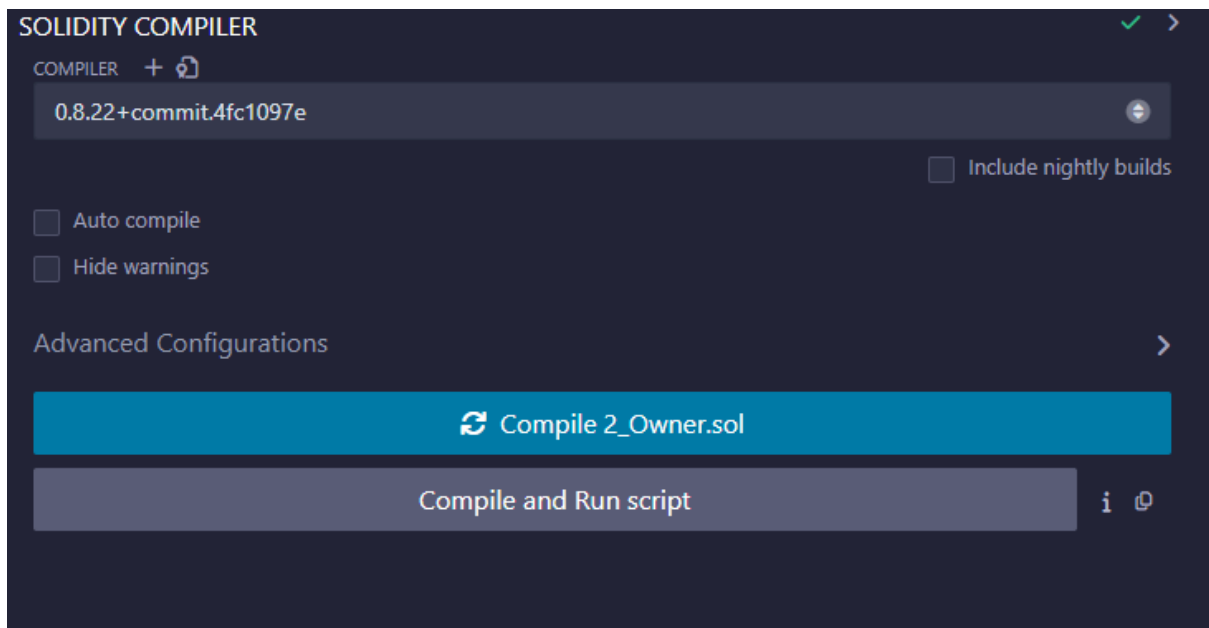
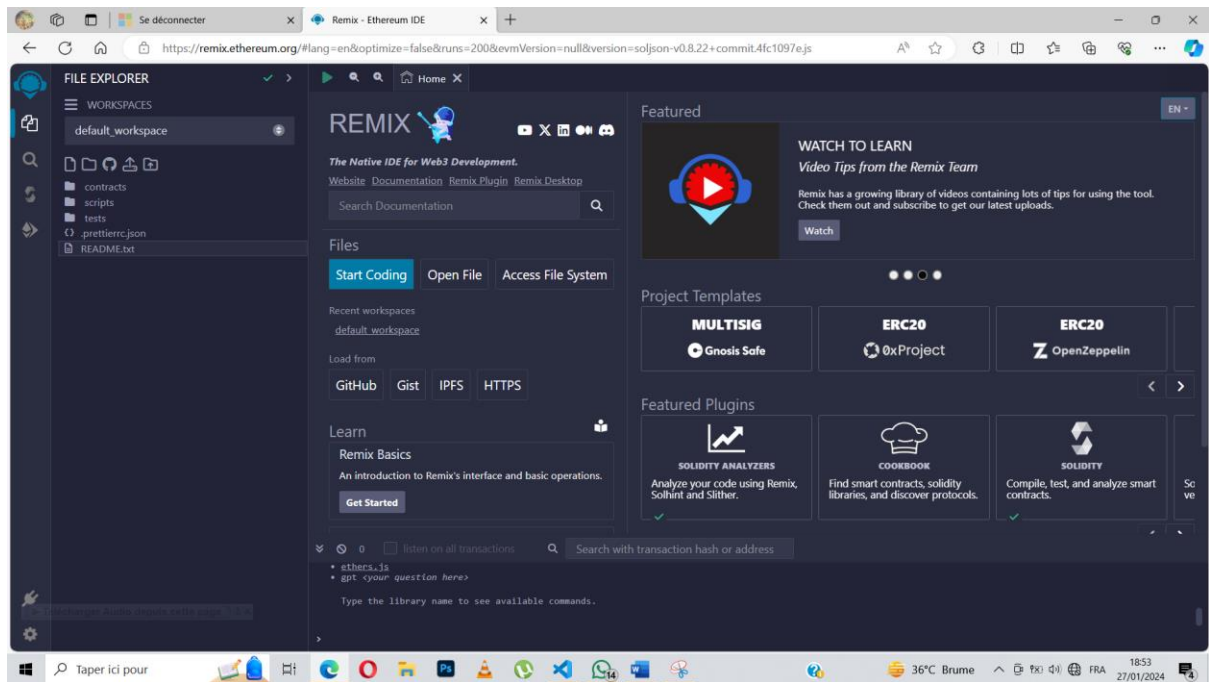


