files
- enable browser caching to store data locally.

31) Git vs Github

Feature	Git	GitHub
What is it?	A **version control system**	A **hosting service** for Git repositories
Use	Tracks changes in source code	Collaborates on Git projects online
Local/Cloud	Local tool	Cloud platform
Commands	`git add`, `commit`, `push`, etc.	Uses Git commands + web UI

> Think of Git as the **engine**, and GitHub as the **car showroom** where you share and collaborate.

32) Why not multiple inheritance in java

Java doesn't support multiple inheritance with classes to avoid the **Diamond Problem**, which causes ambiguity when two parent classes have the same method.

📌 **Example:**

If both `ClassA` and `ClassB` have a method `display()`, and `ClassC` inherits both, which `display()` should it use?

✅ **Java uses interfaces** to allow multiple inheritance **safely**:

java — Copy code

```java
interface A { void display(); }
interface B { void show(); }

class C implements A, B {
    public void display() { ... }
    public void show() { ... }
}
```

The "diamond problem" is an ambiguity that arises in object-oriented programming when a class inherits from multiple classes that share a common ancestor. This creates a diamond-shaped inheritance structure.

33) 5 features of java, python

✅ **Java Features:**

1. **Object-Oriented** – everything is based on classes and objects
2. **Platform Independent** – "Write Once, Run Anywhere" via JVM
3. **Robust** – Strong memory management, exception handling
4. **Multithreading** – Supports concurrent execution of two or more threads
5. **Secure** – No explicit pointer usage, built-in security features

✅ **Python Features:**

1. **Easy Syntax** – Simple, readable code (like plain English)
2. **Interpreted Language** – Runs line-by-line without compiling
3. **Dynamically Typed** – No need to declare variable types
4. **Extensive Libraries** – NumPy, Pandas, OpenCV, TensorFlow, etc.
5. **Multi-Paradigm** – Supports OOP, procedural, and functional programming

34) Exception handling

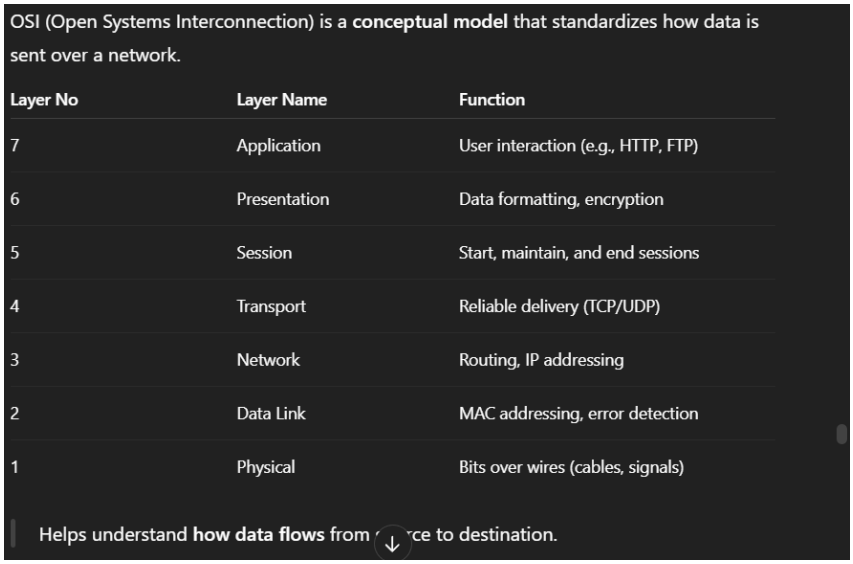
Exception handling is a process of dealing with **runtime errors** so that the program doesn't crash and can continue or exit gracefully.

🔁 **In Python:**

python — Copy cod

```python
try:
    a = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero.")
finally:
    print("This block always runs.")
```

35) OSI Model



OSI (Open Systems Interconnection) is a **conceptual model** that standardizes how data is sent over a network.

Layer No	Layer Name	Function
7	Application	User interaction (e.g., HTTP, FTP)
6	Presentation	Data formatting, encryption
5	Session	Start, maintain, and end sessions
4	Transport	Reliable delivery (TCP/UDP)
3	Network	Routing, IP addressing
2	Data Link	MAC addressing, error detection
1	Physical	Bits over wires (cables, signals)

Helps understand **how data flows** from ↓ rce to destination.

36) Pointer

A **pointer** is a variable that **stores the memory address** of another variable.

Example:

```c
int x = 10;
int *p = &x;

printf("%d", *p);  // Outputs 10
```

- `&x` → address of `x`
- `*p` → value at that address

37) Denormalization

Denormalization is the process of introducing redundancy into a normalized database to improve read performance.

- It combines multiple tables into one.
- Used when frequent joins slow down performance.
- Increases speed but may increase data redundancy and update anomalies.

✅ Use Case: In reporting systems where speed is more important than storage efficiency.

38) Can you build chatgpt using c?

C is a low-level language. It's not suited for developing or training modern AI models like **ChatGPT**, which require:

- Huge libraries (like PyTorch, TensorFlow)
- High-level tensor operations
- GPU acceleration
- Dynamic memory and automatic differentiation

✅ You *can* use C for building some underlying infrastructure, but not for developing or training a transformer model like ChatGPT.

39) C++ vs C: Use-cases where one is preferred over the other.

Feature	C	C++
Type	Procedural	Object-Oriented + Procedural
Use-Cases	OS, Embedded, Drivers	Game dev, GUI apps, large systems
Memory Mgmt	Manual (pointers)	Has classes, constructors, destructors
Code Structure	Flat and function-based	Modular, reusable via classes

> 🔹 **Use C**: When low-level hardware access or performance is key.
> 🔹 **Use C++**: When building apps with complex logic, reusability, or object-oriented features.

