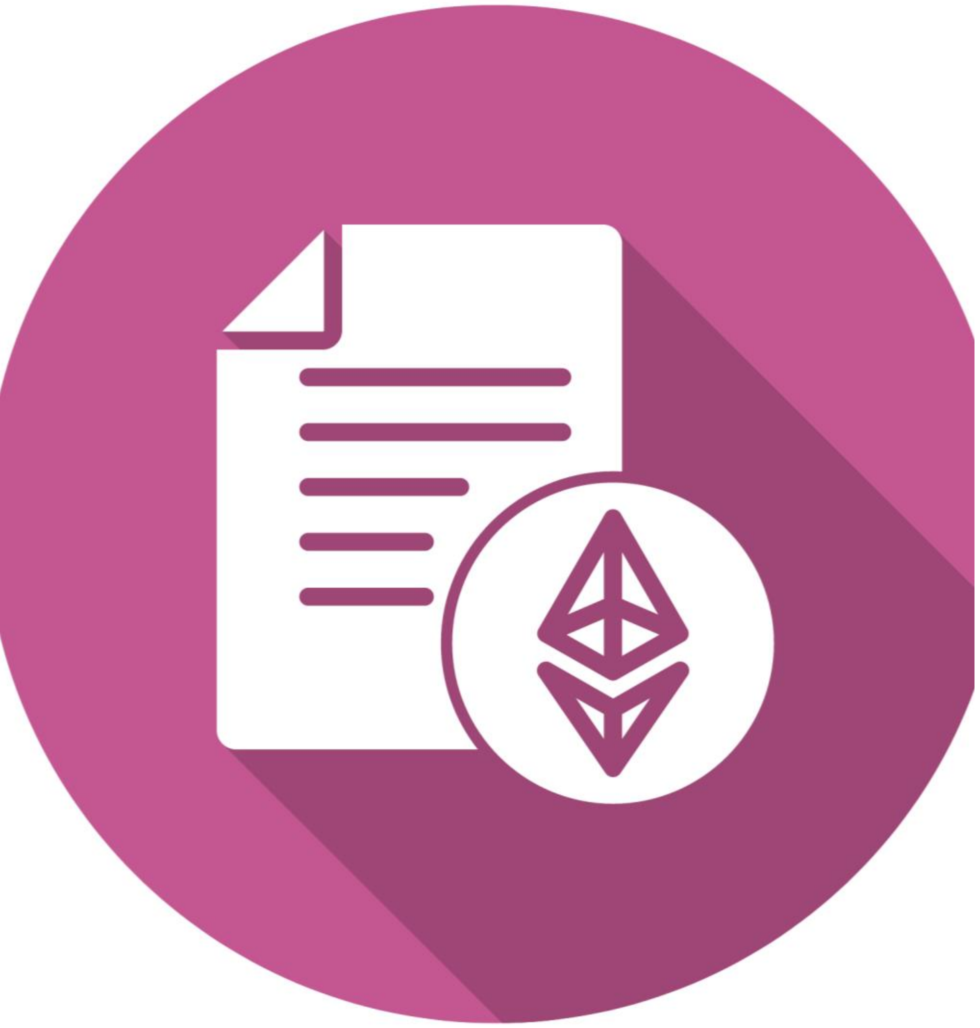
Solidity Basics – 2

Contents



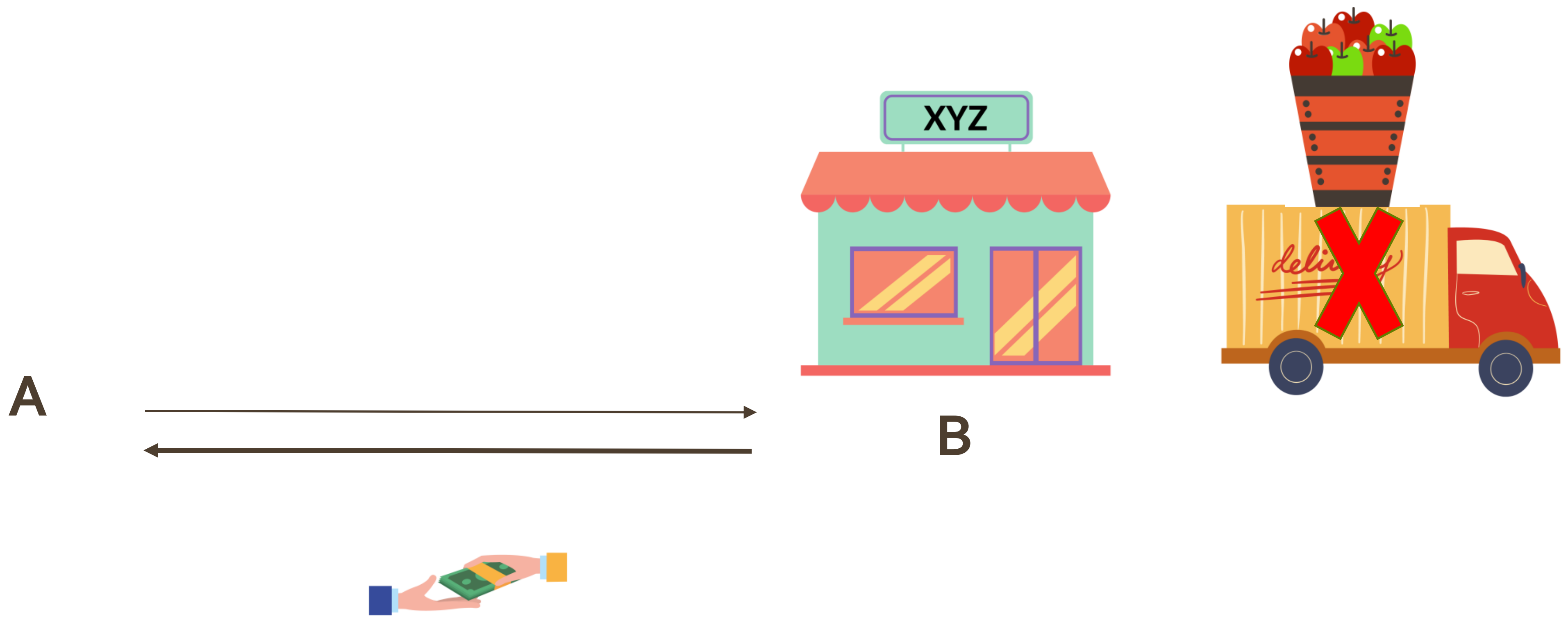
What is a smart contract ?