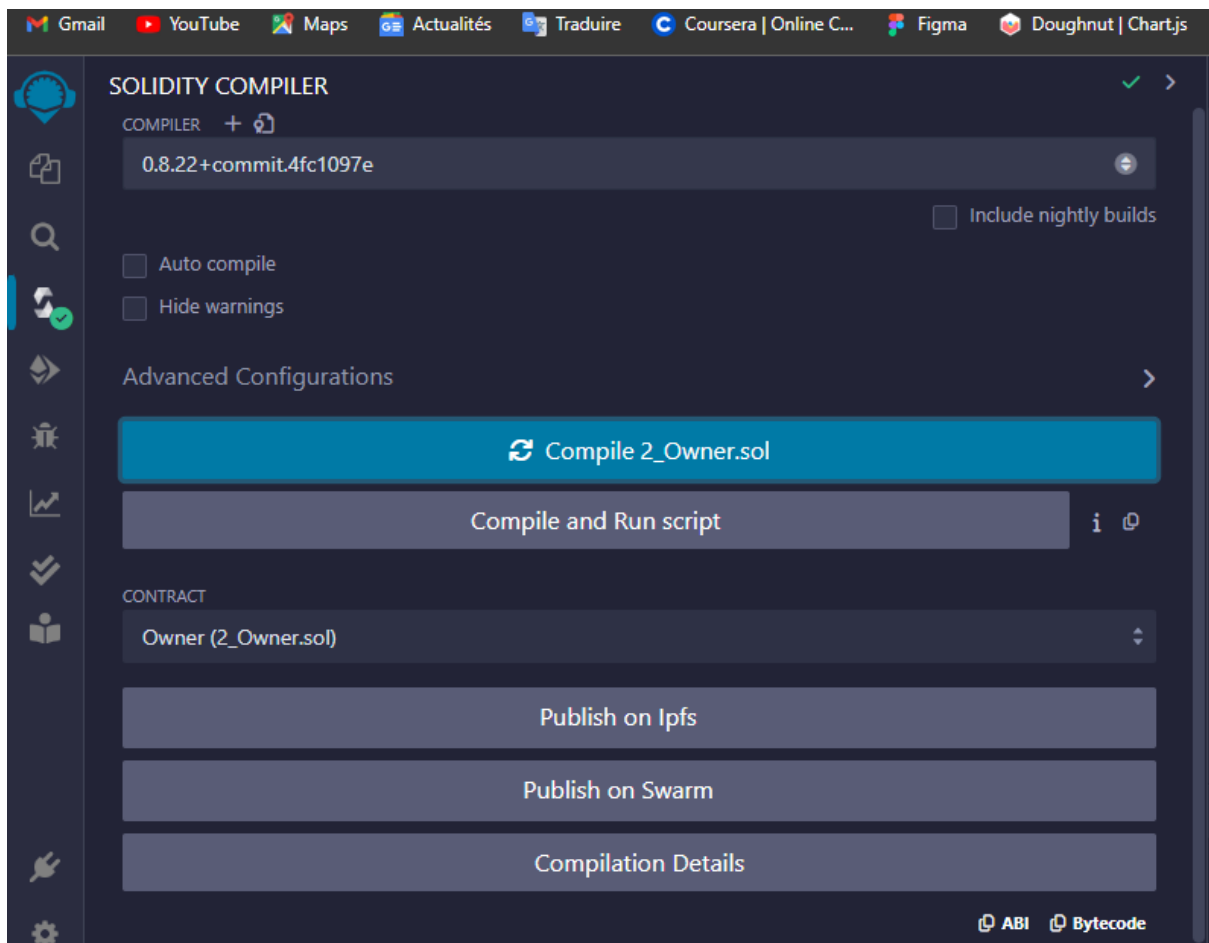


TP BLOCKCHAIN 1

N'ZI KOUADIO MARC-EZECHIEL TLR3

TP BLOCKCHAIN 1






On a de nouveaux champs qui apparaissent



Byte code :

608060405234801561000f575f80fd5b506100556040518060400160405280601b81526020017f4f77
6e657220636f6e7472616374206465706c6f7965642062793a00000000008152503361011260201b6
0201c565b335f806101000a81548173ffffffffffffffffffffffffffff021916908373ffffffffffffffffffffff
ffffffffffff1602179055505f8054906101000a900473ffffffffffffffffffffffffffff1673ffffffffffffff
ffffffffffff165f73ffffffffffffffffffffffffffff167f342827c97908e5e2f71151c08502a66d44b6f758e
3ac2f1de95f02eb95f0a73560405160405180910390a3610337565b6101b0828260405160240161012
89291906102dc565b6040516020818303038152906040527f319af333000000000000000000000000
000
000
17bfffffffffffffffffffffffffffffffffffff83818316178352505050506101b460201b60201c565b50
50565b6101d5816101d06101d860201b6101e1176101f760201b60201c565b60201c565b50565b5f6
a636f6e736f6c652e6c6f6790505f80835160208501845afa505050565b61020960201b610200178190
50919050565b61021161030a565b565b5f81519050919050565b5f828252602082019050929150505
65b5f5b8381101561024a57808201518184015260208101905061022f565b5f8484015250505050565
b5f601f19601f8301169050919050565b5f61026f82610213565b610279818561021d565b935061028
981856020860161022d565b6102928161025565b840191505092915050565b5f73ffffffffffffff
ffffffffffff82169050919050565b5f6102c68261029d565b9050919050565b6102d6816102bc565b82
525050565b5f6040820190508181035f8301526102f48185610265565b905061030360208301846102
cd565b9392505050565b7f4e487b7100
000005f52605160045260245ffd5b610396806103445f395ff3fe608060405234801561000f575f80fd5b
5060043610610034575f3560e01c8063893d20e814610038578063a6f9dae114610056575b5f80fd5b



```
    {
      "indexed": true,
      "internalType": "address",
      "name": "oldOwner",
      "type": "address"
    },
    {
      "indexed": true,
      "internalType": "address",
      "name": "newOwner",
      "type": "address"
    }
  ],
  "name": "OwnerSet",
  "type": "event"
},
{
  "inputs": [],
  "name": "getOwner",
  "outputs": [
    {
      "internalType": "address",
      "name": "",
      "type": "address"
    }
  ],
  "stateMutability": "view",
  "type": "function"
}
]
```