40) Difference between C and Python.

Feature	C	Python
Type	Compiled, low-level	Interpreted, high-level
Syntax	Complex, verbose	Simple, readable
Speed	Faster execution	Slower but fast development
Memory Mgmt	Manual (via pointers)	Automatic (garbage collection)
Use Cases	Embedded systems, OS	Web dev, AI, automation, scripting

> ✅ C for **performance**, Python for **productivity**.

41) Difference Between C++ and Java.

Feature	C++	Java
Compilation	Compiled to machine code	Compiled to bytecode (JVM)
Platform	Platform-dependent	Platform-independent
Memory Mgmt	Manual (with pointers)	Automatic (Garbage Collector)
Multiple Inheritance	Supported via classes	Not supported (only via interfaces)
Performance	Slightly faster	Slightly slower due to JVM overhead

> ✅ Use **C++** for system-level applications.
> ✅ Use **Java** for enterprise, cross-platform applications.

42) IPV4 VS IPV6

IPv4	IPv6
IPv4 has a 32-bit address length	IPv6 has a 128-bit address length
It Supports Manual and DHCP address configuration	It supports Auto and renumbering address configuration
In IPv4 end to end, connection integrity is Unachievable	In IPv6 end-to-end, connection integrity is Achievable
It can generate 4.29×10^{9} address space	The address space of IPv6 is quite large it can produce 3.4×10^{38} address space
The Security feature is dependent on the application	IPSEC is an inbuilt security feature in the IPv6 protocol
Address representation of IPv4 is in decimal	Address representation of IPv6 is in hexadecimal
IPv4 has a header of 20-60 bytes.	IPv6 has a header of 40 bytes fixed
IPv4 can be converted to IPv6	Not all IPv6 can be converted to IPv4
IPv4 consists of 4 fields which are separated by addresses dot (.)	IPv6 consists of 8 fields, which are separated by a colon (:)
IPv4's IP addresses are divided into five different classes. Class A , Class B, Class C, Class D , Class E.	IPv6 does not have any classes of the IP address.
IPv4 supports VLSM(Variable Length subnet mask).	IPv6 does not support VLSM.
Example of IPv4: 66.94.29.13	Example of IPv6: 2001:0000:3238:DFE1:0063:0000:0000:FEFB

OSI Ref. Layer No.	OSI Layer Equivalent	TCP/IP Layer	TCP/IP Protocol Examples
5,6,7	Application, Session, Presentation	Application	NFS, NIS+, DNS, `telnet`, `ftp`, `rlogin`, `rsh`, `rcp`, RIP, RDISC, SNMP, and others
4	Transport	Transport	TCP, UDP
3	Network	Internet	IP, ARP, ICMP
2	Data Link	Data Link	PPP, IEEE 802.2
1	Physical	Physical Network	Ethernet (IEEE 802.3) Token Ring, RS-232, others

43) TCP/IP Protocol

TCP/IP stands for **Transmission Control Protocol / Internet Protocol**. It's the **standard suite** used for internet communication.

💡 4 Layers:

1. **Application Layer** – HTTP, FTP, DNS
2. **Transport Layer** – TCP (reliable), UDP (faster)
3. **Internet Layer** – IP (routing, addressing)
4. **Network Access Layer** – Ethernet, Wi-Fi (physical & data link)

✅ TCP/IP ensures that data is broken into packets, transmitted, routed, and reassembled correctly.

44) What OS are?

An Operating System is system software that manages computer hardware and software resources, and provides services for application programs.

🖥️ Functions of OS:

- Manages **CPU, memory, storage, and I/O devices**
- Handles **file systems**
- Provides **user interface**
- Ensures **security and process management**

Examples: Windows, Linux, macOS, Android

45) Types of OS

- Batch Operating System
- Multi-Programming Operating System
- Multi-Processing Operating System
- Multi-User Operating Systems
- Distributed Operating System
- Network Operating System
- Real-Time Operating System
- Mobile Operating Systems

46) What is an array?

An array is a data structure that stores a fixed-size sequence of elements of the same type.

Example in Python:

```python
arr = [10, 20, 30, 40]
print(arr[2])  # Output: 30
```

Example in C:

```c
int arr[4] = {10, 20, 30, 40};
printf("%d", arr[2]);  // Output: 30
```

47) Different operators in c?

C provides the following **types of operators:**

1. **Arithmetic Operators:** `+`, `-`, `*`, `/`, `%`
2. **Relational Operators:** `==`, `!=`, `>`, `<`, `>=`, `<=`
3. **Logical Operators:** `&&`, `||`, `!`
4. **Assignment Operators:** `=`, `+=`, `-=`, `*=`, `/=`, `%=`
5. **Increment/Decrement:** `++`, `--`
6. **Bitwise Operators:** `&`, `|`, `^`, `~`, `<<`, `>>`
7. **Conditional (Ternary) Operator:** `? :`

48) Diff between SQL and MangoDb

Feature	SQL (e.g., MySQL, PostgreSQL)	MongoDB (NoSQL)
Data Format	Structured (Tables, Rows)	Unstructured (JSON-like Documents)
Schema	Fixed schema	Flexible, dynamic schema
Query Language	Structured Query Language (SQL)	MongoDB Query Language (MQL)
Joins	Supports joins	Limited joins, prefers embedding
Use Case	Relational data, ACID compliance	Big data, unstructured or flexible data

✅ Use SQL for **relational** data; MongoDB for **scalable, flexible** applications.