- Smart contracts are **simply programs stored on a blockchain .**

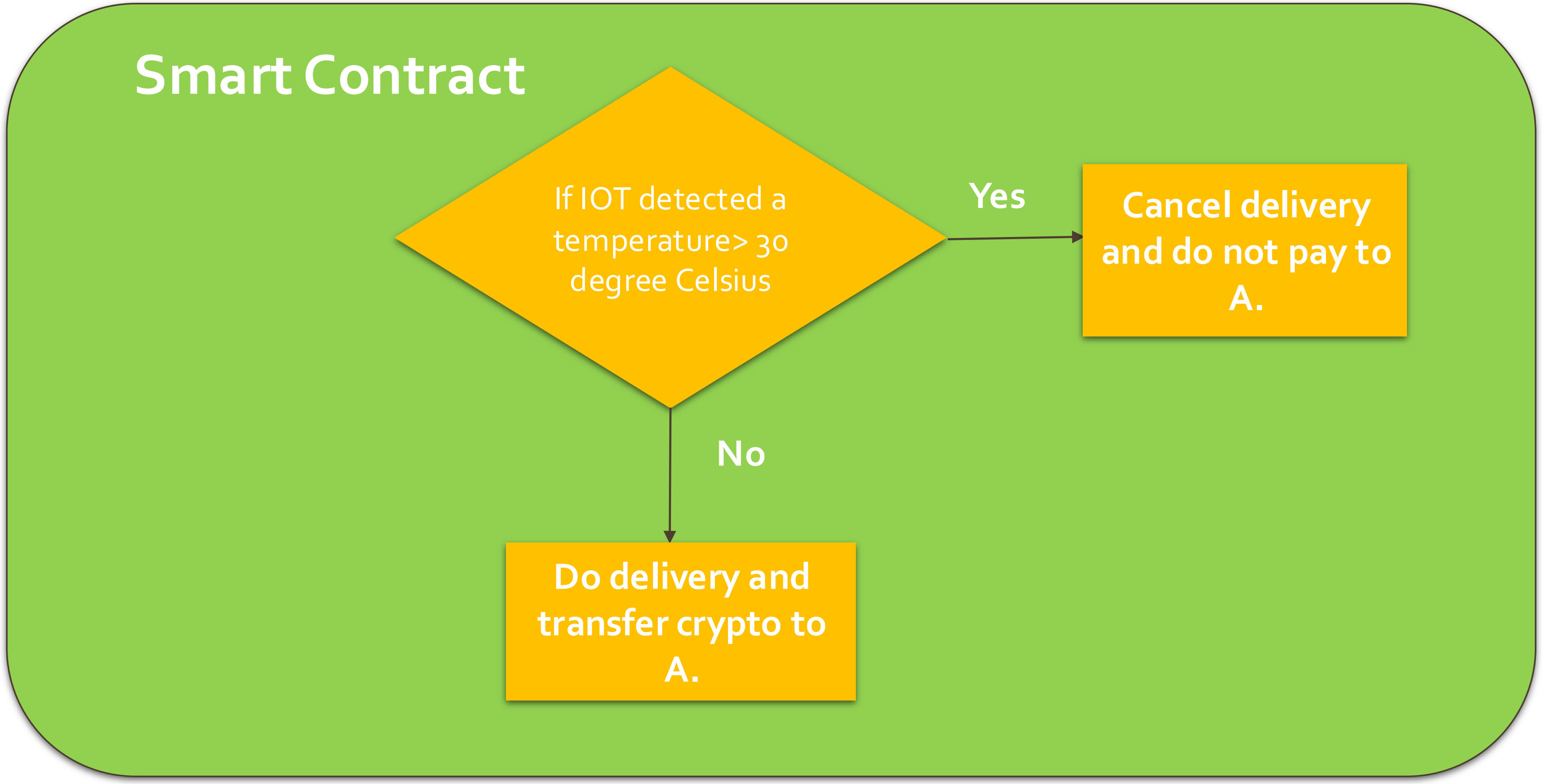
Smart Contract Application

delivery
A
XYZ
B

Smart Contract Application



Smart Contract Application


Smart Contract
If IOT detected a temperature> 30 degree Celsius
Yes
Cancel delivery and do not pay to A.
No
Do delivery and transfer crypto to A.

Note-Assuming optimum temperature <30 degree Celsius.

Smart Contract Features

- Smart Contracts are immutable as they get stored on Blockchain.
- Smart contract contracts have their own accounts where it can store cryptocurrency.
- No human intervention is required for cryptocurrency transfer or receiving.

Question Time

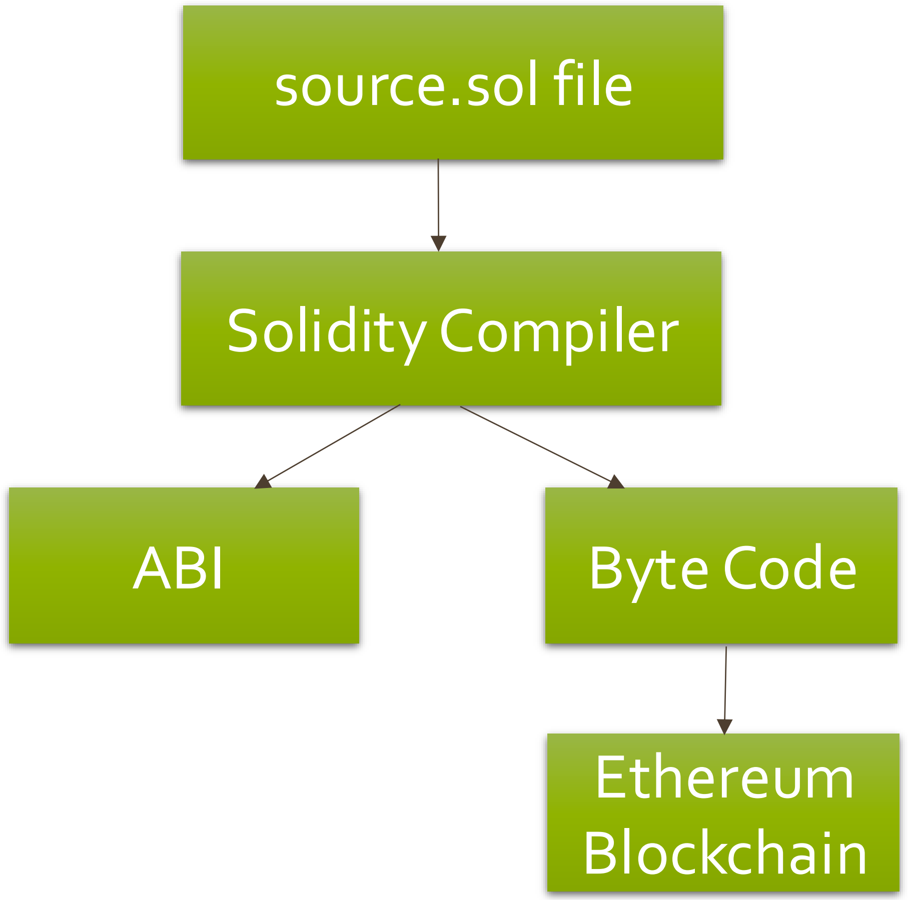
What is solidity?

- Solidity is an object-oriented programming language for implementing smart contracts for the ethereum blockchain.
- High-level statically typed programming language.
- Case sensitive.
- With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

Note – You should follow established development best-practices when writing your smart contracts.

Question Time

Solidity Compilation Process



Solidity Compilation Process

- Contract bytecode is public in readable form.
- Contract doesn’t have to be public.
- Bytecode is immutable because it is getting stored on Blockchain.
- ABI act as a bridge between applications and smart contract.
- ABI and Bytecode cannot be generated without source code.

Question Time

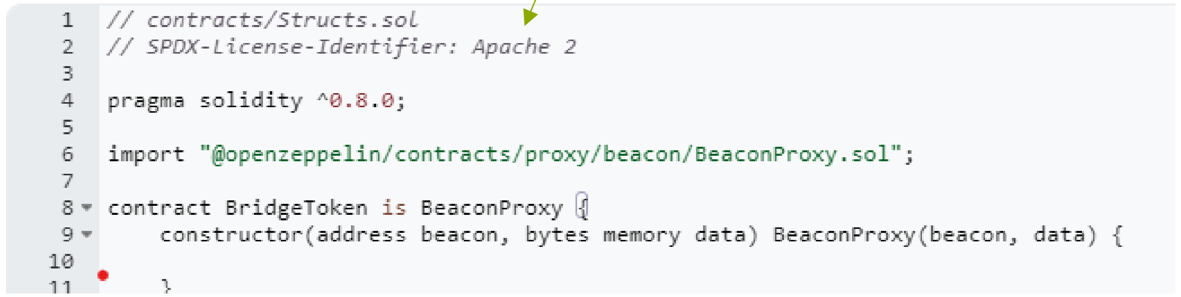
SPDX

- Trust in smart contracts can be better established if their source code is available. Since making source code available always touches on legal problems with regards to copyright, the Solidity compiler encourages the use of machine-readable SPDX license identifiers. Every source file should start with a comment indicating its license.
- Before publishing, consider adding a comment containing "**SPDX-License-Identifier: <SPDX-License>**" to each source file.
- Use "**SPDX-License-Identifier: UNLICENSED**" for non-open-source code.
- Please see SPDX & Etherscan for more information.