DEPLOY & RUN TRANSACTIONS

ENVIRONMENT 

Remix VM (Shanghai)  

VM


ACCOUNT 

0x4B2...C02db (100 ether)   


GAS LIMIT

3000000

VALUE

0 Wei 

CONTRACT



Owner - contracts/2_Owner.sol 

evm version: shanghai


Deploy

☐ Publish to IPFS



Deployed Contracts

OWNER AT 0x9EC...DDE84 (MEMORY)  


Owner contract deployed by: 0x4B209938c481177ec7E8f571ceCaE8A9e22C02db

[vm] from: 0x4B2...C02db to: Owner.(constructor)
value: 0 wei data: 0x608...60033 logs: 1 


0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84

OWNER AT 0x9EC...DDE84 (MEMORY)  


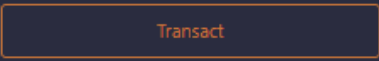
Balance: 0. ETH

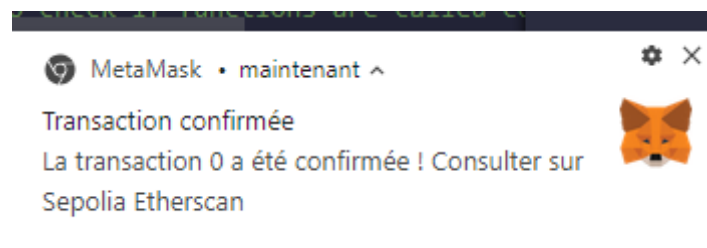
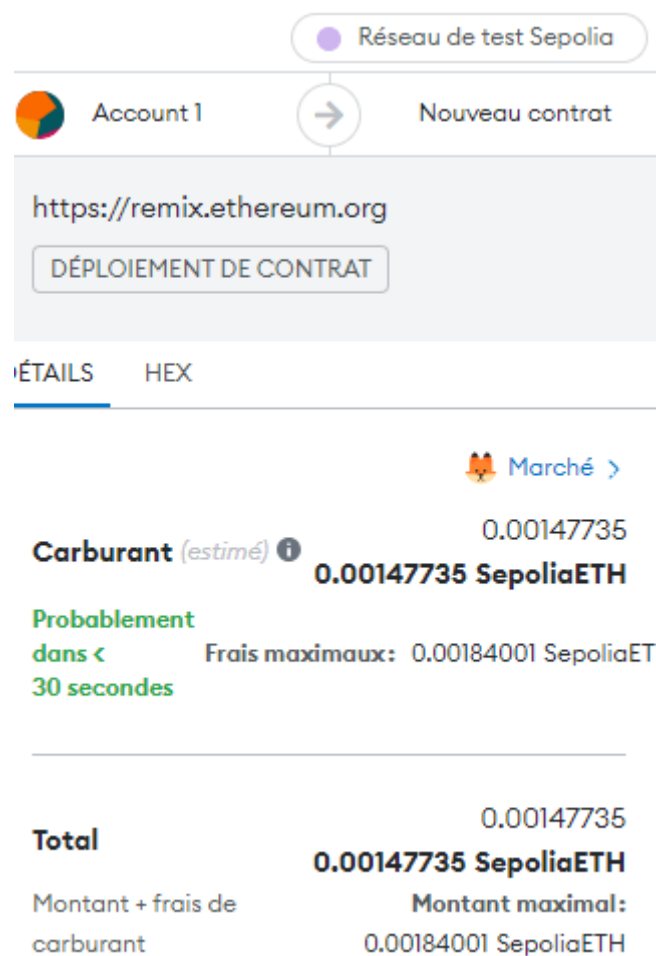
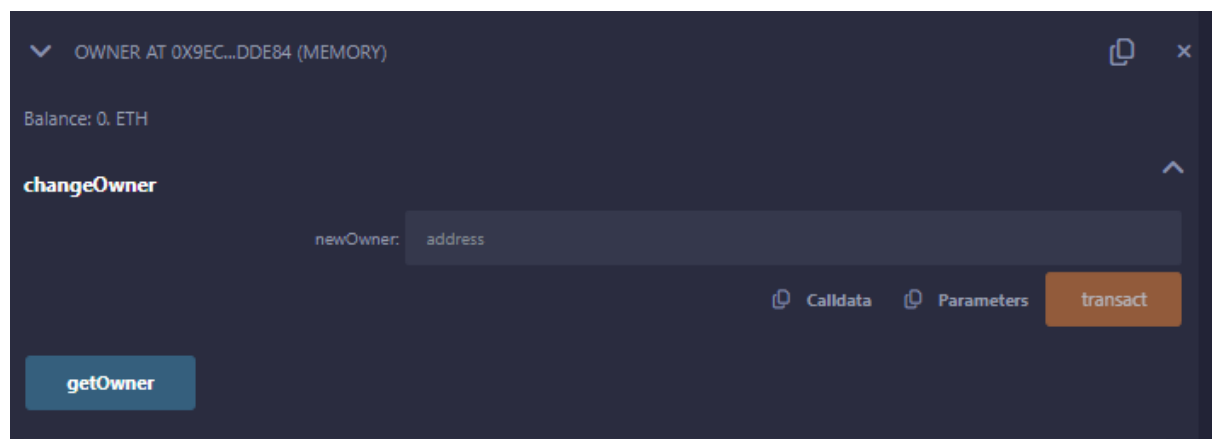
changeOwner address newOwner 