49) Diff btw Primary Key and Foreign Key

Key Type	Primary Key	Foreign Key
Uniqueness	Must be unique	Can have duplicates
Null Values	Cannot be NULL	Can be NULL
Main Purpose	Uniquely identifies a record	Links one table to another
Defined In	Parent table	Child table

> ✅ **Primary Key** = unique ID
> ✅ **Foreign Key** = reference to another table's primary key

50) Which language have oops concept?

- ✅ Full OOP: Java, C++, Python, C#, Ruby
- ✅ Partially OOP: JavaScript (Prototype-based), PHP, Swift, Kotlin

51) Join() in python

`join()` is a **string method** used to **combine elements of a list into a single string** with a specified separator.

Example:

python Copy code

```python
words = ['Hello', 'World']
sentence = ' '.join(words)
print(sentence)  # Output: Hello World
```

52) What is meant by DOM?

DOM stands for **Document Object Model**.

- It is a **tree-like structure** that represents HTML/XML documents.
- Allows programming languages like JavaScript to **access and manipulate web content** dynamically.

Example:

html Copy code

```html
<p id="demo">Hello</p>
<script>
  document.getElementById("demo").innerText = "Hi!";
</script>
```

> ✅ DOM lets you change text, images, styles, etc., **without reloading the page**.

53) Built-in Modules in Python

"Python has many built-in modules that provide ready-to-use functionalities. Some common ones are:

- `math` – for mathematical operations
- `datetime` – for working with dates and times
- `random` – for generating random numbers
- `os` – for interacting with the operating system
- `sys` – for accessing system-specific parameters and functions
- `json` – for working with JSON data
  These modules help avoid writing code from scratch."**

54) What is a Function and Its Uses?

A function is a block of reusable code that performs a specific task. It helps in breaking a large program into smaller, manageable parts.

55) **Constructor:**
A constructor is a special method that initializes objects when a class is created.
In Python, it's defined using __init__().

56) **Encapsulation:**

Encapsulation is wrapping data and methods into a single unit (class).

It protects data from direct access and helps with security and abstraction.

57) **Error Handling:**

Error handling allows a program to deal with unexpected events using try-except.

It prevents crashes and ensures smooth program flow.

58) **Why should we keep a SQL table in optimized manner?**

Optimized tables improve query speed, reduce storage, and enhance performance.

It helps handle large data efficiently and ensures faster access.

59) **Query to Add New Table (SQL):**

```
CREATE TABLE students (id INT, name VARCHAR(50), grade INT);
```

This creates a new table with three columns.

60) **Box Modeling in CSS:**

Box model consists of content, padding, border, and margin.

It defines how elements take up space and interact visually.

61) **Responsive Design:**

Responsive design ensures a website looks good on all screen sizes.

It uses flexible layouts, media queries, and relative units.

62) **Difference Between Class and ID in HTML:**

Class can be used on multiple elements; id is unique to one element.

IDs are referenced with #, classes with . in CSS.

63) **Access Modifiers:**

Access modifiers control visibility of class members (public, private, protected).

They help in encapsulating and protecting data.

64) **Strings:**

Strings are sequences of characters used to store text data.

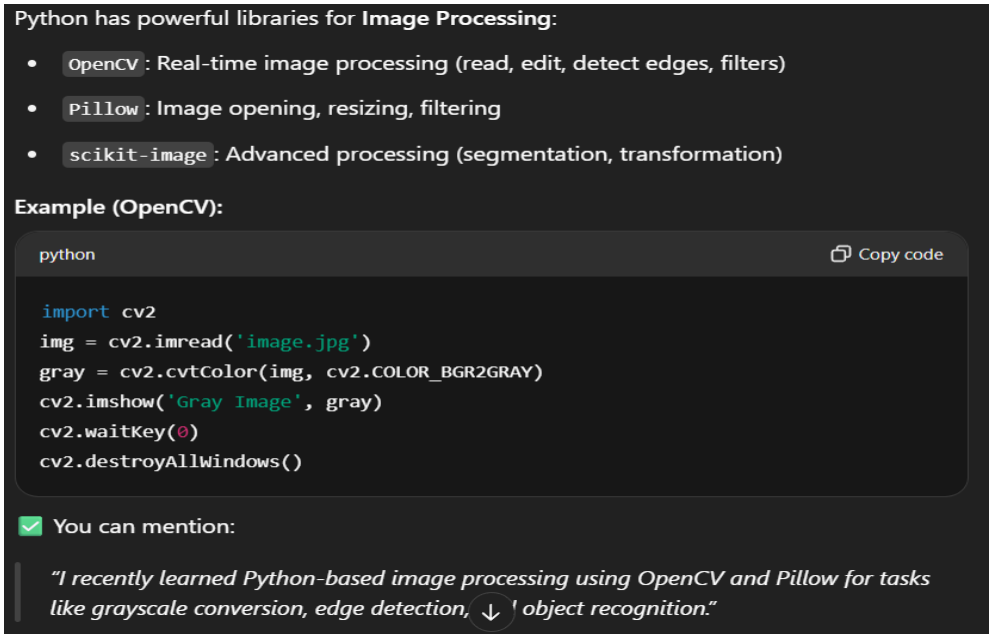
They support operations like slicing, joining, and formatting.

