Solidity Basics – 2

Contents

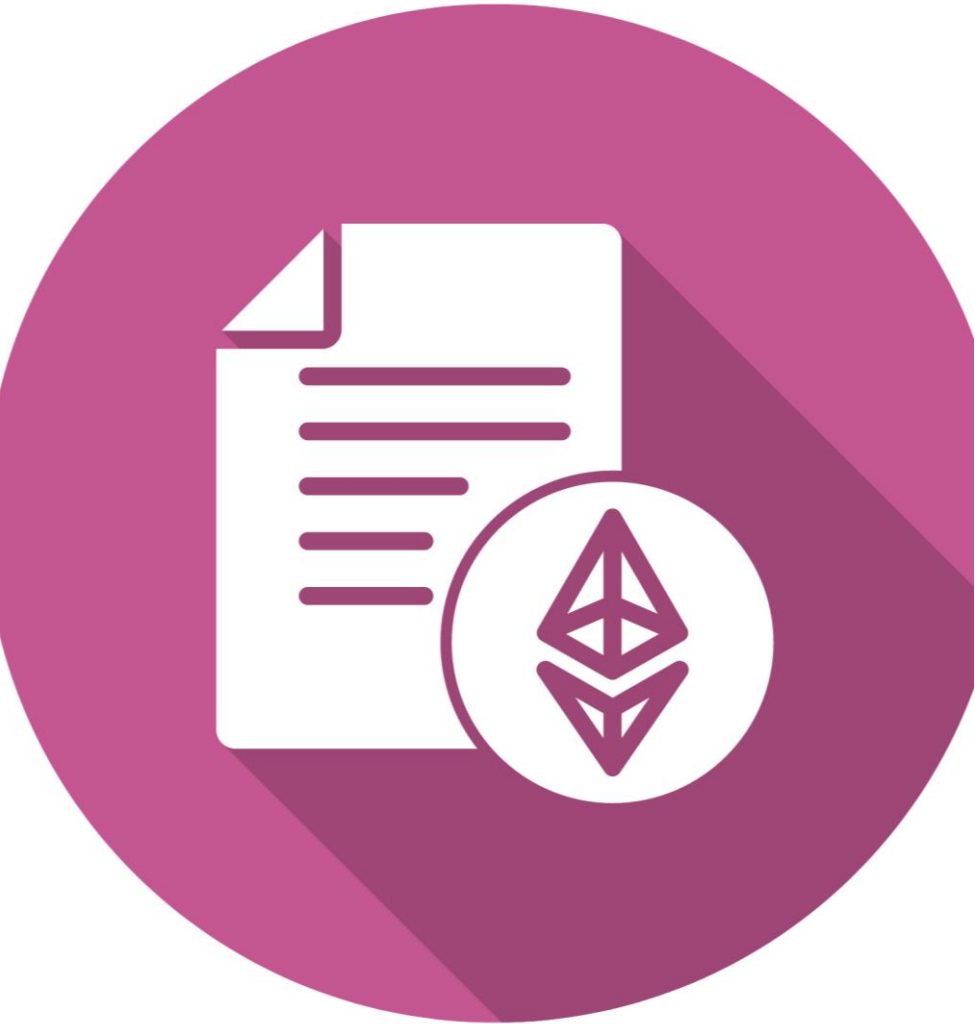|		
What is a smart contract ?	Remix IDE	Visibility Specifier
Smart Contract Application	State Variables	Conditionals
What is solidity ?	Local Variables	Basic Data Types
Solidity Compilation Process	Functions	Much more

What is a smart contract ?