SPDX

SPDX License

File 1 of 7 : Token.sol

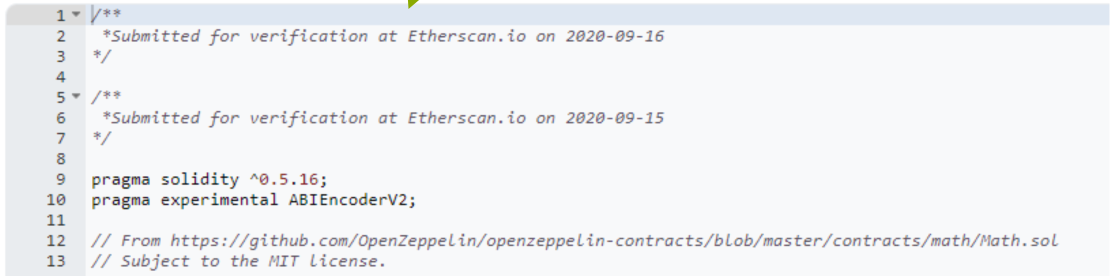


```
// contracts/Structs.sol
// SPDX-License-Identifier: Apache 2

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol";

contract BridgeToken is BeaconProxy {
    constructor(address beacon, bytes memory data) BeaconProxy(beacon, data) {

    }
```

Wrapped BNB (Wormhole) (WBNB)

License : None

Contract Source Code (Solidity)

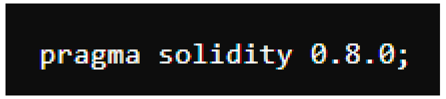


```
/**
 *Submitted for verification at Etherscan.io on 2020-09-16
*/

/**
 *Submitted for verification at Etherscan.io on 2020-09-15
*/

pragma solidity ^0.5.16;
pragma experimental ABIEncoderV2;

// From https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/Math.sol
// Subject to the MIT license.
```

UNI Token

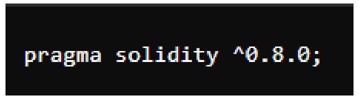
Pragma

- In Solidity, pragma is a directive that specifies the compiler version or configurations that the source file should use. It helps ensure that the code is compiled with the appropriate version of the Solidity compiler, preventing potential issues that could arise from using different compiler versions. The most common use of pragma is to set the compiler version.



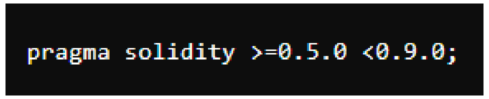
```
pragma solidity 0.8.0;
```

The directive pragma solidity 0.8.0; specifies that the Solidity compiler version must be exactly 0.8.0 to compile the smart contract.



```
pragma solidity ^0.8.0;
```

The caret ^ specifies that the compiler version should be **0.8.0 or any newer version that is backward compatible with 0.8.0**, but not 0.9.0 or later.

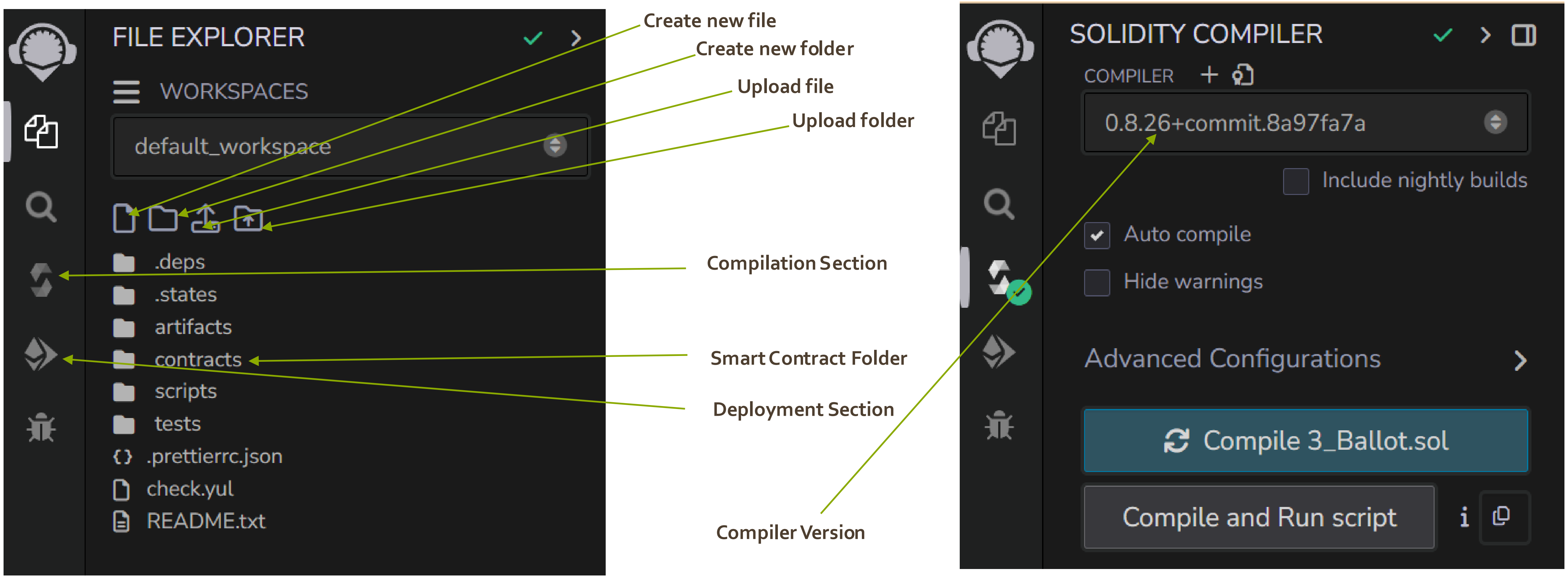


```
pragma solidity >=0.5.0 <0.9.0;
```

This means any version from 0.5.0 up to, but not including, 0.9.0 can be used.This range is useful for ensuring compatibility across multiple versions while avoiding breaking changes introduced in 0.9.0 and beyond.

Question Time

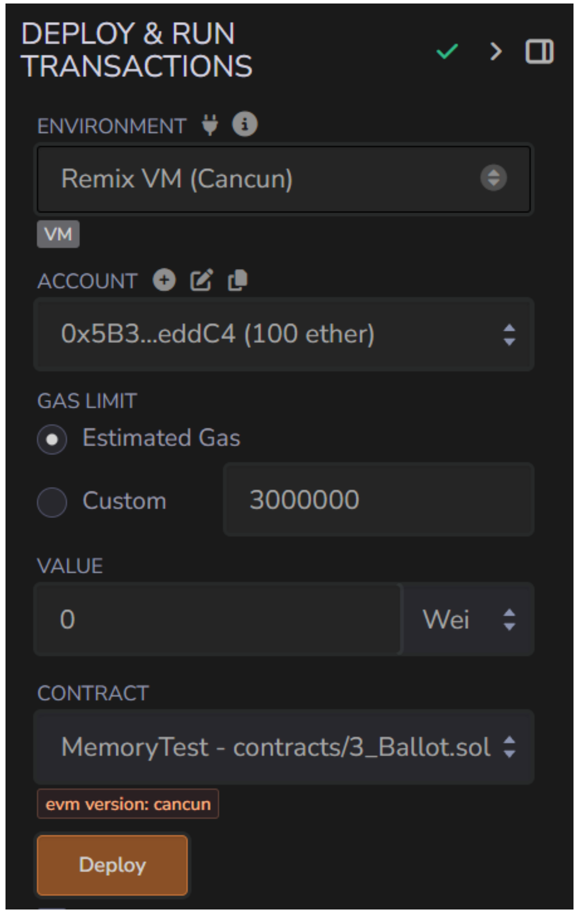
Remix IDE



Left Side Pannel

Compilation Section

Remix IDE


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Remix VM (Cancun)

VM

ACCOUNT

0x5B3...eddC4 (100 ether)

GAS LIMIT

Estimated Gas [✓]