65) **Modifiers:**

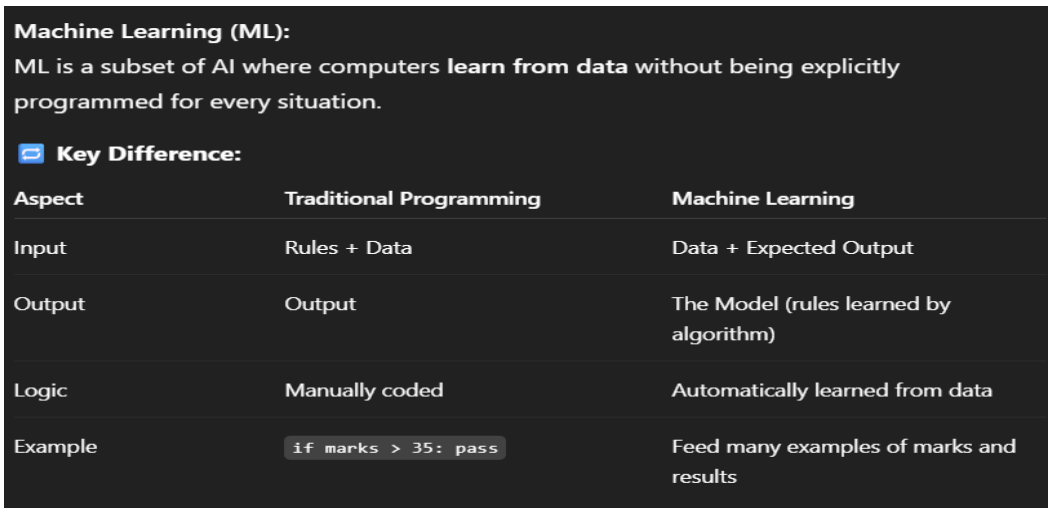
Modifiers are keywords that change the behavior of classes, methods, or variables.

Examples include public, private, static, and final.

66) New Technologies Recently Learned – Python Image Processing



67) What is machine learning and how it is different from normal Programming



68) Flask Working

Flask is a lightweight Python web framework used to build web applications and APIs.

🔧 **How Flask Works:**

1. Flask receives a web request (e.g., user visits a URL).
2. It checks the **route** (URL pattern) and calls the matching function.
3. The function returns a **response** (usually HTML or JSON).
4. Flask sends the response back to the browser.

Example:

python

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Flask!"

