Class Notes (Do not run code)

```
// Here’s the basic syntax of a function:


    function functionName(parameters) public/ private/ internal/ external returns (type) {
    // Function body
}

// Example: A Simple Function

    function getMessage() public view returns (string memory) {
    return message;
}

// Types of Functions

    function getBalance() public view returns (uint) {
    return address(this).balance;
}


// Pure Functions

function addNumbers(uint a, uint b) public pure returns (uint) {
    return a + b;
}


// Write Functions

function updateMessage(string memory _newMessage) public {
    message = _newMessage;
}


// Payable Functions

function deposit() public payable {
    // msg.value contains the amount of Ether sent
```

}

// Visibility in Functions

public: Accessible by anyone (internal and external).
private: Only accessible within the contract.
internal: Accessible only within the contract and derived contracts.
external: Only accessible externally, not from within the same contract.

// Modifiers in Functions
// You can add modifiers to functions for specific behaviors.

view: Reads blockchain data but does not modify it.
pure: Does not read or modify blockchain data.
payable: Allows the function to accept Ether.

// Key Points to Remember
View and Pure Functions do not cost gas unless they are called by another function.
Write Functions cost gas because they modify the blockchain state.
Payable Functions allow your contract to receive Ether.
Always use visibility modifiers and require statements for security.

Code Examples (Do not run code) Practice writing code in Re-mix

```
pragma solidity ^0.8.25;

contract SimpleStorage {
    uint public storedData;

    // Write function: Update data
    function set(uint x) public {
        storedData = x;
    }

    // View function: Read data
    function get() public view returns (uint) {
        return storedData;
    }
}
```

Practice writing code in Re-mix

```
pragma solidity ^0.8.25;

contract PayableExample {
    address public owner;

    constructor() {
        owner = msg.sender;
    }

    // Payable function: Accept Ether
    function deposit() public payable {}

    // View function: Check contract balance
    function getBalance() public view returns (uint) {
        return address(this).balance;
    }
```

```
    // Transfer Ether
    function withdraw(uint amount) public {
        require(msg.sender == owner, "Only owner can withdraw");
        payable(owner).transfer(amount);
    }
}
```