Custom 3000000

VALUE

0 Wei

CONTRACT

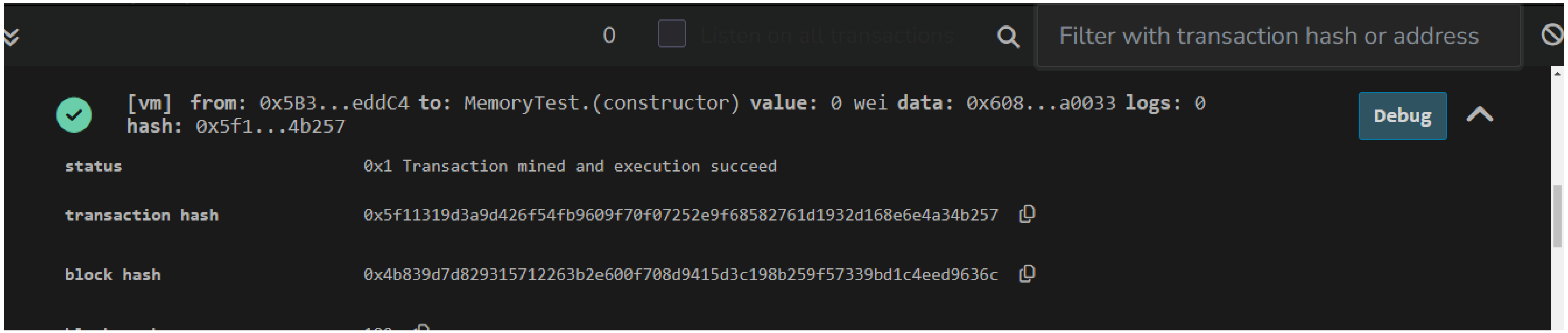
MemoryTest - contracts/3_Ballot.sol

evm version: cancun

Deploy

Deployment Section

Remix IDE

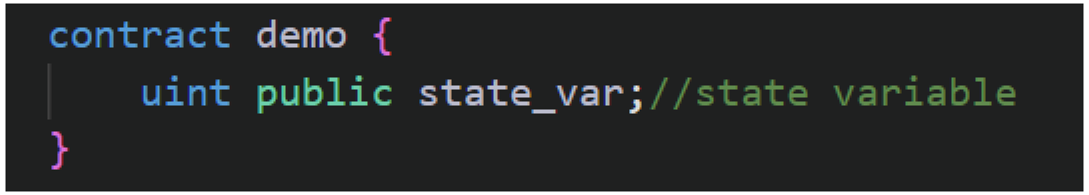

0
Listen on all transactions
Filter with transaction hash or address
Debug

```
[vm] from: 0x5B3...eddC4 to: MemoryTest.(constructor) value: 0 wei data: 0x608...a0033 logs: 0
hash: 0x5f1...4b257

status              0x1 Transaction mined and execution succeed

transaction hash    0x5f11319d3a9d426f54fb9609f70f07252e9f68582761d1932d168e6e4a34b257

block hash          0x4b839d7d829315712263b2e600f708d9415d3c198b259f57339bd1c4eed9636c
```

Transaction Status Section

Question Time

State Variables

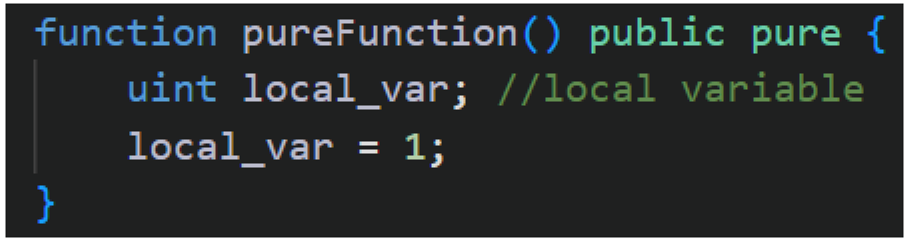


```
contract demo {
    uint public state_var;//state variable
}
```

- Permanently stored in contract storage.
- Cost gas(expensive) .
- Reading of state variable is free but writing to it is costly.

Question Time

Local Variables

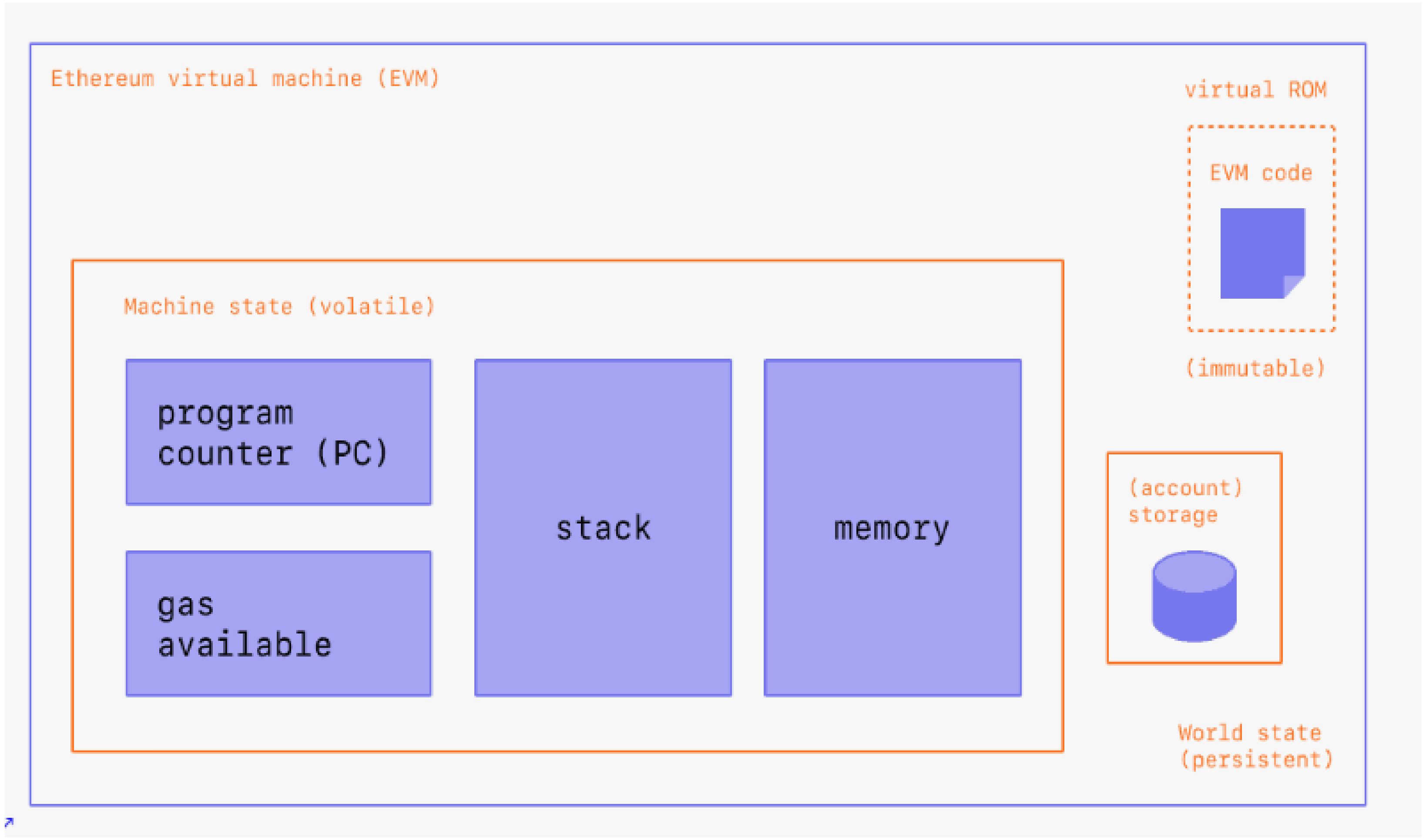


```
function pureFunction() public pure {
    uint local_var; //local variable
    local_var = 1;
}
```