app.run()
```

69) linear regression

Linear Regression is a statistical method used in machine learning to model the **relationship between input (X)** and **output (Y)** using a straight line:

🔸 **Formula:** `Y = mX + b`

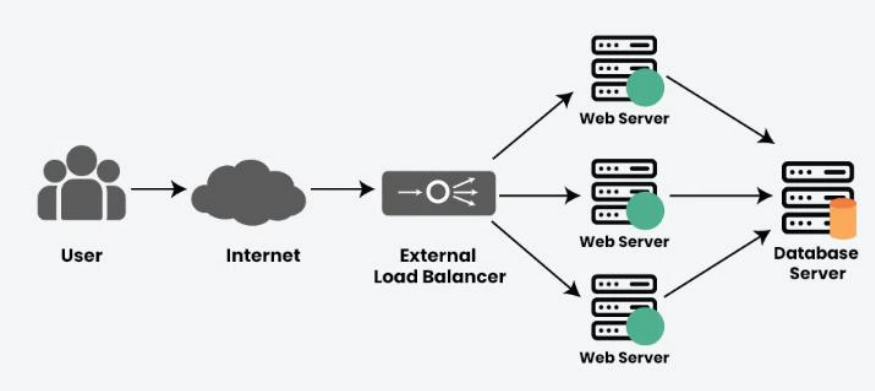
where:

- "`Y` is predicted value"
- "`X` is input"
- "`m` is slope (weight)"
- "`b` is intercept"

Use case:

Predicting **house prices, sales, marks**, etc., based on past data.

70) Load Balancer



A load balancer is a networking device or software application that distributes and balances the incoming traffic among the servers to provide high availability, efficient utilization of servers, and high performance. A

load balancer works as a “traffic cop” sitting in front of your server and routing client requests across all servers

- Load balancers are highly used in cloud computing domains, data centers, and large-scale web applications where traffic flow needs to be managed.
- It simply distributes the set of requested operations effectively across multiple servers and ensures that no single server bears too many requests.

71) Module vs package in python

1. What is the differnce between a module and package?

Feature	Module	Package
Definition	Single file with Python code	Directory with multiple modules and `__init__.py`
File Extension	`.py` file	Directory containing `__init__.py`
Usage	Group related code	Structured organization for large codebases
Example	`math_operations.py`	`mypackage/` with `__init__.py`, `math_operations.py`, `string_operations.py`
Import Example	`import math_operations`	`from mypackage import math_operations`
Initialization	Not required	Requires `__init__.py`
Organization	Flat structure	Hierarchical structure

72) Python interpreted?

2. Is Python a compiled language or an interpreted language?

Python is primarily an interpreted language. This means that Python code is executed line by line by an interpreter at runtime, rather than being compiled into machine code beforehand like in compiled languages.

73) Bytecode

Bytecode in Python is an intermediate code that is generated after the source code is compiled. It is a low-level, platform-independent representation of the source code that is executed by the Python Virtual Machine (PVM).

Here's a breakdown of the process:

- **Source Code:** This is the human-readable code that you write in Python.
- **Compilation:** When you run a Python program, the interpreter first compiles the source code into bytecode. This bytecode is a set of instructions that the PVM understands.
- **Execution:** The PVM then executes the bytecode, which is then translated into machine code that the computer's processor can understand.

74) Py

3. What are the benefits of using Python language as a tool in the present scenario?

- Simplicity
- Versatility
- Extensive Libraries and Frameworks
- Strong Community Support
- Portability
- Development Speed
- Dynamic Typing
- Open Source

75) Py

4. What are global, protected and private attributes in Python?

Global variables are public variables defined in the global scope. To use a global variable inside a function, the global keyword is required.

Protected attributes are marked with a single underscore (e.g., _sara). While they can still be accessed and modified from outside the class, responsible developers should avoid doing so.

Private attributes are marked with a double underscore (e.g., __ansh). These cannot be accessed or modified directly from outside the class, and any such attempt will result in an AttributeError.

76) Pandas

6. What is Pandas?

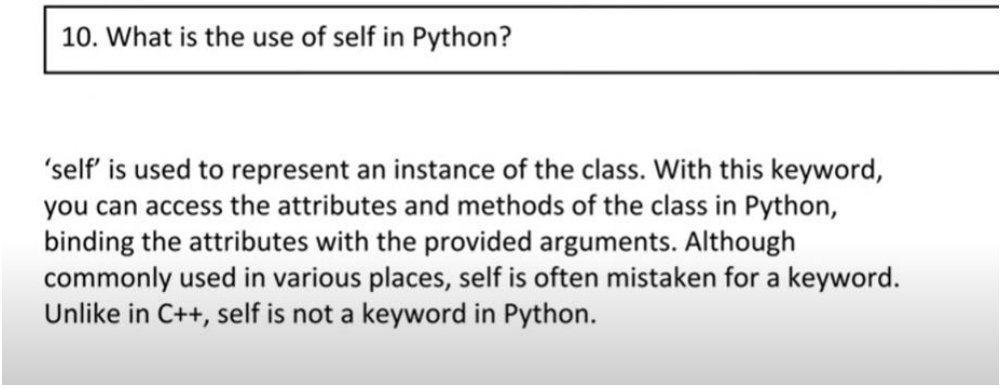
Pandas is an open-source Python library , which supports data structures for data-based operations associated with data analyzing and data manipulation. Pandas, with its rich sets of features, fits in every role of data operation, whether it be related to implementing different algorithms or solving complex business problems.

77) Exception Handling

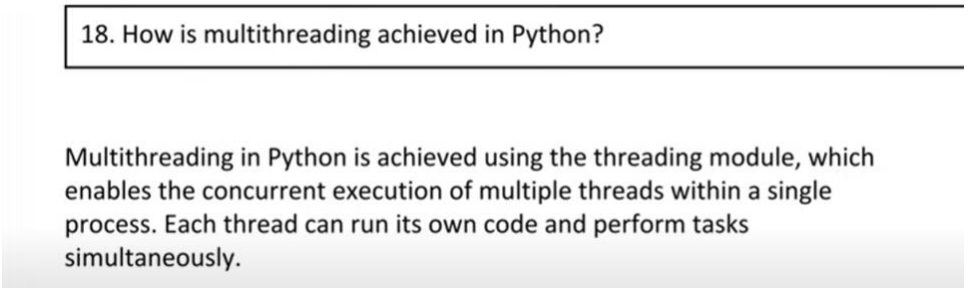
7. How is Exceptional handling done in Python?

In Python, the main keywords for handling exceptions are try, except, and finally. The try block contains the code that is monitored for errors. If an error occurs, the except block is executed to handle the exception. The finally block has the unique feature of executing code after the try block, regardless of whether an error occurred or not.

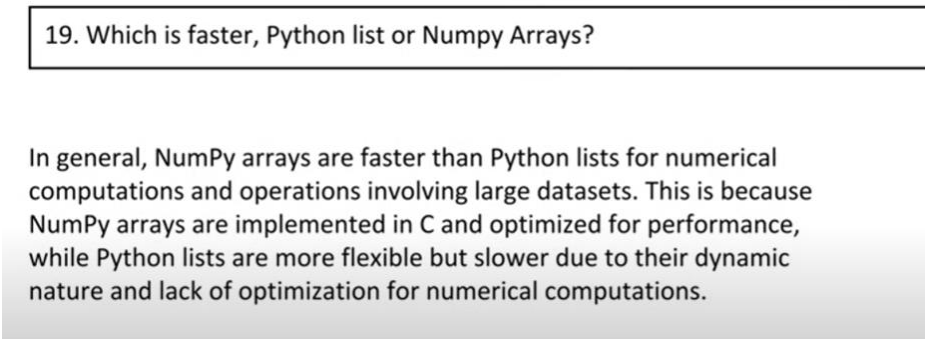
78) Self



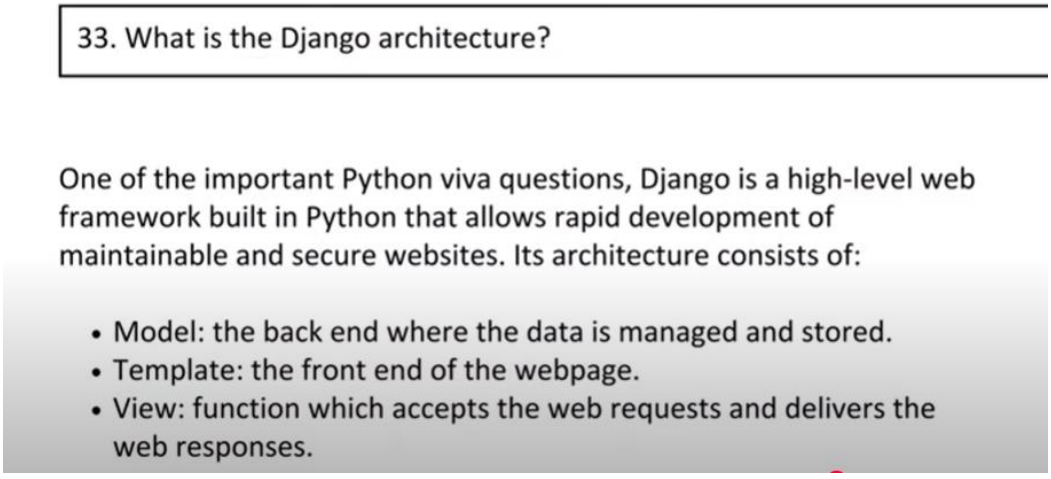
79) Multi



80) Numpy



81) Djo



82) Generators

Python generator functions are a powerful tool for creating iterators. In this article, we will discuss how the generator function works in Python.

Generator Function in Python

A generator function is a special type of function that returns an iterator object. Instead of using return to send back a single value, generator functions use yield to produce a series of results over time. This allows the function to generate values and pause its execution after each yield, maintaining its state between iterations.

Basic Code Example:

