



BLOCKCHAIN TECHNOLOGY LAB

(20CP406P)

LAB ASSIGNMENT - 6



B.Tech in Computer Science and Engineering Dept.,
Pandit Deendayal Energy University,
Gandhinagar



Name: Mire Kishorkumar Patel

Roll No.: 19BCP080

Branch: Computer Engineering

Lab Assignment 6

Aim: Learn Syntactical details of Solidity through simple Smart Contracts

Introduction:

Solidity contracts are similar to the classes in other object-oriented programming languages. Data that can change these variables are securely contained in them as state variables. The EVM function call takes place and the context is switched when a function is performed on a separate instance (contract), rendering the state variables inaccessible. For anything to occur, a contract or its function must be invoked. The following are some fundamental attributes of contracts:

- **Constructor:** A special method developed using the constructor keyword, that is only used once, during the creation of the contract.
- **State Variables:** State Variables are the variables used to store the contract's current state.
- **Functions:** The status of the contracts can be manipulate by using functions to change the state variables.

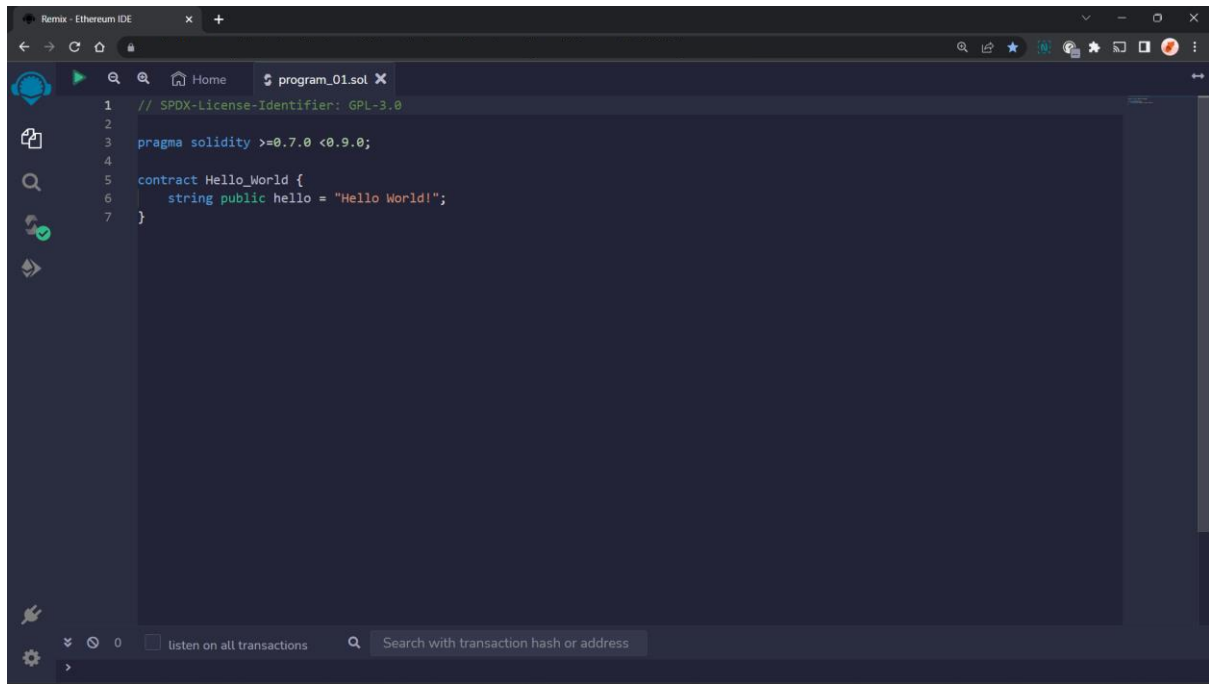
Syntax:

```
contract <contract_name>{  
  
    constructor() <visibility>{  
        .....  
    }  
    // rest code  
}
```

Source Code:

[1]

