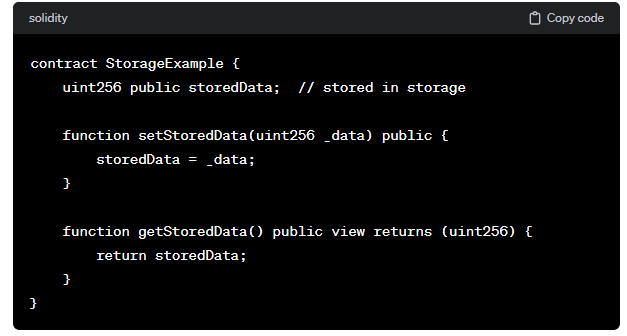
**(Storage Vs Memory in Solidity):**

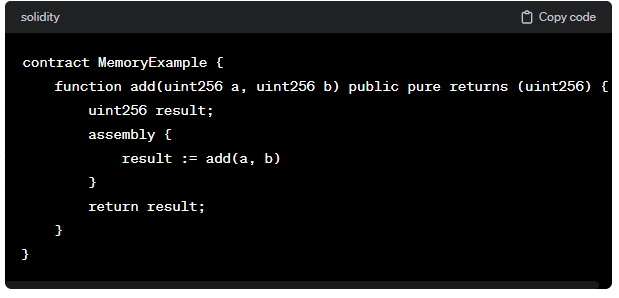
In Solidity, as well as in many programming languages, the terms "storage" and "memory" refer to different types of data locations where variables can be stored. Understanding the distinction between storage and memory is important for writing efficient and secure smart contracts.

1. **Storage:**
   * **Persistent Storage:** Variables declared with the **storage** keyword are stored permanently on the blockchain. This is used for state variables in smart contracts, and the data persists between function calls and transactions.
   * **Costly Operations:** Reading from and writing to storage are relatively expensive operations in terms of gas costs. Writing to storage consumes more gas than reading from storage.



**Memory:**

* **Temporary Storage:** Variables declared with the **memory** keyword are temporary and exist only during the execution of a function. They are used for holding data that is needed during the execution of a function and is not required to persist on the blockchain.
* **Low Gas Costs:** Operations on memory are cheaper in terms of gas compared to storage operations.



It's crucial to choose the right type of data location to optimize gas usage and prevent unnecessary persistence of data. In general, use **storage** for state variables that need to persist across multiple transactions, and use **memory** for temporary data within a function's scope.

Additionally, there is another keyword called **calldata**, which is used to store function arguments and is read-only. Understanding the use of each storage type is essential for writing secure and gas-efficient smart contracts in Solidity.

|  |  |
| --- | --- |
| **Storage** | **Memory** |
| Holds state variables. | Holds local variables defined inside functions if they are reference types |
| Persistent | Not persistent |
| Cost gas | No gas |
| Like a computer HDD | Like a computer RAM |

**Code:**

//SPDX-License-Identifier: GPL-3.0

pragma solidity 0.8.0;

contract demo{

    string[] public student=['Mohan','Rohan','Ankush'];

    function mem() public view

    {

        string[] memory s1=student;

        s1[0]='Akash';

    }

    function sto() public

    {

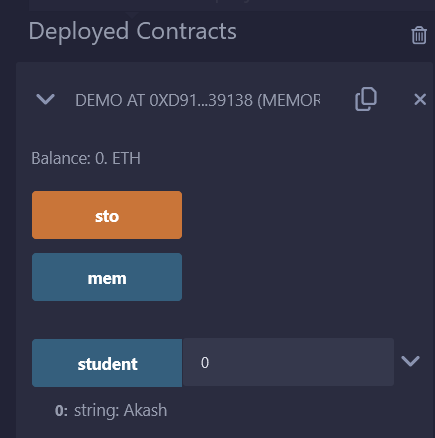
        string[] storage s1=student;

        s1[0]='Akash';

    }

}

**Output:**



-> If we click on Student first we can see 0 -> Rohan.

-> Now if we click on men and again click on student we see the same output.

-> Now if we click on sto and then click on student we can see the output as per screenshot.