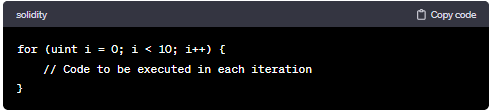
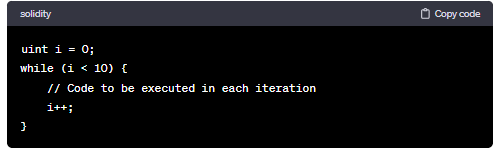
**Loops - For loop, While loop, and Do-While loop:**

In Solidity, loops are used to repeat a set of instructions multiple times. There are two main types of loops in Solidity: **for** loops and **while** loops.

**For Loops:**. It consists of an initialization statement, a condition, and an iteration statement.

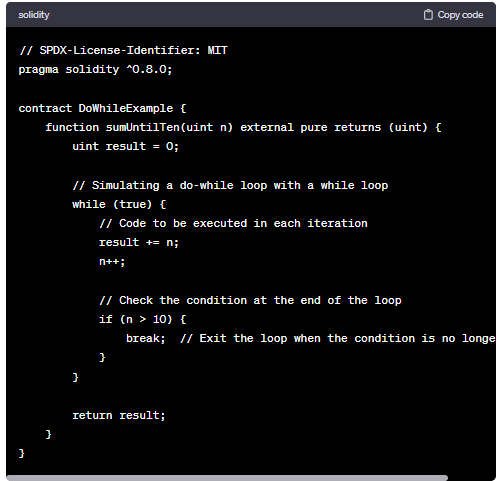


**While Loops:** The **while** loop continues executing as long as the specified condition is true.



It's important to be cautious when using loops in smart contracts to avoid gas-related issues. Infinite loops or loops with an undetermined number of iterations should be avoided to prevent potential attacks on the contract due to excessive gas consumption.

**Do-While Loops:** Solidity does not have a built-in **do-while** loop like some other programming languages. However, you can simulate the behavior of a **do-while** loop using a **while** loop with a pre-check.



Keep in mind that while this approach achieves the behavior of a **do-while** loop, you should be careful with the loop conditions to prevent infinite loops and excessive gas consumption in your smart contracts.

**Code:**

**For-Loop:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract loops{

    function loop() public pure returns(uint) {

        uint count=0;

        for(uint i=0; i<10; i=i+2){

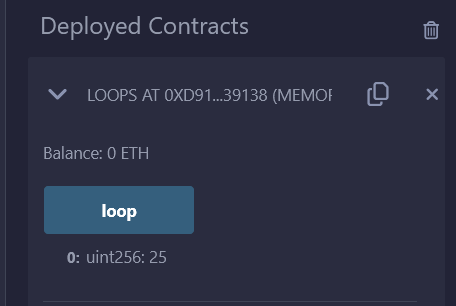
            count = count+5;

        }

        return count;

    }

}



**While-Loop:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract loops{

    function loop() public pure returns(uint) {

        uint count=0;

        uint j=0;

        while(j==1){

            count = count+5;

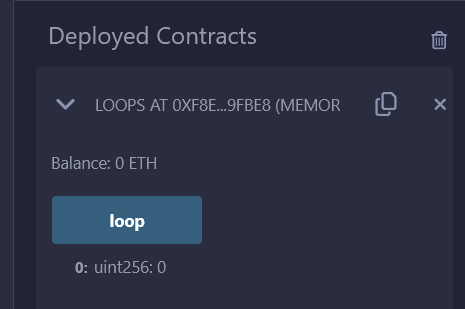
            j = j+2;

        }

        return count;

    }

}



**Do-While-Loop:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract loops{

    function loop() public pure returns(uint) {

        uint count=0;

        uint j=0;

        do{

            count = count+5;

            j = j+2;

        } while(j<10);

        return count;

    }

}