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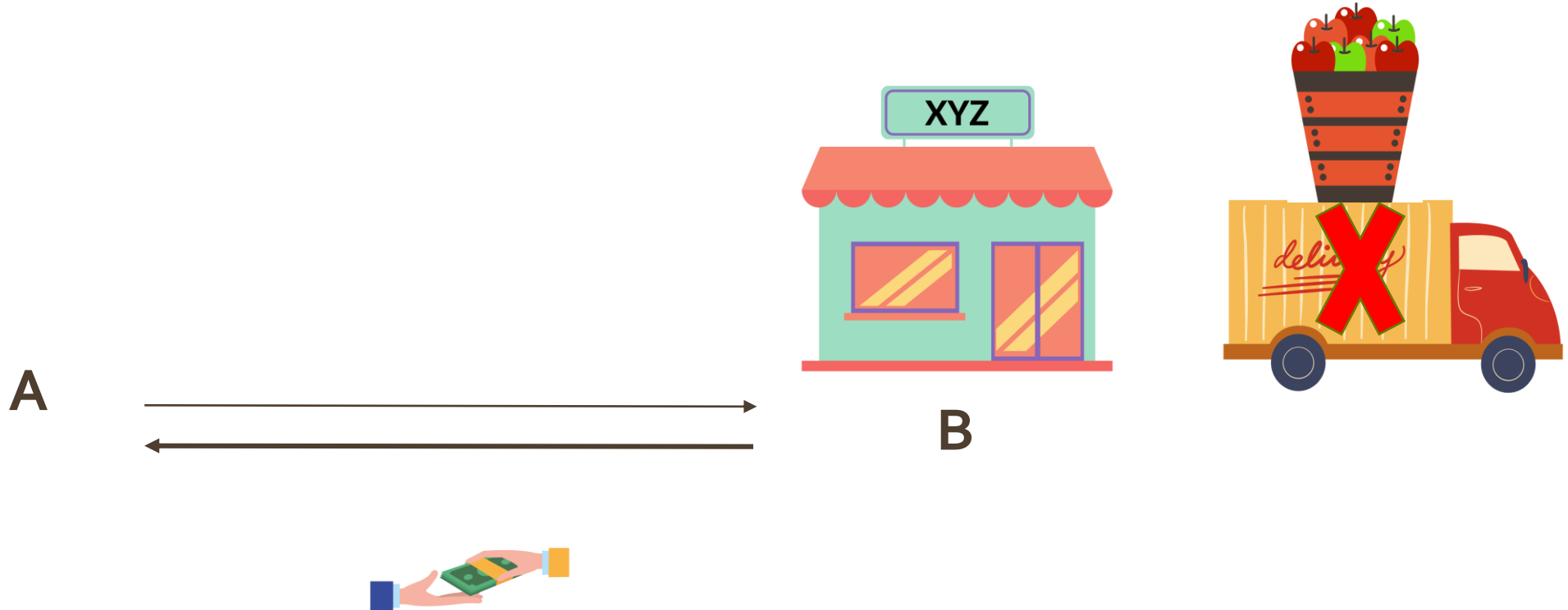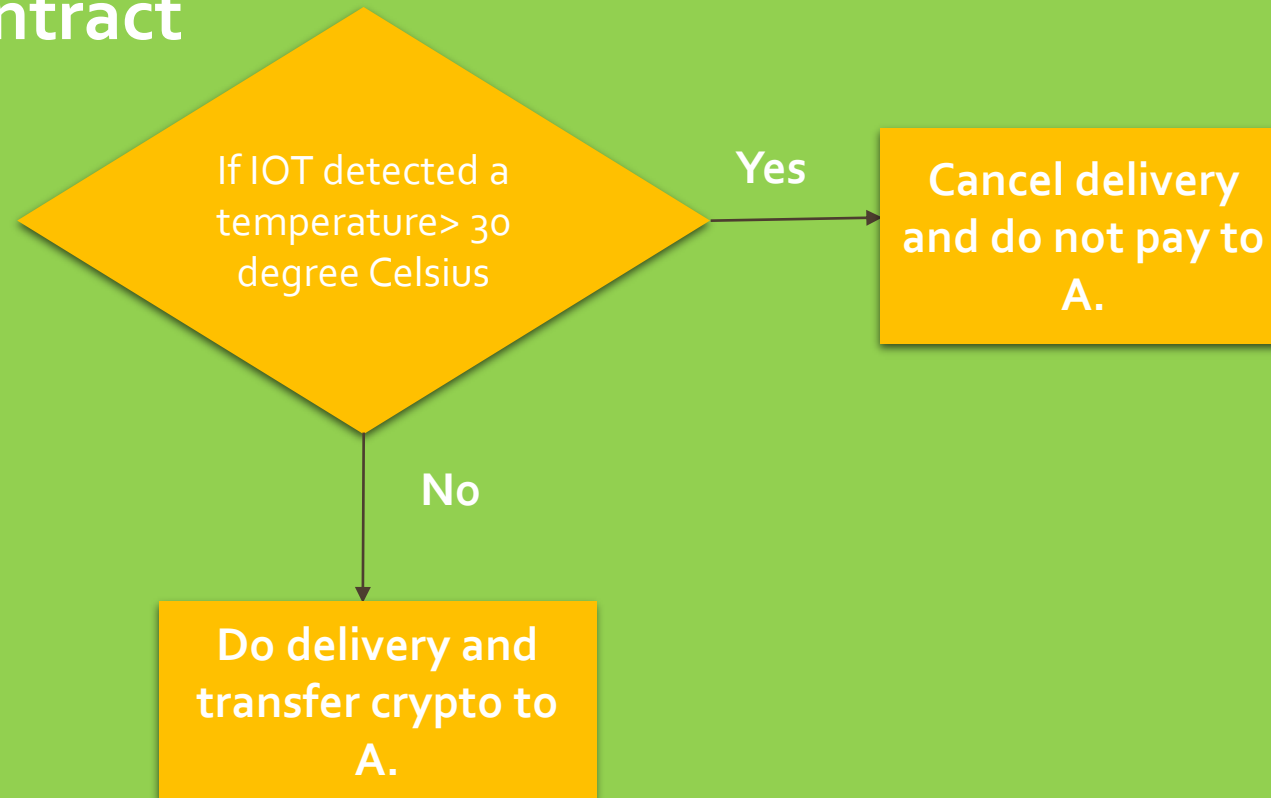
Smart Contract Application