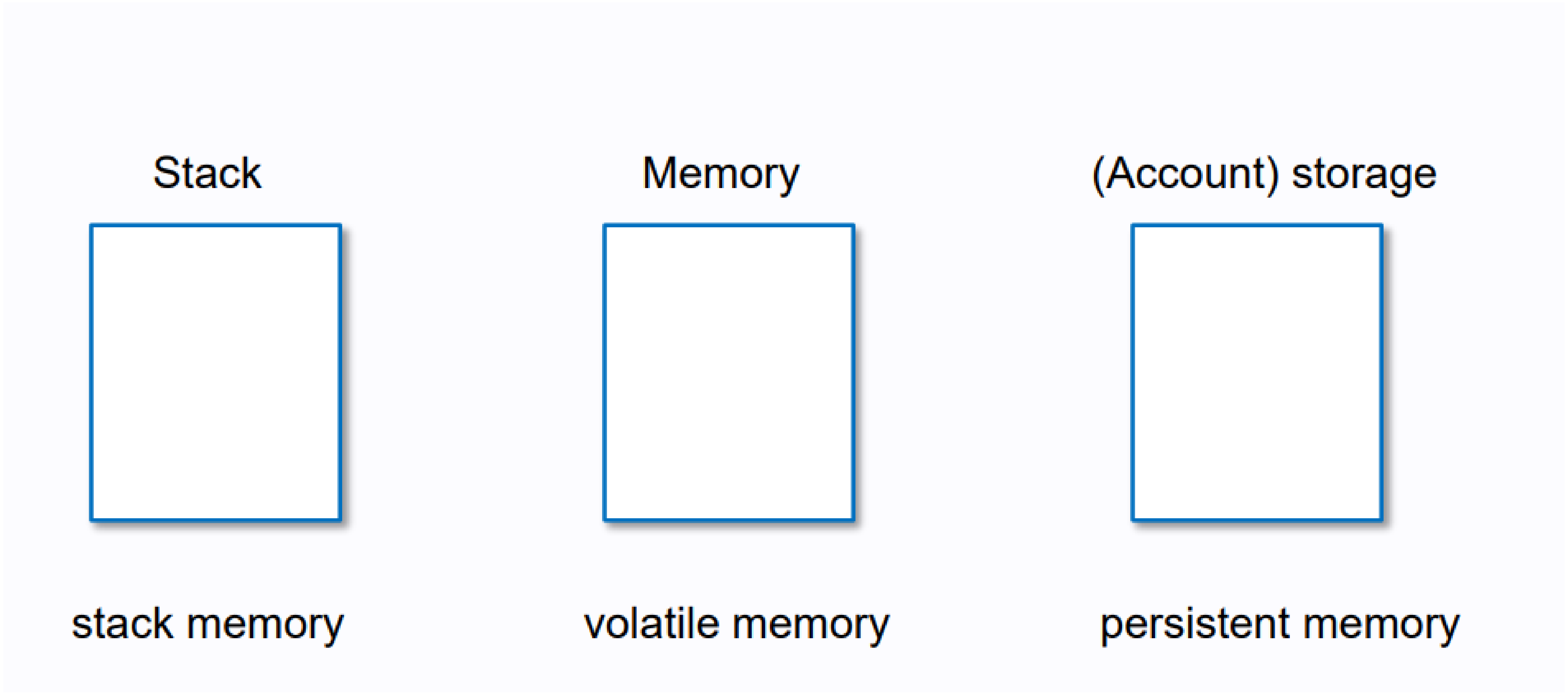
- Declared inside functions and are kept on the stack , not on storage.
- Don’t cost gas.

Question Time

EVM


Ethereum virtual machine (EVM)
virtual ROM
EVM code
(immutable)
Machine state (volatile)
program counter (PC)
gas available
stack
memory
(account) storage
World state (persistent)

Storage Area



Question Time

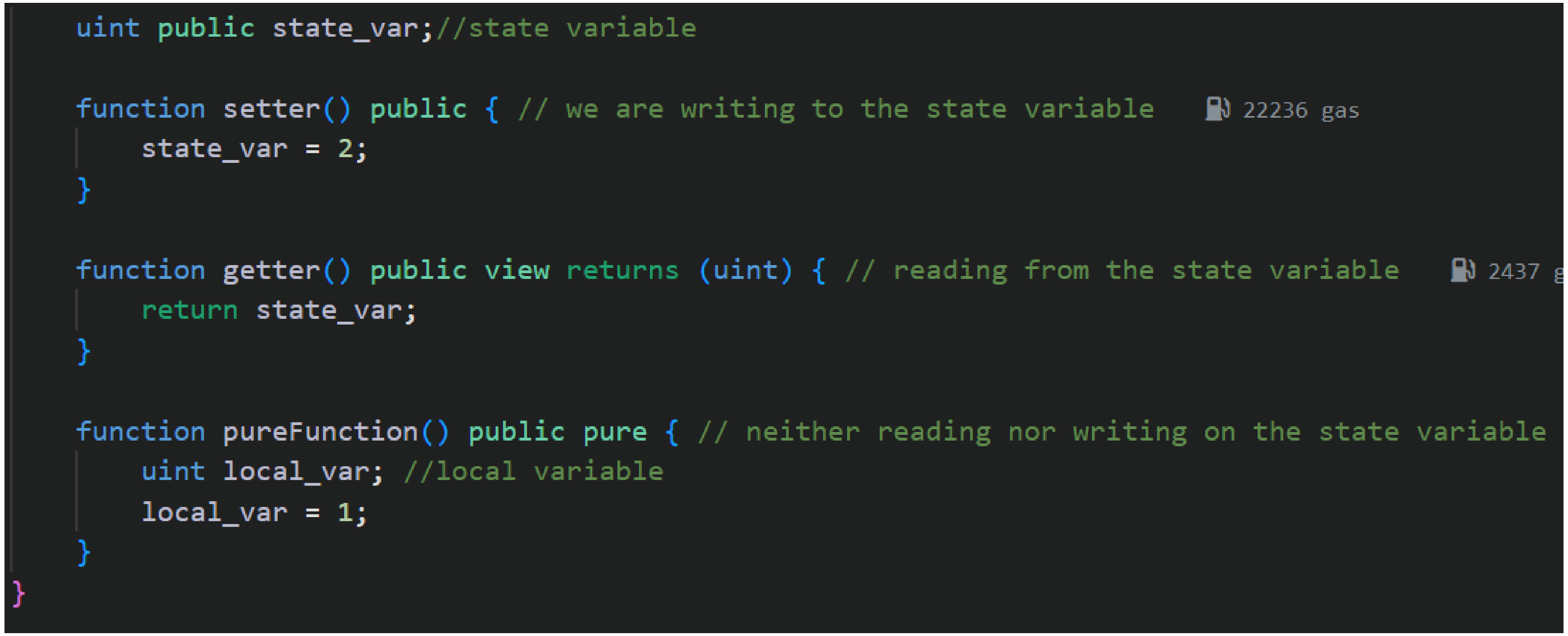
Functions

- When you declare a public state variable a getter function is automatically created.
- For public state variables a getter() function is automatically created.

View Vs Pure

Function Type	State Variable	
	Read	Write
View	✓	✗
Pure	✗	✗
		✓

Functions Example - 1



```
    uint public state_var;//state variable

    function setter() public { // we are writing to the state variable    22236 gas
        state_var = 2;
    }

    function getter() public view returns (uint) { // reading from the state variable    2437 g
        return state_var;
    }

    function pureFunction() public pure { // neither reading nor writing on the state variable
        uint local_var; //local variable
        local_var = 1;
    }
}
```

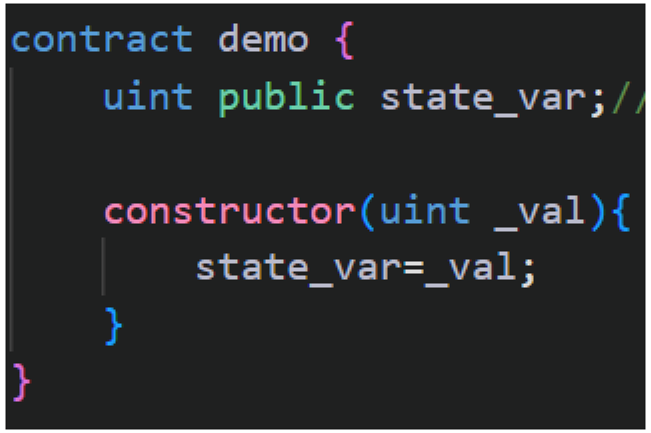
Code – Click Here

Function Example - 2

- Click Here

Question Time

Constructor


```
contract demo {
    uint public state_var;//

    constructor(uint _val){
        state_var=_val;
    }
}
```

- Executed only once.
- You can create only one constructor and that is optional.
- A default constructor is created by the compiler if there is no explicitly defined constructor.

Question Time

Basic Data Types

- Integer Data Type
- Bool Data Type
- Address Data Type
- Bytes Data Type

Integer Data Type

int	uint
int8 to int256	uint8 to uint256
int alias to int256	uint alias to uint256

Signed and Unsigned integers can be of various sizes.

By default int and uint are initialized to zero.

Overflow get detected at compile time.

Integer Data Type

Range	
int8 : - 128 to +127	**uint8** : 0 to 255
int16 : - 32768 to +32767	**uint16** : 0 to 65535
-2^(n-1) to 2^(n-1)-1	0 to 2^(n)-1

Question Time

Bool Data Type

- bool public value = true;
- Bool data type value can be either true or false.
- By default value is false if not initialized.

Address Data Type

- address public addr = "0xBE4024Fa7461933F930DD3CEf5D1a01363E9f284"
- The `address` type is a 160-bit value that does not allow any arithmetic operations.