```
def fun(max):
    cnt = 1
    while cnt <= max:
        yield cnt
        cnt += 1

ctr = fun(5)
for n in ctr:
    print(n)
```

83) De

42. Is Django better than flask?

Django is better suited for large, complex applications with many built-in features, while Flask is better for small to medium projects requiring more flexibility and control; the choice depends on your specific needs and project requirements.

84) Flask

43. What is flask? Explain it's benefits

Flask is a lightweight and flexible web framework for Python that provides the essential tools to build web applications, making it easy to get started and scale as needed.

- Simplicity and Minimalism
- Flexibility
- Modular and Extensible
- Wide Adoption and Community Support

85) Libraries

Libraries in Python

A library is a collection of modules and packages that provide pre-written functionality for your program. Libraries are typically larger and more feature-rich than packages or modules.

Why use libraries?

To avoid writing common functionality from scratch.

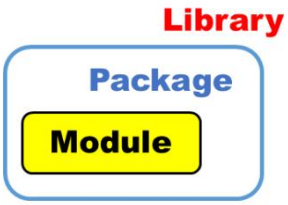
To leverage powerful tools developed by the community.

Example: Python has many popular libraries, such as:

- Pandas: For data manipulation.
- Matplotlib: For plotting and visualization.

 # Using a library (Pandas):

```
import pandas as pd
```



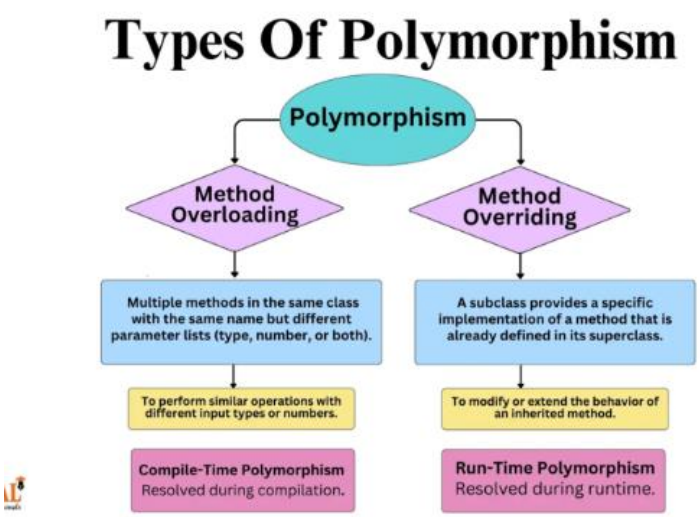
Concept	Key Feature	Example
Module	A single Python file with reusable code.	import math or custom file
Package	A directory of modules with an __init__.py.	from my_package import <*name*>
Library	A collection of modules/packages for functionality.	import pandas

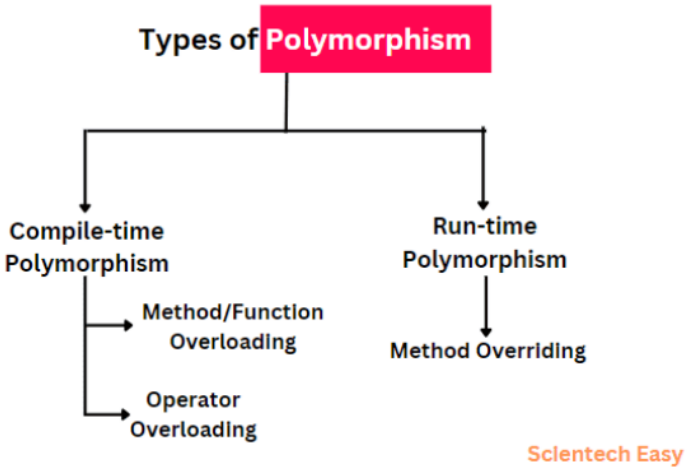
Data Analytics, data visualization and ML

Application	Library	Description	Install Command
Data Analytics	**Pandas**	Data manipulation and analysis.	pip install pandas
	NumPy	Numerical computing with array support.	pip install numpy
	SciPy	Scientific computing and technical computing.	pip install scipy
	Statsmodels	Statistical modeling and testing.	pip install statsmodels
	Dask	Parallel computing for large datasets.	pip install dask
Data Visualization	**Matplotlib**	Basic plotting and visualization.	pip install matplotlib
	Seaborn	Statistical data visualization.	pip install seaborn
	Plotly	Interactive graphs and dashboards.	pip install plotly
Machine Learning & Deep Learning	**Scikit-learn**	Classic machine learning algorithms.	pip install scikit-learn
	TensorFlow	Deep learning and ML models.	pip install tensorflow
	PyTorch	Deep learning with dynamic computation.	pip install torch torchvision
	Keras	High-level deep learning API.	pip install keras
	XGBoost	Gradient boosting for structured data.	pip install xgboost

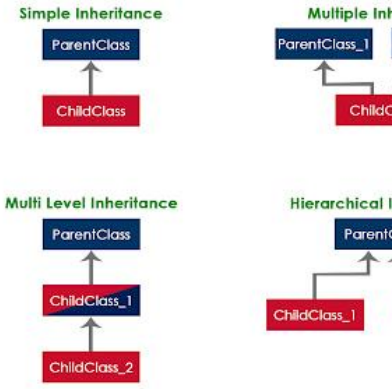
Web Scraping, web development and game development

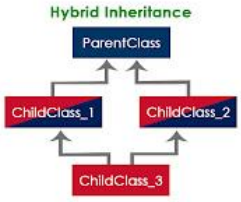
Application	Library	Description	Install Command
Web Scraping	**BeautifulSoup**	Parsing HTML and XML for data extraction.	pip install beautifulsoup4
	Scrapy	Advanced web scraping framework.	pip install scrapy
	Selenium	Browser automation for scraping dynamic sites.	pip install selenium
	Requests	HTTP library for fetching web pages.	pip install requests
	Lxml	Fast XML and HTML parsing.	pip install lxml
Web Development	**Django**	Full-stack web framework.	pip install django
	Flask	Lightweight web framework.	pip install flask
	FastAPI	High-performance API framework.	pip install fastapi
Game Development	**Pygame**	Game development library.	pip install pygame
	Arcade	Advanced 2D game development library.	pip install arcade
	Panda3D	Real-time 3D rendering and game creation.	pip install panda3d





Sclentech Easy





Access Modifiers in Python Encapsulation

Class member access specifier	Access from own class	Accessible from derived class	Accessible from object
Private member	Yes	No	No
Protected member	Yes	Yes	NO
Public member	Yes	Yes	Yes

C	C++	PYTHON
Longer lines of code as compared to python.	Longer lines of code as compared to python.	3-5 times shorter than C/C++ programs.
Longer lines of code as compared to python.	Longer lines of code as compared to python.	Python has no declaration.
C is a compiled language.	C++ is a compiled language.	Python is an interpreted language.
C supports procedural programming.	C++ supports both procedural and object oriented programming.	Python supports procedural, object-oriented, and functional programming.
C supports procedural programming.	C++ support both single and multiple inheritance	Python supports5 types of inheritance

86) Favourite subject in academics – Python, Networking

87) have done any project image processing: Yes

CODING

1) **Tell me your preferred language:** Python, C
2) **concatenation of string**

