1. Smart contract and types of smart contract

A Smart Contract is a self-executing contract where the terms of the agreement (rules and conditions) are directly written into code and stored on a blockchain.

- It **automatically executes** when certain predefined conditions are met.
- No need for **middlemen** (like lawyers, banks).
- Smart contracts are **transparent**, **secure**, and **irreversible**.

Example - If Person A sends ₹1000 to Person B **only when** a product is delivered — this condition can be coded into a smart contract. Once the product is confirmed delivered, the contract automatically releases the money.

Types: -

1. Smart Legal Contracts

- These are **legal agreements** written in code.
- They work just like real-world legal contracts and can be used in court.
- Example: An insurance contract that automatically pays money if your car is damaged and proof is uploaded.

2. Decentralized Autonomous Organizations (DAOs)

- A DAO is an **organization run by smart contracts** instead of humans.
- Decisions are made by users voting (no single boss or owner).
- Example: A crowdfunding app where money is sent to the project only if enough people donate

3. Application Logic Contracts (ALCs)

- These contracts control how a decentralized app (DApp) works.
- They handle things like login, payments, or what features users can access.
- Example: In a music app, the smart contract gives access to songs only after the user pays.

2. Ethereum and it's comparison with Bitcoin

Ethereum: -

- **Ethereum** is a **blockchain platform** just like Bitcoin, but it does **more than just sending money**.
- It allows people to create and run **smart contracts** and **decentralized apps (DApps)**.
- It was created in **2015** by **Vitalik Buterin**.
- Its own currency is called **Ether (ETH)**, which is used to run programs on the Ethereum network.

Think of Ethereum like a computer system on the blockchain where you can build apps.

Bitcoin: -

- **Bitcoin** is the **first cryptocurrency**, created in **2009** by **Satoshi Nakamoto**.
- It is mainly used to **send and receive digital money**.
- It works like **digital gold** — people buy it and store it for value.

Think of Bitcoin as a calculator — it can only do one thing: send/receive money.

🔸 Feature	🪙 Bitcoin	🔷 Ethereum
Purpose	Digital money (used like gold)	Run smart contracts and DApps
Year Created	2009	2015
Founder	Satoshi Nakamoto	Vitalik Buterin
Currency Name	Bitcoin (BTC)	Ether (ETH)
Smart Contract Support	No smart contracts support	Yes, fully supports smart contracts
Speed	Slower (10 minutes per block)	Faster (12–15 seconds per block)
Supply Limit	Fixed supply (21 million BTC)	No fixed limit (more ETH can be created)
Main Use	Payment and store of value	Build apps, run contracts, make payments too

3. Various components of Ethereum.

1. Ethereum Virtual Machine (EVM)

- It's the **heart of Ethereum**, like the brain of the network.
- It **runs smart contracts** and makes sure everyone gets the same result.

Think of it like a computer that runs programs (smart contracts) on Ethereum.

2. Smart Contracts

- These are **self-executing codes** stored on the Ethereum blockchain.
- They run automatically when certain conditions are met.

No human needed – once written, they do their job without stopping.

3. Ether (ETH)

- Ether is the **cryptocurrency** of Ethereum.
- It's used to **pay for transactions** and run smart contracts (called "gas fee").

You need ETH to use Ethereum, like fuel for a car.

4. Accounts

- Two types:
    1. **Externally Owned Accounts (EOA):** Controlled by users (you and me).
    2. **Contract Accounts:** Controlled by smart contract code.

EOA is like your bank account, Contract account is like an automatic machine.

5. Gas

- Gas is the **fee** you pay to do any action on Ethereum (like send ETH or run a contract).
- It stops people from overloading the system.

More work = more gas fee.

6. Nodes

- Nodes are **computers** connected to the Ethereum network.
- They help **verify and store** every transaction and smart contract.

All nodes have the same copy of the Ethereum blockchain.

7. Solidity

- Solidity is the **programming language** used to write smart contracts on Ethereum.

Like C++ or Java, but made for Ethereum smart contracts.

4. Short note

a) Ethereum Virtual Machine (EVM)

- The **EVM** is the **core part** of Ethereum. It works like a **virtual computer** that runs smart contracts.
- Every Ethereum node runs the EVM to ensure that all smart contracts are executed the same way everywhere.

Key Points:

- It runs the code written in smart contracts.
- It makes Ethereum a **decentralized world computer**.
- Code is converted to **bytecode**, which EVM understands and executes.

Think of EVM as the engine of Ethereum that processes all the logic behind smart contracts.

b) Ethereum Programming Language

- Ethereum uses specific languages to **write smart contracts**.
- The main one is **Solidity**, but there are a few others too.