program_01.sol

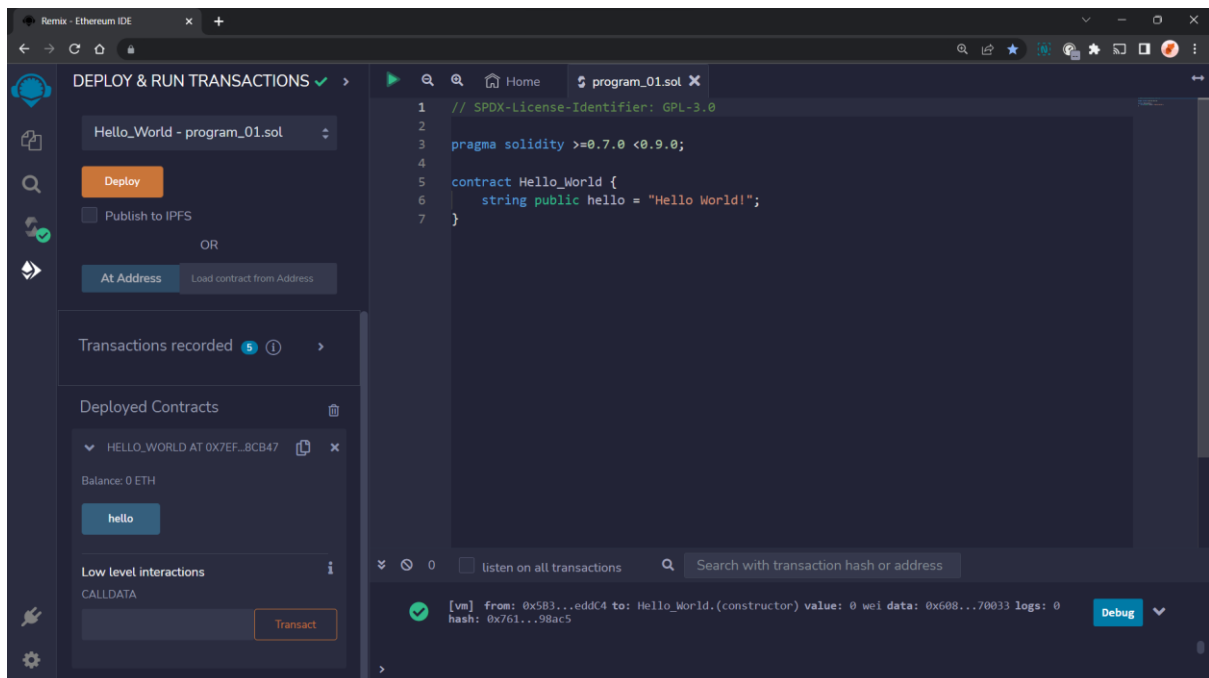


The screenshot shows the Remix Ethereum IDE interface. The main editor displays the source code for `program_01.sol`. The code is as follows:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 contract Hello_World {
6     string public hello = "Hello World!";
7 }
```

The interface includes a sidebar on the left with icons for Explorer, Search, and other tools. The bottom status bar shows "0" transactions and a search field for transaction hashes or addresses.

Output:



The screenshot shows the Remix Ethereum IDE interface after the deployment and execution of the `Hello_World` contract. The left sidebar is expanded to show the "DEPLOY & RUN TRANSACTIONS" panel.

DEPLOY & RUN TRANSACTIONS

- Contract: `Hello_World - program_01.sol`
- Buttons: `Deploy`, `Publish to IPFS`, `At Address`, `Load contract from Address`
- Transactions recorded: 1
- Deployed Contracts: `HELLO_WORLD AT 0X7EF...8CB47`
- Balance: 0 ETH
- Buttons: `hello`, `Transact`
- Low level interactions: `CALLDATA`

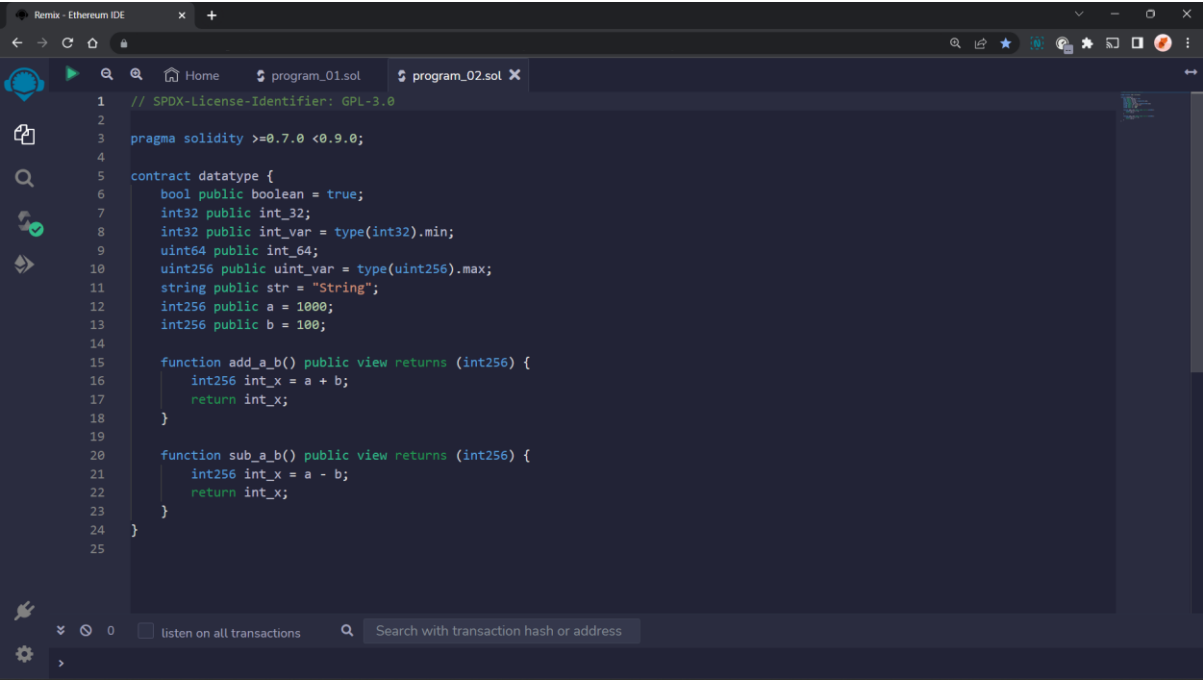
The main editor shows the source code for `program_01.sol`. The bottom status bar shows a green checkmark and the following message:

```
[vm] from: 0x583...edd4 to: Hello_World.(constructor) value: 0 wei data: 0x608...70033 logs: 0
hash: 0x761...98ac5
```

The interface also includes a sidebar on the left with icons for Explorer, Search, and other tools. The bottom status bar shows "0" transactions and a search field for transaction hashes or addresses.

[2]

program_02.sol



Output:

