Blockchain:

1. Super secure digital ledger.
2. Records Transactions.
3. Data cannot be manipulated.
4. Does Not need a middleman.
5. Each record of a transaction is called a block.
6. Blocks are permanent and data in them cannot be changed.
7. Every transaction changes the network's overall state and these are grouped into blocks.

Purpose of Blockchain:

1. Blockchain is a type of distributed ledger technology (DLT), so it allows recording transactions and related data in multiple places at the same time.
2. Block technology provides safety of the information contained in the blocks as it can neither be deleted nor updated.Only a new block containing information about newer transactions can be added to the chain.In this way data is secured and immutable.

Limitations of Databases:

1. Entire data can be dropped mistakenly.
2. Cannot share sensitive information with others.
3. No authenticity of details stored.
4. Users don't have power over their own data.
5. People can compromise the database.

Bitcoins and Blockchain:

1. Secure network that cannot be hacked.
2. Blockchain starts with a genesis state . when they begin, like the birth of the network.

Types of Blockchain:

1. **Private:** Use only internally, and when we know the users (eg. Hyperledger Fabric).
2. **Public:** Everyone can see what is going on (Bitcoin, Ethereum).
3. **Hybrid:** In case you want to combine the first two

<u>Blocks:</u>

1. Has 3 elements.
2. The data is in a block.
3. The 32 bit whole number is called a nonce, it is randomly generated when a block is created, which then generates a block header hash.
4. Hash is 256 bit number wedded to nonce.
5. Nonce generates cryptographic hash.
6. All transactions shared across the nodes are contained in blocks, which are contained in blocks, which are chained together to create ledger.
7. All the data in the ledger is secured by cryptographic hashing and digital signature and validated by a census algorithm.

<u>Nodes:</u>

Nodes can be any kind of electronic device that maintains copies of the blockchain and keeps the network functioning.

Every node has its own copy of the blockchain and the network must algorithmically approve any newly mined block for the chain to be updated, trusted and verified. Since blockchains are transparent, every action in the ledger can be easily checked and viewed. Each participant is given a unique alphanumeric identification number that shows their transactions.

<u>Centralized System:</u>

This system has a single entity as its authority and the entire data is held by that entity.
Ex:-Banks

<u>Decentralized System:</u>

In this system the authorization is done through an openly known protocol.In this system data is stored by the participants/users i.e every node/user on that network holds a copy of the ledger.
Ex:-Peer to Peer network.

Distributed:

Blockchain works on the principle of Trustless Consensus which means a particular entity is not entrusted with the responsibility to maintain the ledgers instead this power to update the nodes is distributed among all the nodes in the network.It involves processes like Proof of Work(PoW) and Proof of Stake(PoS).

Advantages of Decentralized over Centralized System:

1. Fault tolerance:-Data is secured as there is no single source from which the system can be compromised as data is distributed to every node over the entire network.
2. Sovereignty:-The users in the network can exactly know when and where their data is used.
3. No middlemen:-As blockchain involves direct transactions from one person to another without any involvement of third party or banks.The taxes on transactions can be saved.

Key Concepts:

1. Cryptographic hash and digital signature.
2. Immutable ledger.
3. P2P Network.
4. Consensus Algorithm
5. Block Validation

Creating Blockchain from scratch:

1. Create a block.
2. Add the data(headers and body).
3. Hash the block
4. Chain the blocks together.

Digital Signatures:

1. Only you can sign.
2. Anyone can verify.
3. Tied to a message.

Cryptography and Digital Signature:

Digital signatures is a technique that binds a person/entity to the digital data. It is calculated from the data and secure key known only by the signer.

Cryptography is used for securing information by converting it into a code that can only be deciphered by someone who has the appropriate key or password.

Digital signatures use cryptography to create unique digital fingerprints of a document that can be used to verify its authenticity and ensure its integrity.

Hash Functions:

Hashing is a one-way function and with a properly designed algorithm, there is no way to reverse the hashing process to reveal the original input.

There are five requirements for a cryptographic hash function:

1. The input can be of any length.
2. The output has a fixed length.
3. The hash function is relatively easy to compute for any input.
4. The hash function is one-way (this means it is extremely hard if not impossible to determine the input from the output).
5. The hash function is collision-free (there can't be two different messages producing the same hash value).

Smart Contacts:

A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. The line **uint storedData;** declares a state variable called **storedData** of type **uint** (unsigned integer of 256 bits).

A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. The line **uint storedData;** declares a state variable called **storedData** of type **uint** (unsigned integer of 256 bits). You can think of it as a single slot in a database that you can query and alter by calling functions of the code that manages the database. In this example, the contract defines the functions **set** and **get** that can be used to modify or retrieve the value of the variable.

Practice Codes:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract twitter{
    mapping(address => string) public tweets;

    function create(string memory _tweet) public{
        tweets[msg.sender]=_tweet;
    }

    function get(address _owner) public view returns (string memory){
        return tweets[_owner];
    }

}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract twitter{
    mapping(address => string[]) public tweets;

    function create(string memory _tweet) public{
        tweets[msg.sender].push(_tweet);
    }

    function get(address _owner,uint i) public view returns (string memory){
        return tweets[_owner][i];
    }

}
```

Storage Examples:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

Subcurrency Examples:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping(address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent(address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    // Errors allow you to provide information about
    // why an operation failed. They are returned
    // to the caller of the function.
    error InsufficientBalance(uint requested, uint available);

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        if (amount > balances[msg.sender])
            revert InsufficientBalance({
                requested: amount,
                available: balances[msg.sender]
            });

        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Sources:

1. https://x.com/muskanjain0401/status/1710210195832594633?s=48
2. https://builtin.com/blockchain
3. https://github.com/Vatshayan/Money-Transcation-Security-Blockchain-Project
4. https://inversegravity.net/2019/crypto-hash-digital-signature/
5. https://docs.soliditylang.org/en/v0.8.21/introduction-to-smart-contracts.html