Common Languages:

Language	Use
Solidity	Most popular; looks like JavaScript or C++
Vyper	Simple and secure; Python-like syntax
LLL	Low-level; rarely used by beginners

Solidity is the most widely used language to write smart contracts on Ethereum.

c) Runtime Bytecode

- When you write a smart contract in Solidity, it gets **compiled** (converted) into something called **bytecode**.
- This bytecode is what the **EVM understands and executes**.

Key Points:

- Bytecode is stored on the blockchain.
- EVM runs this code to perform contract actions (like sending money, updating data, etc).

Runtime bytecode = final version of your contract that actually runs on Ethereum.

5. Solidity in detail

- **Solidity** is a **programming language** designed specifically for writing **smart contracts** on Ethereum.

- It was developed by the Ethereum team and is similar in style to **JavaScript, C++, and Java**, making it easier for developers familiar with these languages to learn.

How Solidity Works:

- You write code in Solidity.
- The code is compiled into bytecode.
- Bytecode is uploaded and stored on Ethereum blockchain.
- Ethereum Virtual Machine (EVM) runs this bytecode to execute your contract.

Key Features of Solidity:

1. **Statically Typed:**
   You must declare variable types (like int, string) before using them.
2. **Contract-Oriented:**
   Instead of traditional programs, you write **contracts** which contain **functions and data**.
3. **Supports Inheritance:**
   Contracts can inherit properties and functions from other contracts, helping code reuse.
4. **Events:**
   Solidity allows smart contracts to **emit events** that external apps (like websites) can listen to.
5. **Gas Efficient:**
   Since every operation costs gas, Solidity encourages writing efficient code.

Solidity Importance: -

- It is the **main language for Ethereum smart contracts**.
- Helps create **trustless, decentralized applications**.
- Enables automation of agreements without intermediaries.

6. **Bytecode in Ethereum and its importance and list the different language supported by Ethereum.**

What is Bytecode?

- When you write a smart contract in a high-level language like **Solidity**, it cannot run directly on the Ethereum network.
- So, the code is **compiled** into **bytecode**, which is a low-level, machine-readable code.
- This **bytecode** is what the **Ethereum Virtual Machine (EVM)** actually understands and executes.

Importance of Bytecode:

1. **Executable Code:**
   Bytecode is the actual code that runs on every Ethereum node's EVM. Without bytecode, the contract cannot work.
2. **Blockchain Storage:**
   The bytecode is **stored on the Ethereum blockchain**, making the smart contract immutable (unchangeable).
3. **Decentralized Execution:**
   All nodes execute the same bytecode to ensure trustless and consistent contract execution.

- Languages supported by Ethereum

Language	Description
Solidity	Most popular; similar to JavaScript and C++
Vyper	Python-like syntax; focused on security and simplicity
LLL	Low-level Lisp-like language; more complex but powerful
Bamboo	Experimental language focusing on security and simplicity
Yul	Intermediate language for optimizing bytecode; used for low-level programming

7. Working of smart contract

A **Smart Contract** is a self-executing program stored on the blockchain. It automatically runs and performs actions when certain conditions are met — **without needing any middleman**.

Step-by-Step Working:

1. Write the Contract

- The developer writes the smart contract using a language like **Solidity**.
- It includes **rules and conditions** (like "if A happens, then do B").

2. Compile to Bytecode

- The contract is **compiled** into **bytecode**, which is readable by the **Ethereum Virtual Machine (EVM)**.

3. Deploy to Blockchain

- The compiled bytecode is **uploaded to the Ethereum blockchain** using a transaction.
- After deployment, it gets a **unique address**.

4. Users Interact with It

- Anyone can **interact** with the smart contract by sending transactions (e.g., payment, voting, etc.).
- When the conditions in the contract are fulfilled, it **automatically executes the defined action**.

5. EVM Executes the Code

- The Ethereum Virtual Machine (EVM) runs the contract code **on all nodes**, ensuring the result is the same everywhere.
- The **result is stored** on the blockchain, permanently.

Simple Example:

Let's say you create a smart contract for **automatic rent payment**.

Rules in the contract:

- If today is the 1st of the month → transfer ₹10,000 from tenant to owner.

What happens:

- On the 1st, the contract checks the date.
- If the condition is true, it **automatically transfers** the money — no one needs to approve it.

Additional points: -

Feature	**Benefit**
Automation	Works automatically without human action
Transparency	Anyone can see the contract code
Security	Once deployed, it can't be changed easily
Trustless	No need to trust anyone — it runs as written