Question Time

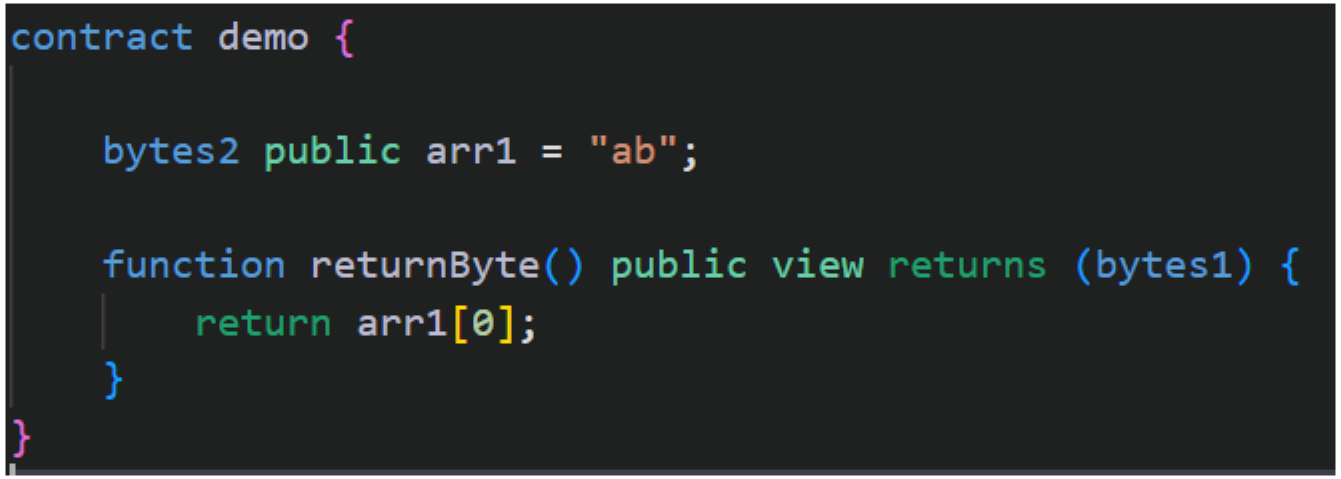
Bytes Data Type

- Bytes data type is used to store strings. Range - bytes1, bytes2, …..,bytes32.
- It stores characters.
- bytes1 public arr1="a";
- bytes2 public arr2="ab";
- bytes3 public arr3="abc";
- Everything that will be stored in the bytes array will be in hexadecimal number.
- arr3 will look this

61	62	63
0	1	2

- Click Here – Character To Hexadecimal Table
- Padding of 0 takes place if initialized characters are less than the byte size.

Bytes Data Type

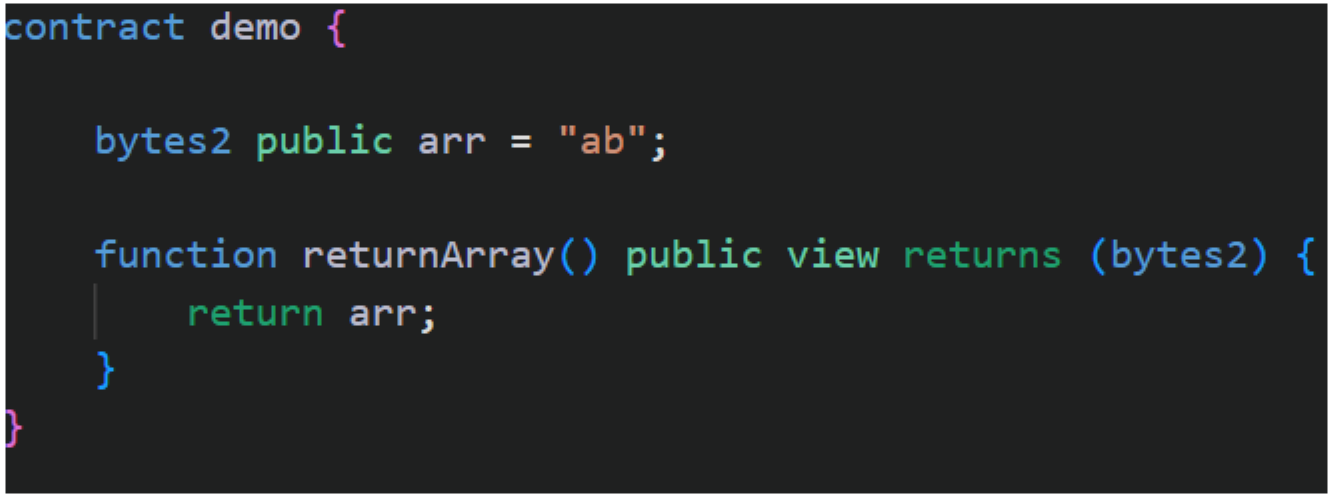


```
contract demo {

    bytes2 public arr1 = "ab";

    function returnByte() public view returns (bytes1) {
        return arr1[0];
    }
}
```

Output - 0x61

Example - 1

Bytes Data Type



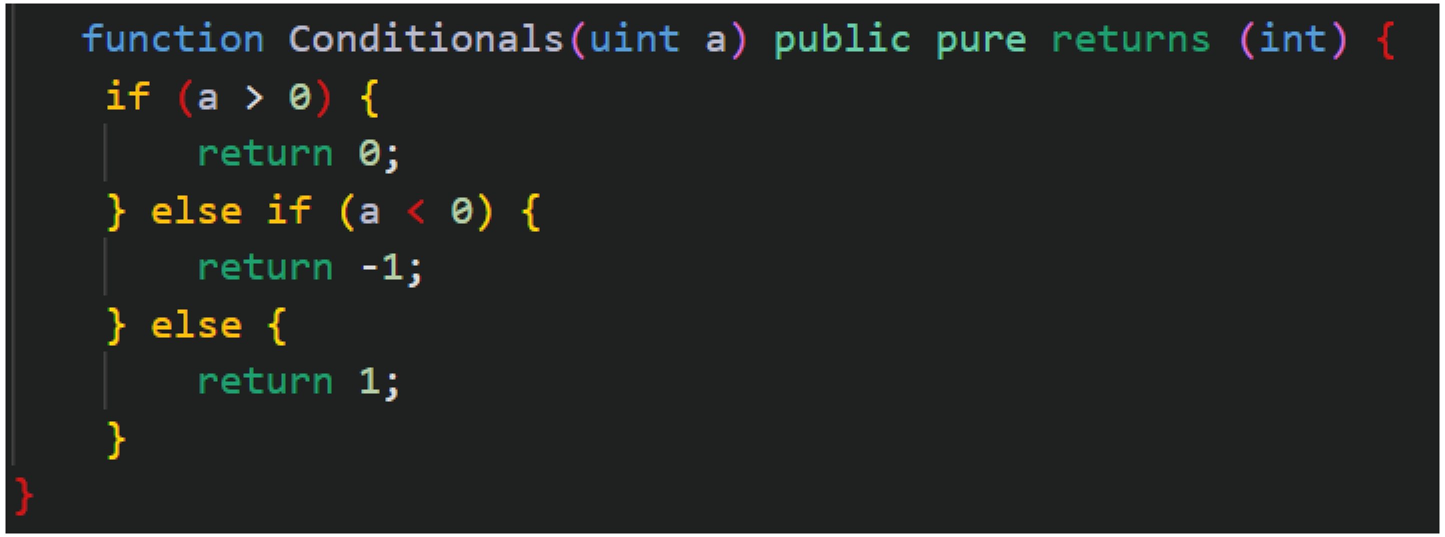
```
contract demo {

    bytes2 public arr = "ab";

    function returnArray() public view returns (bytes2) {
        return arr;
    }
}
```