```python
s1 = "Hello"
s2 = "World"
result = s1 + s2
print("Concatenated string:", result)
```

C:

```c
#include <stdio.h>
#include <string.h>

int main() {
    char s1[20] = "Hello";
    char s2[] = "World";

    strcat(s1, s2);  // Concatenates s2 to s1

    printf("Concatenated string: %s\n", s1);
    return 0;
}
```

3) **Write a code of binary search and explain.**

Copy Edit

```python
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid  # Found
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1  # Not found

# Example
arr = [1, 3, 5, 7, 9]
print(binary_search(arr, 7))  # Output: 3
```

Explanation:

Binary search works by repeatedly dividing the search interval in half. If the target is less than the middle element, we search the left half; if greater, we search the right half.

4) **reverse a string in one line.**

```python
print("hello"[::-1])  # Output: "olleh"
```

5) **quick sort algorithm**

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    left = [x for x in arr[1:] if x <= pivot]
    right = [x for x in arr[1:] if x > pivot]
    return quick_sort(left) + [pivot] + quick_sort(right)

# Example
print(quick_sort([10, 5, 2, 3]))
```

6) **Bubble sort algorithm**

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Example
print(bubble_sort([64, 34, 25, 12]))
```

7) **Print a pyramid star pattern with proper spacing.**

```python
rows = 5
for i in range(1, rows + 1):
    # Print spaces
    print(" " * (rows - i), end="")
    # Print stars with space
    print("* " * i)
```

⭐ **Output:**

```markdown
    *
   * *
  * * *
 * * * *
* * * * *
```

8) **Swap 2 numbers without using 3rd variable.**

```python
a, b = 10, 20
a = a + b
b = a - b
a = a - b
print(a, b)
```

C:

```c
#include <stdio.h>
int main() {
    int a = 10, b = 20;
    a = a + b;
    b = a - b;
    a = a - b;
    printf("Swapped: %d %d\n", a, b);
    return 0;
}
```

9) **Add 2 numbers without using "+" Operator.**

```python
def add(a, b):
    while b != 0:
        carry = a & b
        a = a ^ b  # sum without carry
        b = carry << 1
    return a

print(add(5, 3))  # Output: 8
```

- **^ (XOR): adds bits without carrying.**
- **& (AND): finds carry bits.**
- **<< (Left shift): moves the carry to the next higher bit position.**

10) Prime or not

```c
int main() {

  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d", &n);

  // 0 and 1 are not prime numbers
  // change flag to 1 for non-prime number
  if (n == 0 || n == 1)
    flag = 1;

  for (i = 2; i <= n / 2; ++i) {

    // if n is divisible by i, then n is not prime
    // change flag to 1 for non-prime number
    if (n % i == 0) {
      flag = 1;
      break;
    }
  }

  // flag is 0 for prime numbers
  if (flag == 0)
    printf("%d is a prime number.", n);
  else
    printf("%d is not a prime number.", n);
```

```python
python

def is_prime(n):
    if n <= 1:
        return False  # 0 and 1 are not prime
    for i in range(2, n // 2 + 1):
        if n % i == 0:
            return False
    return True