getOwner

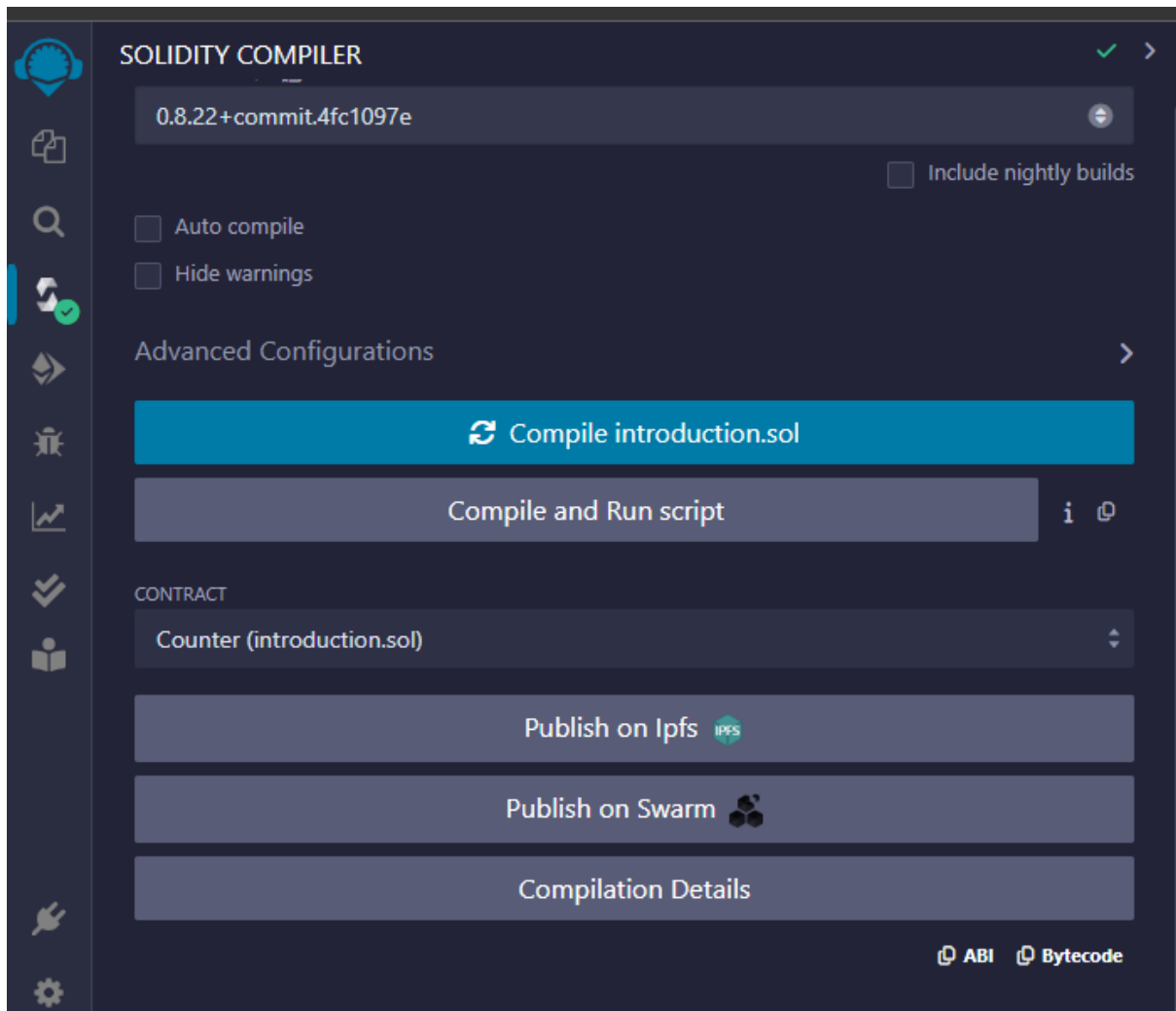
Low level interactions 

CALLDATA



TP BLOCKCHAIN 2



The image shows the Solidity Compiler interface within a development environment. On the left is a vertical sidebar with icons for Explorer, Search, Run and Debug, Test Runner, Coverage, Console, and Settings. The main panel is titled "SOLIDITY COMPILER" and features a version selector set to "0.8.22+commit.4fc1097e", a checkbox for "Include nightly builds", and options for "Auto compile" and "Hide warnings". Below these are "Advanced Configurations" and a large blue button labeled "Compile introduction.sol". A secondary button "Compile and Run script" is also present. The "CONTRACT" section shows "Counter (introduction.sol)" selected. At the bottom, there are buttons for "Publish on Ipfs" (with an IPFS icon), "Publish on Swarm" (with a Swarm icon), and "Compilation Details". The footer includes links for "ABI" and "Bytecode".

SOLIDITY COMPILER

0.8.22+commit.4fc1097e

☐ Include nightly builds

☐ Auto compile

☐ Hide warnings

Advanced Configurations

Compile introduction.sol

Compile and Run script

CONTRACT

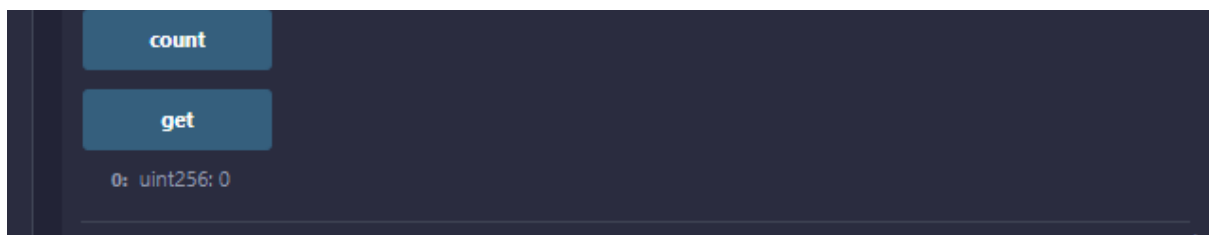
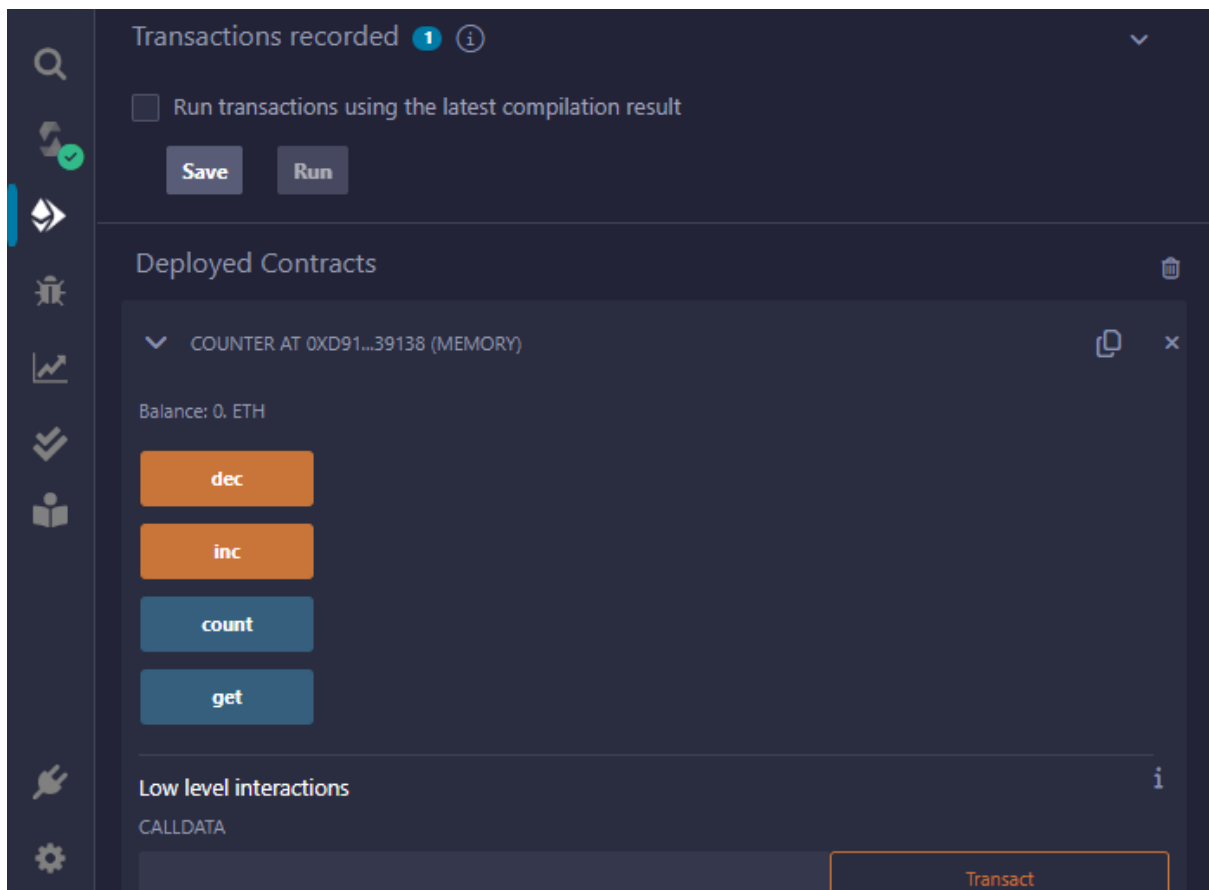
Counter (introduction.sol)