A →→→→ 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

Remix IDE

- Sample program with [Remix IDE](#)

Question Time

Solidity Compilation Process
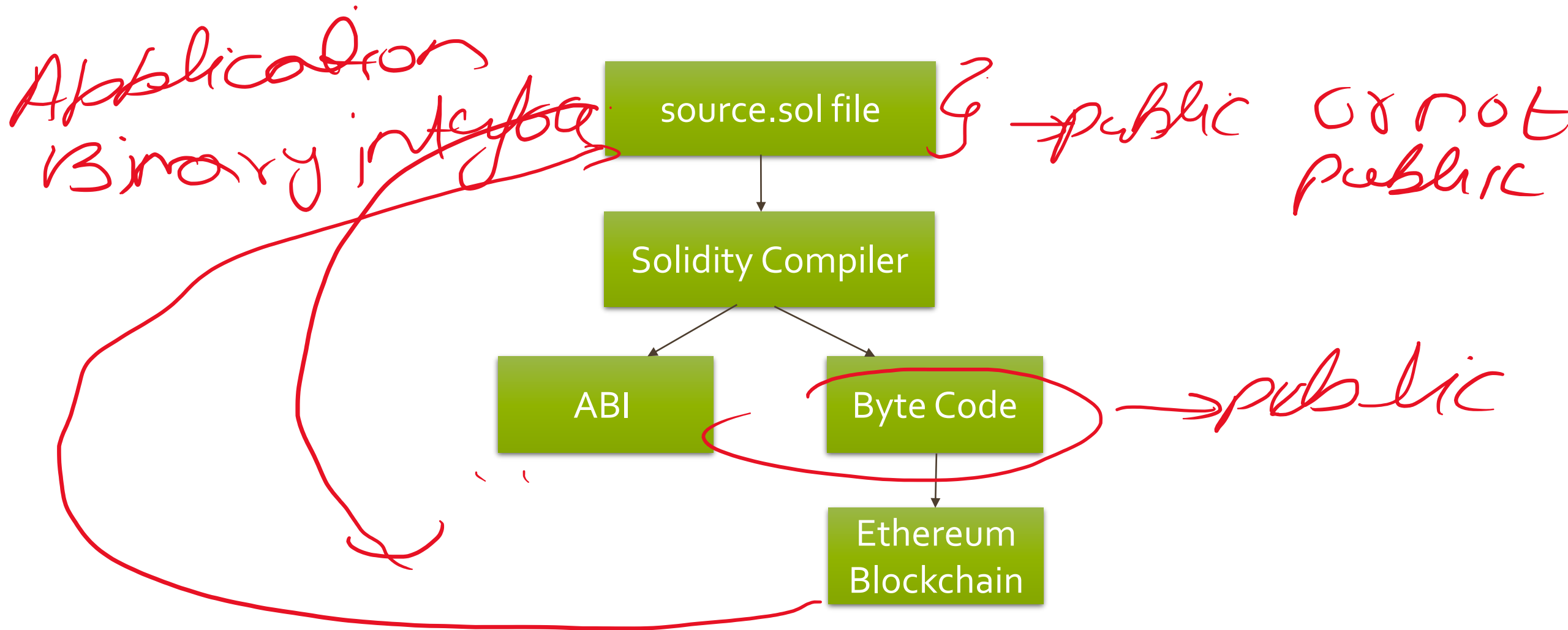
Solidity Compilation Process

- 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 for more information.