Output - 0x6162

Example - 2

Question Time

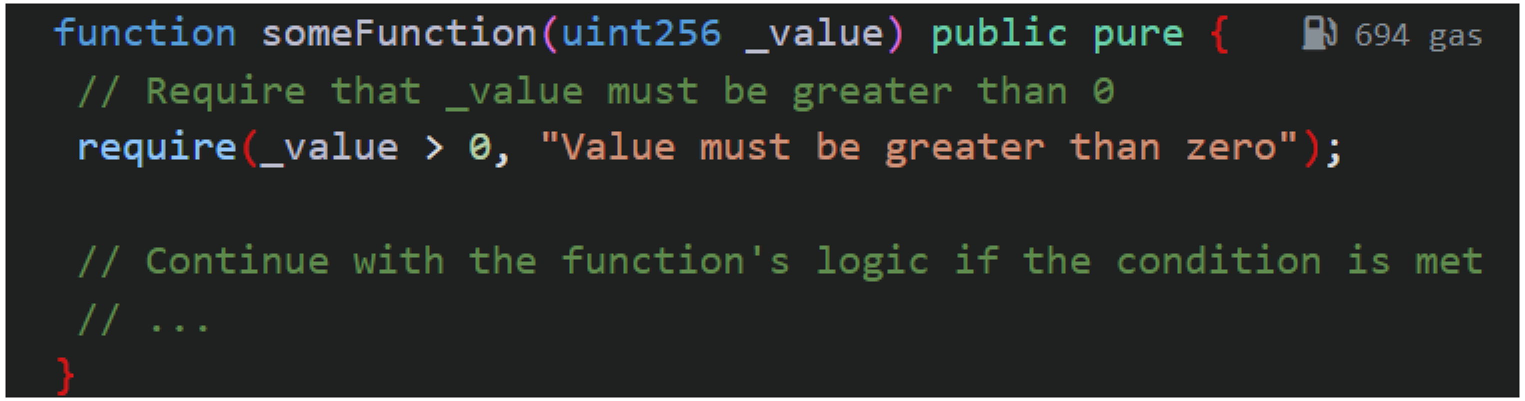
Conditionnels



```
    function Conditionals(uint a) public pure returns (int) {
     if (a > 0) {
         return 0;
     } else if (a < 0) {
         return -1;
     } else {
         return 1;
     }
}
```

Code

Require



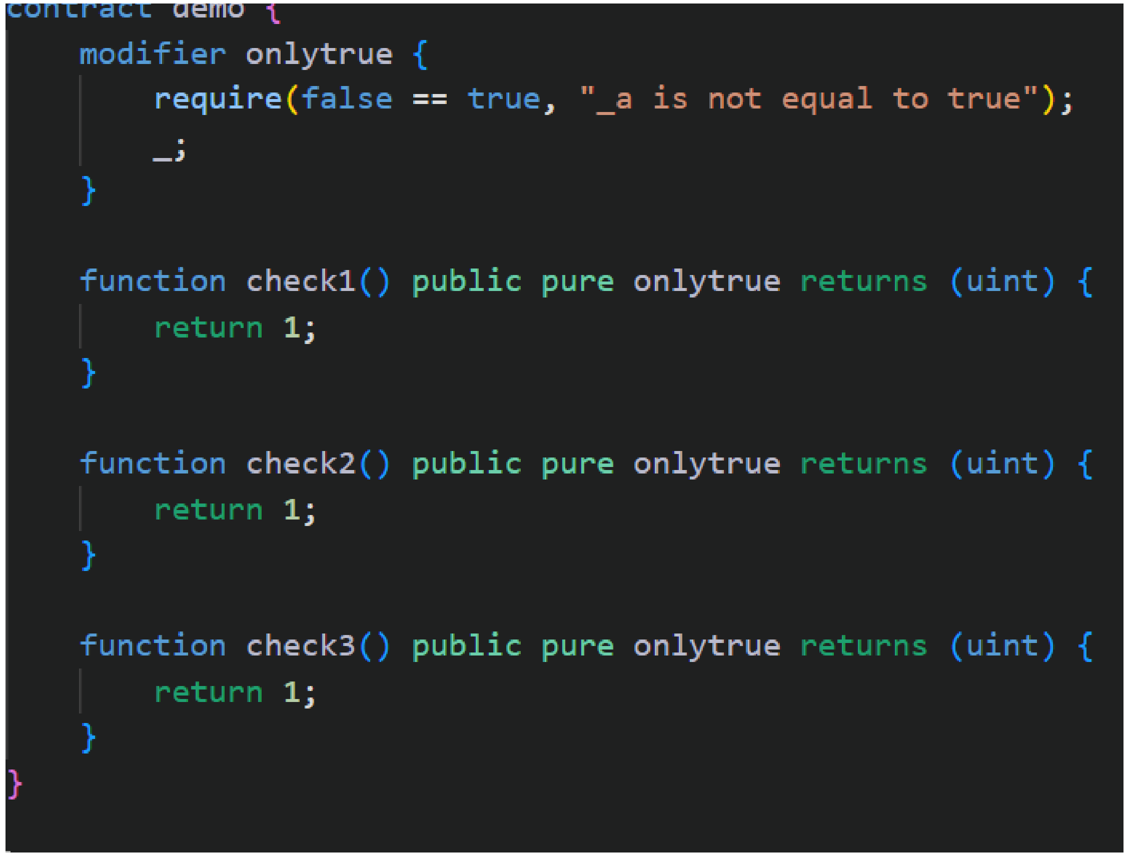
```
function someFunction(uint256 _value) public pure {    694 gas
 // Require that _value must be greater than 0
 require(_value > 0, "Value must be greater than zero");

 // Continue with the function's logic if the condition is met
 // ...
}
```

[Code]

Require

Example 2 - [Code]

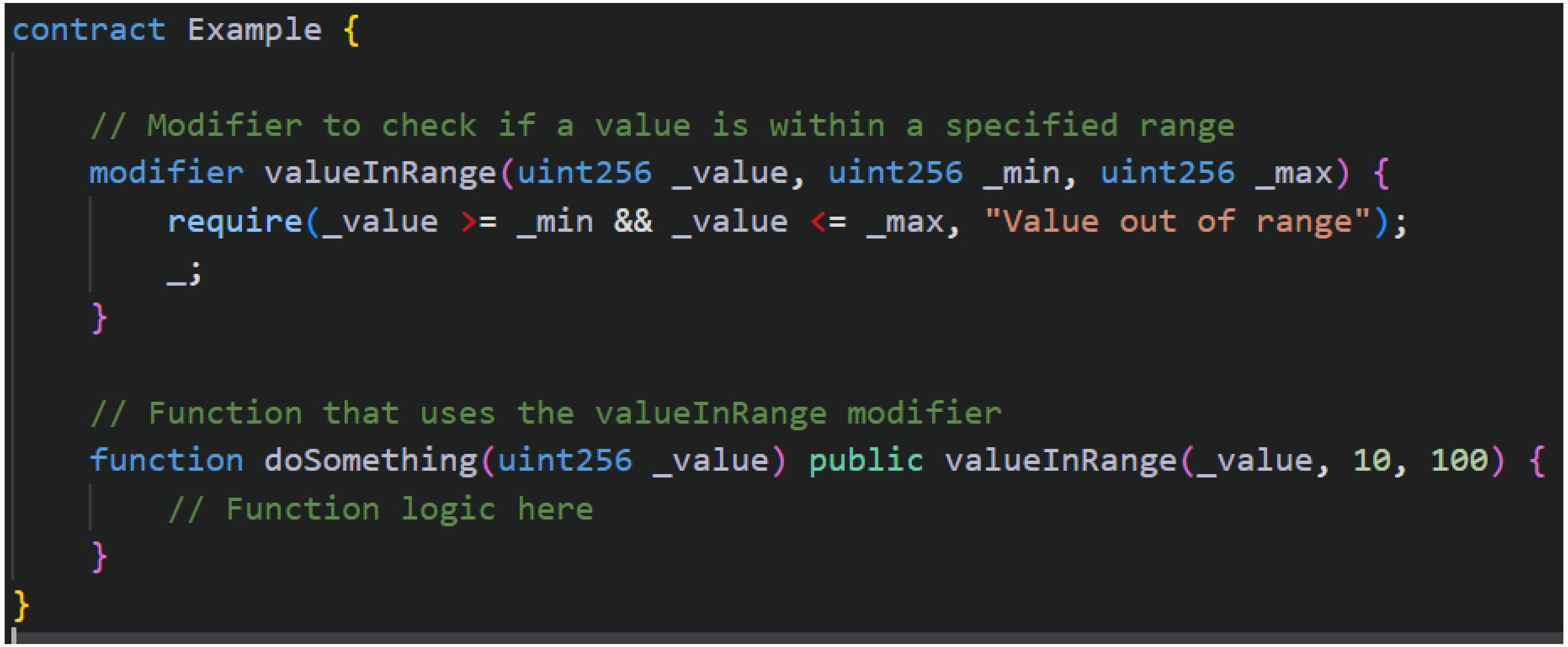
Modifier



```
contract demo {
    modifier onlytrue {
        require(false == true, "_a is not equal to true");
        _;
    }

    function check1() public pure onlytrue returns (uint) {
        return 1;
    }

    function check2() public pure onlytrue returns (uint) {
        return 1;
    }

    function check3() public pure onlytrue returns (uint) {
        return 1;
    }
}
```