Publish on Ipfs

Publish on Swarm

Compilation Details

ABI Bytecode

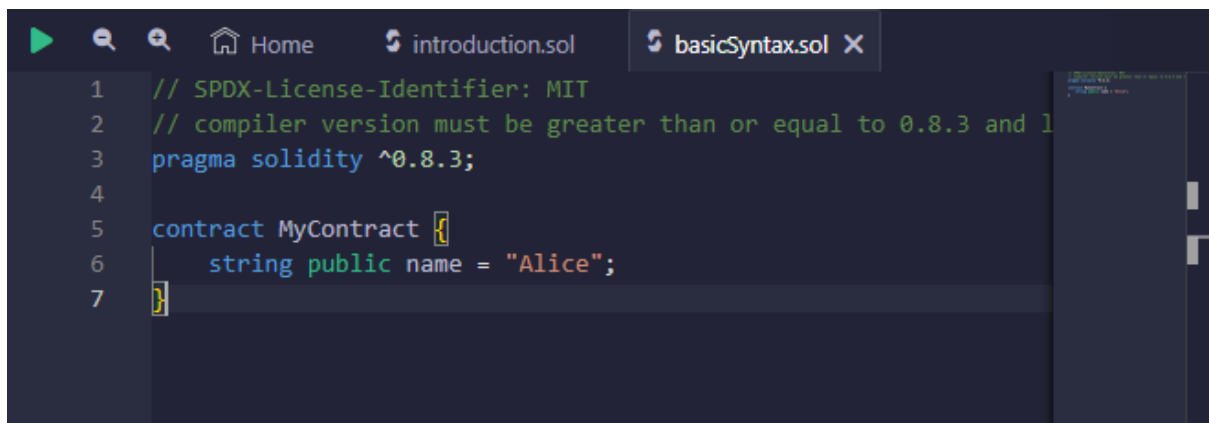


Le bytecode a été stocké dans la mémoire de la blockchain «Ethereum », dans le compte du contrat.

2. BASIC SYNTAX :




```
1 // SPDX-License-Identifier: MIT
2 // compiler version must be greater than or equal to 0.8.3 and 1
3 pragma solidity ^0.8.3;
4
5 contract HelloWorld {
6     string public greet = "Hello World!";
7 }
```



```
1 // SPDX-License-Identifier: MIT
2 // compiler version must be greater than or equal to 0.8.3 and 1
3 pragma solidity ^0.8.3;
4
5 contract MyContract {
6     string public name = "Alice";
7 }
```

we will look into them in the following sections.

 **Assignment**


1. Delete the HelloWorld contract and its content.
2. Create a new contract named "MyContract".
3. The contract should have a public state variable called "name" of the type string.
4. Assign the value "Alice" to your new variable.

Check Answer

Show answer

Next

Well done! No errors.

 LearnEth is modify

3. PRIMITIVE DATA TYPES :

```
3
4 contract Primitives {
5     bool public boo = true;
6
7     /*
8     uint stands for unsigned integer, meaning non negative integers
9     different sizes are available
10     uint8   ranges from 0 to 2 ** 8 - 1
11     uint16  ranges from 0 to 2 ** 16 - 1
12     ...
13     uint256 ranges from 0 to 2 ** 256 - 1
14     */
15     uint8 public u8 = 1;
16     uint public u256 = 456;
17     uint public u = 123; // uint is an alias for uint256
18
19     /*
20     Negative numbers are allowed for int types.
21     Like uint, different ranges are available from int8 to int256
22     */
23     int8 public i8 = -1;
24     int public i256 = 456;
25     int public i = -123; // int is same as int256
26
27     address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;
28
29     // Default values
30     // Unassigned variables have a default value
```

LEARNETH

< 3/19 >

Later in the course, we will look at data structures like Mappings, Arrays, Enums, and Structs.

Watch a video tutorial on Primitive Data Types.

★ Assignment

1. Create a new variable `newAddr` that is a `public address` and give it a value that is not the same as the available variable `addr`.
2. Create a `public` variable called `neg` that is a negative number, decide upon the type.
3. Create a new variable, `newU` that has the smallest `uint` size type and the smallest `uint` value and is `public`.

Tip: Look at the other address in the contract or search the internet for an Ethereum address.

Check Answer

Show answer

Next

Well done! No errors.

```
13
14
15     /*
16     uint8 public u8 = 1;
17     uint public u256 = 456;
18     uint public u = 123; // uint is an alias for uint256
19     uint public newU;
20
21     /*
22     Negative numbers are allowed for int types.
23     Like uint, different ranges are available from int8 to int256
24     */
25     int8 public i8 = -1;
26     int public i256 = 456;
27     int public i = -123; // int is same as int256
28     int public neg = -256;
29
30     address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;
31     address public newAddr = 0x777788889999AaAAbBbCcCddDdeeeEfffFcCcC;
32
33     // Default values
34     // Unassigned variables have a default value
35     bool public defaultBool; // false
36     uint public defaultUint; // 0
37     int public defaultInt; // 0
38     address public defaultAddr; // 0x0000000000000000000000000000000000000000
39 }
```