State Variables

- Permanently stored in contract storage.

- Cost gas(expensive) .

- Reading of state variable is free but writing to it is costly.

Question Time

Local Variables

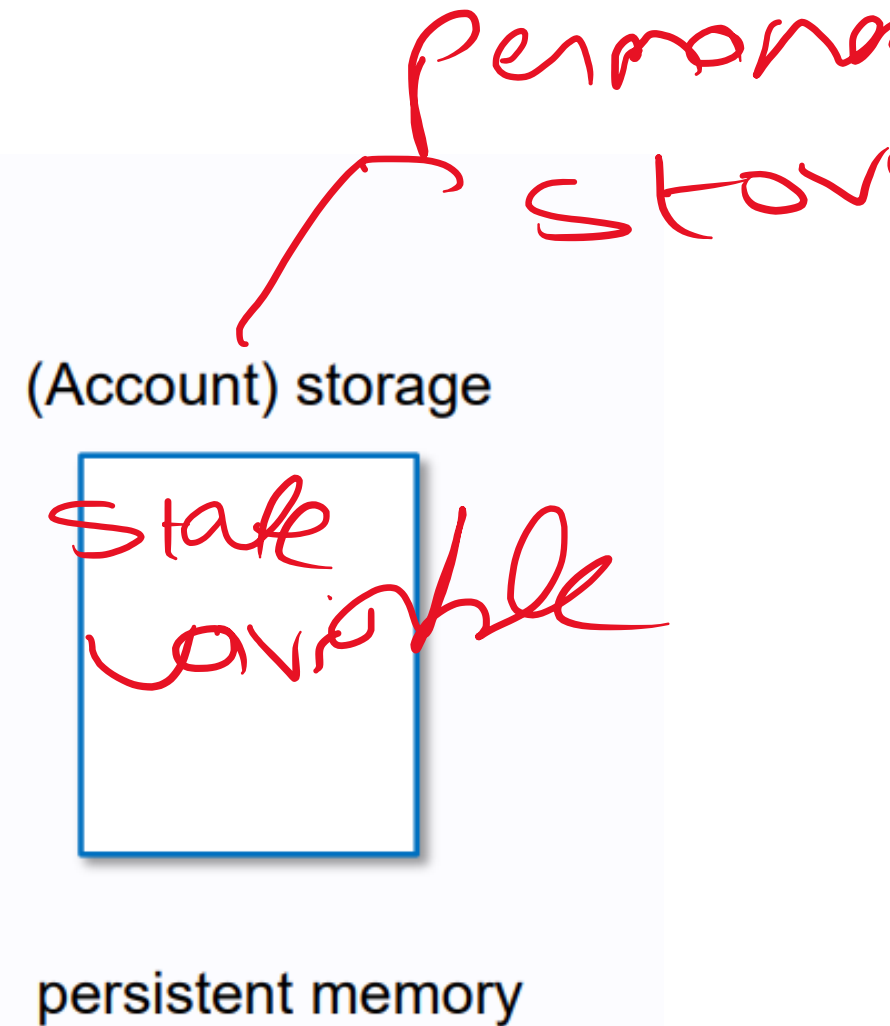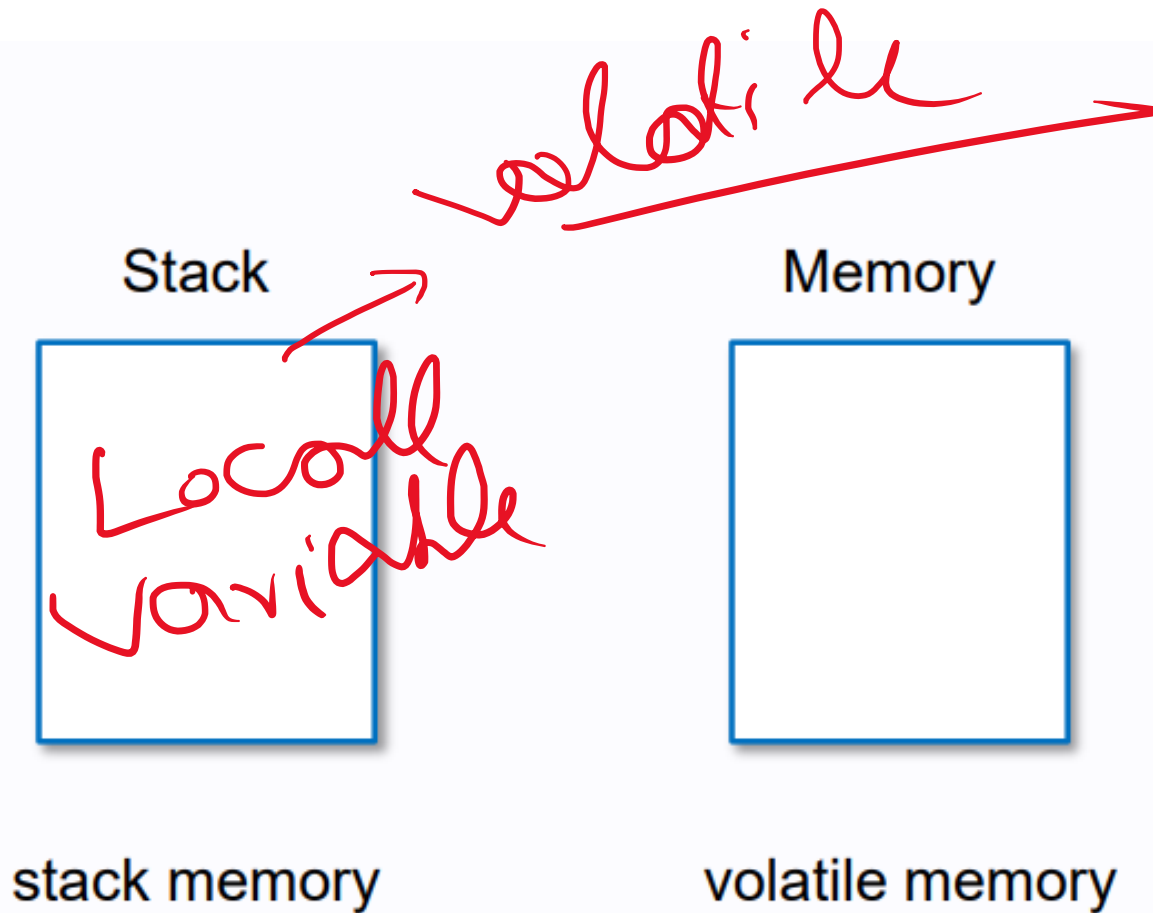- Declared inside functions and are kept on the stack , not on storage.

