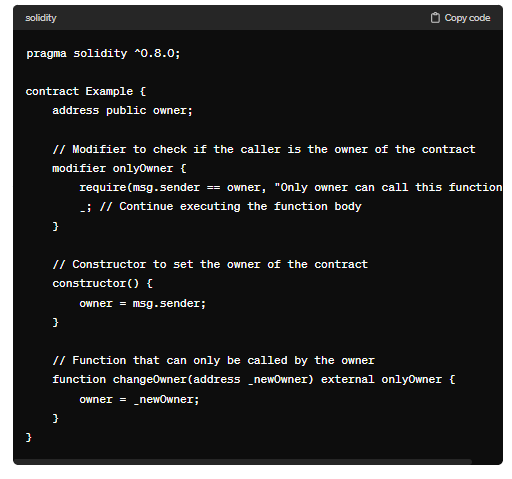
**Function Modifiers:**

In Solidity, a function modifier is a way to add custom logic to functions in a contract. Modifiers are typically used to perform common checks or actions that need to be executed before or after a function is called. They help in reducing code duplication and improving readability.



Modifiers can be combined and reused across multiple functions within a contract, making Solidity code more modular and easier to maintain.

This is a special Type Function. We can use modifier to reuse the code.

**Code:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract Modifier{

    function function1() public pure TestCode returns(string memory) {

        return "fun1 say hi";

    }

    function function2() public pure TestCode returns(uint \_x){

        \_x=20;

    }

    function function3() public view TestCode returns(address){

        return msg.sender;

    }

    modifier TestCode() {

        for(uint \_i=0;\_i<10;\_i++) {

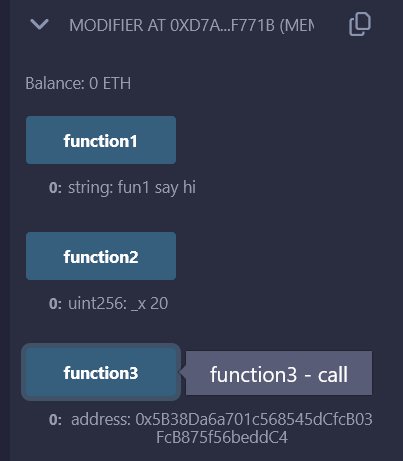
            }

        \_;

    }

}

**Output:**



//SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract auction{

    address public owner = msg.sender;

    modifier onlyOwner(){

        require(owner ==msg.sender,"you are not the owner");

        \_;

    }

    function startAuction() public view onlyOwner{

    }

    function stopAuction() public view onlyOwner{

    }

    function checkStatus() public view onlyOwner{

    }

    uint public age=30;

    modifier temp(uint \_x) {

        age = age+\_x;

        \_;

    }

    function changeAge(uint \_y) public temp(\_y){

    }

}

**Output:**