0

☐ listen on all transactions

4. LES VARIABLES :

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Variables {
5     // State variables are stored on the blockchain.
6     string public text = "Hello";
7     uint public num = 123;
8
9
10    function doSomething() public { 203 gas
11        // Local variables are not saved to the blockchain.
12        uint i = 456;
13
14        // Here are some global variables
15        uint timestamp = block.timestamp; // Current block timestamp
16        address sender = msg.sender; // address of the caller
17    }
18 }
```

LEARNETH

< 4/19 >

In this example, we use `block.timestamp` (line 14) to get a Unix timestamp of when the current block was generated and `msg.sender` (line 15) to get the caller of the contract function's address.

A list of all Global Variables is available in the [Solidity documentation](#).

Watch video tutorials on [State Variables](#), [Local Variables](#), and [Global Variables](#).

★ **Assignment**

1. Create a new public state variable called `blockNumber`.
2. Inside the function `doSomething()`, assign the value of the current block number to the state variable `blockNumber`.

Tip: Look into the global variables section of the Solidity documentation to find out how to read the current block number.

Check Answer

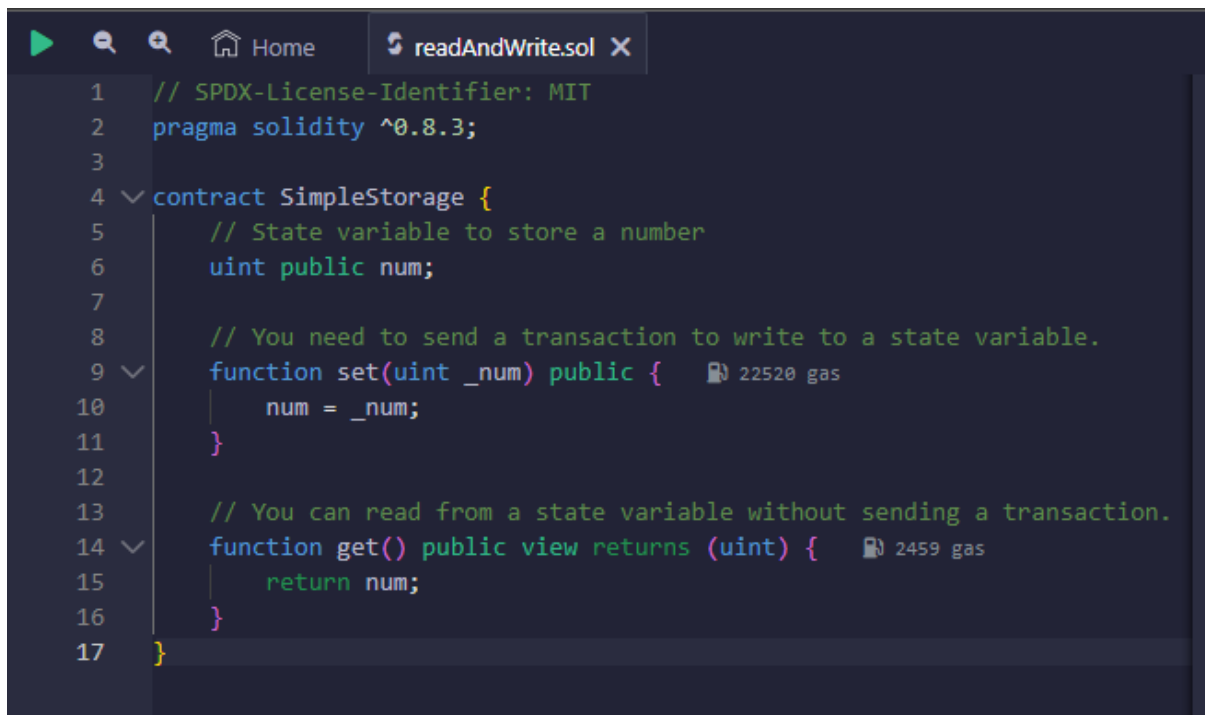
Show answer

Next

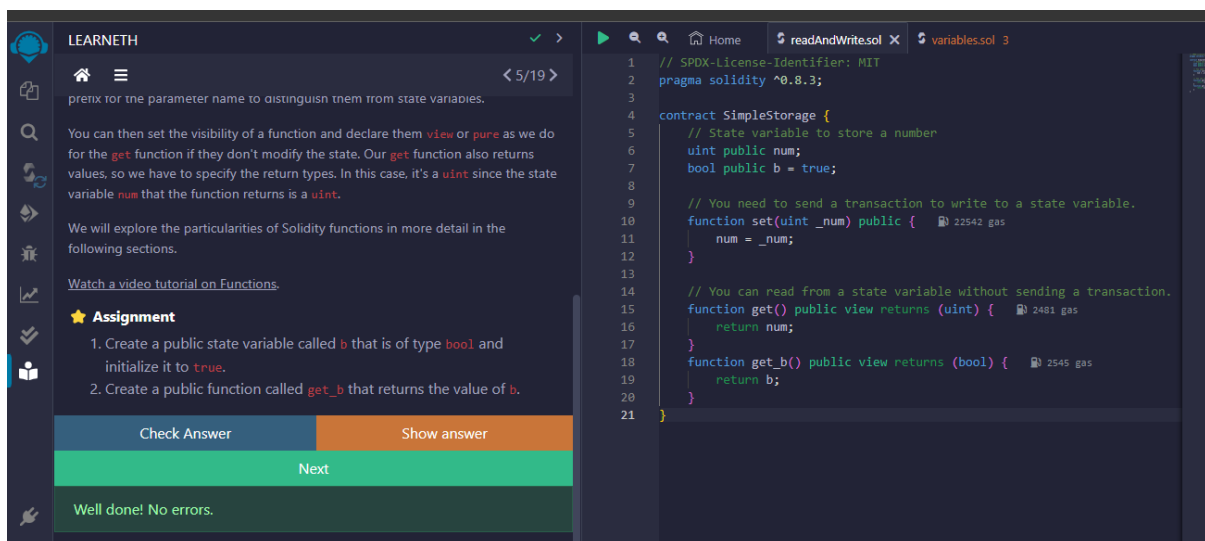
Well done! No errors.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Variables {
5     // State variables are stored on the blockchain.
6     string public text = "Hello";
7     uint public num = 123;
8     uint public blockNumber;
9
10    function doSomething() public { 22338 gas
11        // Local variables are not saved to the blockchain.
12        uint i = 456;
13
14        // Here are some global variables
15        uint timestamp = block.timestamp; // Current block timestamp
16        address sender = msg.sender; // address of the caller
17        blockNumber = block.number;
18    }
19 }
```