- Don't cost gas.

Question Time

EVM

local variable

reference data type

state variable

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

volatile

personal store

Stack

Local variable

Memory

(Account) storage

State variable

stack memory

volatile memory

persistent memory
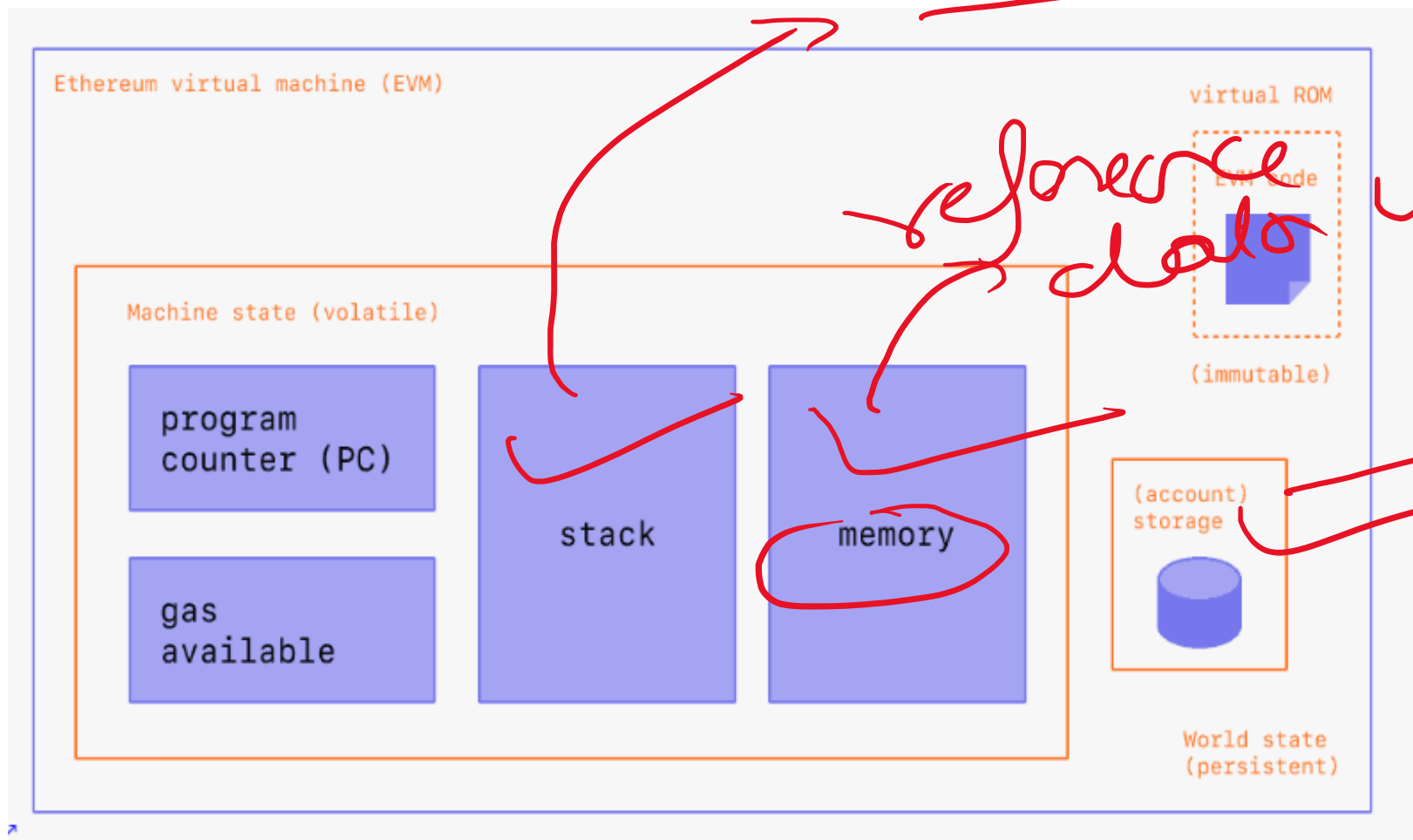
Important

- State variables are always in storage.

- Function arguments are always in memory.

- Local variables of value type (i.e. neither array, nor struct nor mapping) are stored in the stack.

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	✖	✖

```solidity
// SPDX-License-
Identifier: GPL-3.0

pragma solidity
>=0.7.0 <0.9.0;


contract demo{
  uint public num;



   uint abc;
 function setter()
public { //we are
writing to the state
variable
    uint check=abc;
    num=2;
 }
 function getter()
public view
returns(uint){//read
ing from the state
```

Quiz Time 1 – 10 min

Question Time

Constructor

- 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

Signed and Unsigned integers can be of various sizes.

int8 to int256

uint8 to uint256

int alias to int256

uint alias to uint256

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.

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 arr1="ab";

    function returnByte() public view returns(bytes2)
    {
        return arr1;
    }
}
```

Output - 0x6162

Example - 2

Question Time

Address Data Type

- address public addr = "0xBE4024Fa7461933F930DD3CEf5D1a01363E9f284"

- The address type is a 160-bit value that does not allow any arithmetic operations.

Question Time

Conditionnels

Question Time

Require

Question Time

Modifier

```solidity
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;
    }


}
```

Question Time

Loop

```
contract demo{

    function check1() public pure{
        for(uint i=0;i<7;i++){

        }


        while(true==true){

        }


        do{

        }while(true==true);
    }

}
```

Question Time

Visibility

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

Quiz Time 2 – 10 min

Question Time

Thank You

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