# Example usage
num = int(input("Enter a number: "))
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

11) Palindrome

```python
python

def is_palindrome(s):
    return s == s[::-1]

# Example usage
text = input("Enter a string or number: ")
if is_palindrome(text):
    print(f"{text} is a palindrome.")
else:
    print(f"{text} is not a palindrome.")
```

```c
int main() {
    char str[100], reversed[100];
    int i, len, flag = 1;

    printf("Enter a string: ");
    scanf("%s", str);

    len = strlen(str);

    for (i = 0; i < len; i++) {
        reversed[i] = str[len - i - 1];
    }
    reversed[len] = '\0';

    if (strcmp(str, reversed) == 0)
        printf("%s is a palindrome.\n", str);
    else
        printf("%s is not a palindrome.\n", str);

    return 0;
}
```

```c
int main() {
    int n, original, reversed = 0, rem;

    printf("Enter a number: ");
    scanf("%d", &n);

    original = n;

    while (n != 0) {
        rem = n % 10;
        reversed = reversed * 10 + rem;
        n /= 10;
    }

    if (original == reversed)
        printf("%d is a palindrome.\n", original);
    else
        printf("%d is not a palindrome.\n", original);

    return 0;
}
```

12) Reverse the num

```python
def reverse_number(n):
    rev = 0
    while n > 0:
        rem = n % 10
        rev = rev * 10 + rem
        n //= 10
    return rev

num = int(input("Enter a number: "))
print(f"Reversed number: {reverse_number(num)}")
```

```c
#include <stdio.h>

int main() {
    int num, rev = 0, rem;

    printf("Enter a number: ");
    scanf("%d", &num);

    while (num != 0) {
        rem = num % 10;
        rev = rev * 10 + rem;
        num /= 10;
    }

    printf("Reversed number: %d\n", rev);
    return 0;
}
```

13) Fibonacci

```python
def fibonacci(n):
    a, b = 0, 1
    sequence = []
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

# Example usage
num_terms = int(input("Enter number of Fibonacci terms: "))
print(f"Fibonacci sequence: {fibonacci(num_terms)}")
```

```c
#include <stdio.h>
int main() {

  int n;

  int t1 = 0, t2 = 1;
  int nextTerm = t1 + t2;

  // get no. of terms from user
  printf("Enter the number of terms: ");
  scanf("%d", &n);

  // print the first two terms t1 and t2
  printf("Fibonacci Series: %d, %d, ", t1, t2);

  // print 3rd to nth terms
  while (nextTerm <= n) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2;
  }

  return 0;
}
```

14) Factorial

```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

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

if num < 0:
    print("Factorial does not exist for negative numbers.")
else:
    print(f"Factorial of {num} is {factorial(num)}")
```

15) Sum of Digits

```python
num = int(input("Enter a number: "))
sum_digits = 0

while num > 0:
    digit = num % 10
    sum_digits += digit
    num = num // 10

print("Sum of digits:", sum_digits)
```

📌 **Sample Output**

```yaml
Enter a number: 1234
Sum of digits: 10
```

✅ 1. Find Duplicates in a List

```python
lst = [1, 2, 3, 2, 4, 5, 1, 6]
duplicates = []

for i in lst:
    if lst.count(i) > 1 and i not in duplicates:
        duplicates.append(i)

print("Duplicates:", duplicates)
```

📌 **Output:**

```makefile
Duplicates: [1, 2]
```

✅ 2. Remove Duplicates from a List (Keep Order)

```python
lst = [1, 2, 3, 2, 4, 5, 1, 6]
unique = []

for i in lst:
    if i not in unique:
        unique.append(i)

print("List without duplicates:", unique)
```

📌 **Output:**

```less
List without duplicates: [1, 2, 3, 4, 5, 6]
```

16) **we need to find number of pairs (i, j) (i and j can overlap), such that number Arri, Arrj converted to string then added (string addition), then if the resulting number lies between x and y, then it is valid pair. find all such pairs (all, not only unique), E_{X} - n = 5 x = 20 y = 50 Arr = [2, 5, 7, 6, 3] String addition example "2" + "7" = "27" -> 27 converted to int**

```python
# Read input
N = int(input())

# Read array elements as strings
arr = input().split()

# Read the range values
x, y = map(int, input().split())

count_pairs = 0

# Check all pairs (i, j) including i == j
for i in range(N):
    for j in range(N):
        concatenated = arr[i] + arr[j]
        val = int(concatenated)
        if x <= val <= y:
            print(f"({arr[i]},{arr[j]}) -> {val}")
            count_pairs += 1

# Output the total count
print("Total Valid Pairs:", count_pairs)
```

Sample Input:

```
5
2 5 7 6 3
20 50
```

Sample Output:

```rust
(2,5) -> 25
(2,7) -> 27
(2,6) -> 26
(2,3) -> 23
(3,2) -> 32
(3,5) -> 35
(3,7) -> 37
(3,6) -> 36
(3,3) -> 33
Total Valid Pairs: 9
```

We are given:

- `n = 5`
- `x = 20`
- `y = 50`
- `Arr = [2, 5, 7, 6, 3]`

We need to:

- Count **all** pairs `(i, j)` such that:
  - Convert `Arr[i]` and `Arr[j]` to strings,
  - Concatenate them (string addition),
  - Convert the result back to an integer,
  - If the resulting number is between `x` and `y` inclusive, it is a **valid pair**.

We also need to:

- Print the valid pairs,
- Count them.

17) **Given a number of crates with dimensions, and the dimensions of an aisle, find how many crates can fit or what total volume or weight the aisle can hold.**