Code

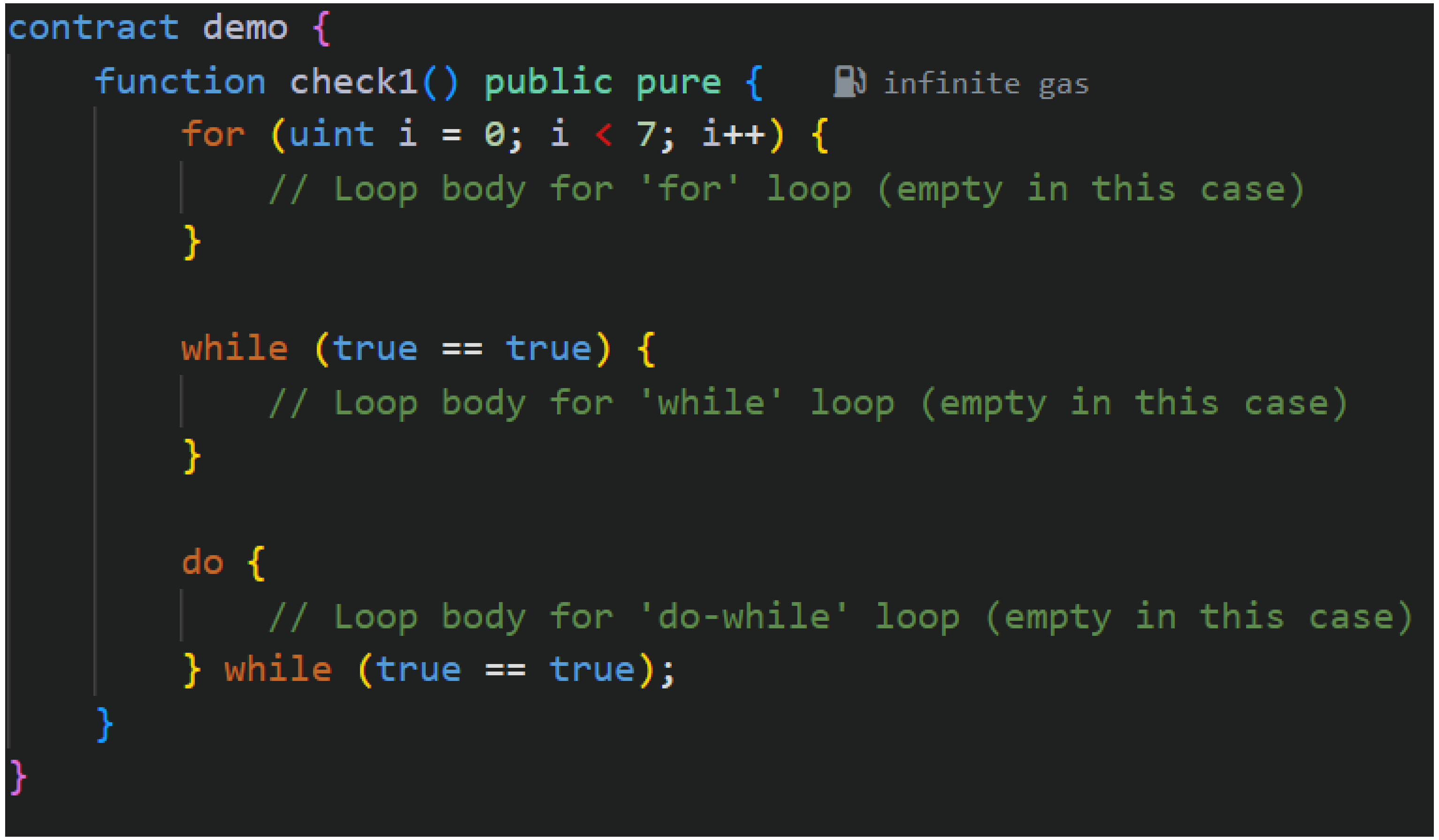
Modifier


```
contract Example {

    // Modifier to check if a value is within a specified range
    modifier valueInRange(uint256 _value, uint256 _min, uint256 _max) {
        require(_value >= _min && _value <= _max, "Value out of range");
        _;
    }

    // Function that uses the valueInRange modifier
    function doSomething(uint256 _value) public valueInRange(_value, 10, 100) {
        // Function logic here
    }
}
```

Code

Question Time

Loop



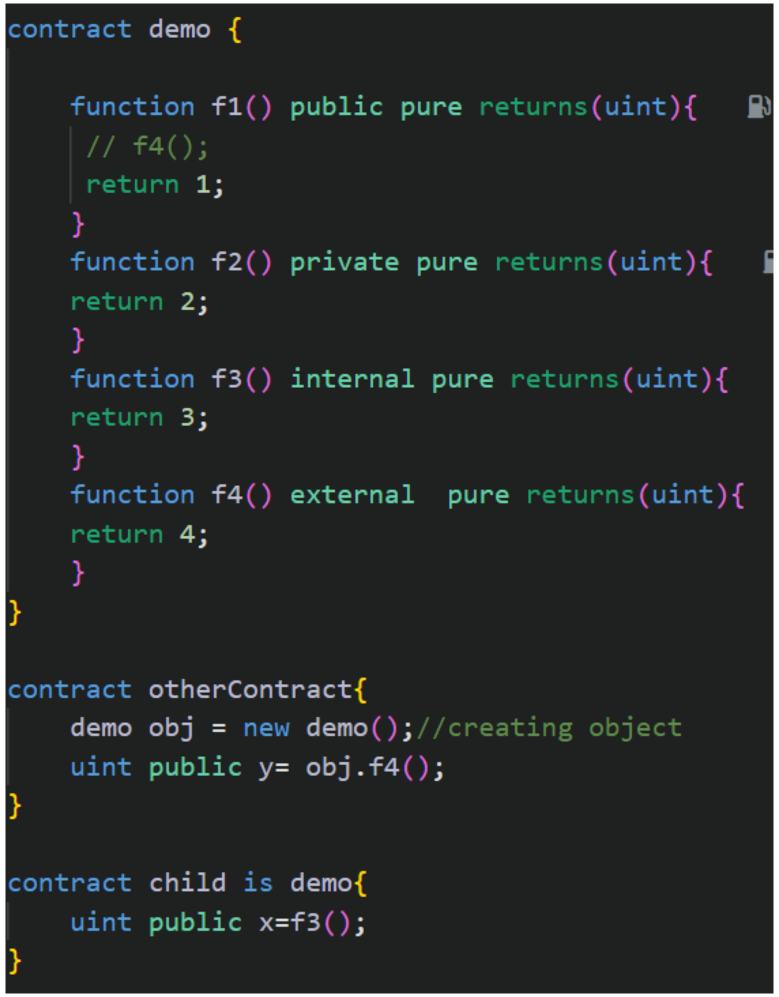
```
contract demo {
    function check1() public pure {    infinite gas
        for (uint i = 0; i < 7; i++) {
            // Loop body for 'for' loop (empty in this case)
        }

        while (true == true) {
            // Loop body for 'while' loop (empty in this case)
        }

        do {
            // Loop body for 'do-while' loop (empty in this case)
        } while (true == true);
    }
}
```

Code

Visibility

	PUBLIC	PRIVATE	INTERNAL	EXTERNAL
Outside World	✓			✓
Within Contract	✓	✓	✓	
Derived Contract	✓		✓	✓
Other Contracts	✓			✓

Visibility


```
contract demo {

    function f1() public pure returns(uint){
     // f4();
     return 1;
    }
    function f2() private pure returns(uint){
    return 2;
    }
    function f3() internal pure returns(uint){
    return 3;
    }
    function f4() external  pure returns(uint){
    return 4;
    }
}

contract otherContract{
    demo obj = new demo();//creating object
    uint public y= obj.f4();
}

contract child is demo{
    uint public x=f3();
}
```

Code

Question Time

Thank You

- Please Like and Subscribe :)
- Instagram - @codeeater21
- LinkedIn - @KshitijWeb3