5.1 FUNCTIONS - READING AND WRITING TO A STATE VARIABLE



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract SimpleStorage {
5     // State variable to store a number
6     uint public num;
7
8     // You need to send a transaction to write to a state variable.
9     function set(uint _num) public { 22520 gas
10         num = _num;
11     }
12
13     // You can read from a state variable without sending a transaction.
14     function get() public view returns (uint) { 2459 gas
15         return num;
16     }
17 }
```



LEARNETH

premix for the parameter name to distinguish them from state variables.

You can then set the visibility of a function and declare them **view** or **pure** as we do for the **get** function if they don't modify the state. Our **get** function also returns values, so we have to specify the return types. In this case, it's a **uint** since the state variable **num** that the function returns is a **uint**.

We will explore the particularities of Solidity functions in more detail in the following sections.

[Watch a video tutorial on Functions.](#)

★ **Assignment**

1. Create a public state variable called **b** that is of type **bool** and initialize it to **true**.
2. Create a public function called **get_b** that returns the value of **b**.

Check Answer Show answer

Next

Well done! No errors.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract SimpleStorage {
5     // State variable to store a number
6     uint public num;
7     bool public b = true;
8
9     // You need to send a transaction to write to a state variable.
10    function set(uint _num) public { 22542 gas
11        num = _num;
12    }
13
14    // You can read from a state variable without sending a transaction.
15    function get() public view returns (uint) { 2481 gas
16        return num;
17    }
18    function get_b() public view returns (bool) { 2545 gas
19        return b;
20    }
21 }
```

5.2 FUNCTIONS - VIEW AND PURE

5.3 FUNCTIONS - MODIFIERS AND CONSTRUCTORS

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract FunctionModifier {
    // We will use these variables to demonstrate how to use
    // modifiers.
    address public owner;
    uint public x = 10;
    bool public locked;

    constructor() {
        // Set the transaction sender as the owner of the contract.
        owner = msg.sender;
    }

    // Modifier to check that the caller is the owner of
    // the contract.
    modifier onlyOwner() {
        require(msg.sender == owner, "Not owner");
        // Underscore is a special character only used inside
        // a function modifier and it tells Solidity to
        // execute the rest of the code.
        _;
    }

    // Modifiers can take inputs. This modifier checks that the
    // address passed in is not the zero address.
    modifier validAddress(address addr) {
```

contract (line 11) sets the initial value of the owner variable upon the creation of the contract.

[Watch a video tutorial on Function Modifiers.](#)

★ **Assignment**

1. Create a new function, `increaseX` in the contract. The function should take an input parameter of type `uint` and increase the value of the variable `x` by the value of the input parameter.
2. Make sure that `x` can only be increased.
3. The body of the function `increaseX` should be empty.

Tip: Use modifiers.

Check AnswerShow answer

Next

Well done! No errors.

```
32 function changeOwner(address _newOwner) public onlyOwner validAddress(_newOwner) {
33     owner = _newOwner;
34 }
35
36
37 modifier biggerThan0(uint y) {
38     require(y > 0, "Not bigger than x");
39     _;
40 }
41
42 modifier increaseXbyY(uint y) {
43     _;
44     x = x + y;
45 }
46
47 function increaseX(uint y) public onlyOwner biggerThan0(y) increaseXbyY(y) {
48
49 }
50
51 // Modifiers can be called before and / or after a function.
52 // This modifier prevents a function from being called while
53 // it is still executing.
```

LearnEthereum is modifying .learneth/solidity Beginner Course/5.3 Functions - Modifiers and Constructors/modifiersAndConstructors_test.sol

5.4 FUNCTIONS - INPUTS AND OUTPUTS

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Function {
5     // Functions can return multiple values.
6     function returnMany() infinite gas
7         public
8         pure
9         returns (
10             uint,
11             bool,
12             uint
13         )
14     {
15         return (1, true, 2);
16     }
17
18     // Return values can be named.
19     function named() infinite gas
20         public
21         pure
22         returns (
23             uint x,
24             bool b,
25             uint y
26         )
27     {
28         return (1, true, 2);
29     }
30 }
```

Arrays can also be used as return parameters as shown in the function `arrayOutput` (line 76).

You have to be cautious with arrays of arbitrary size because of their gas consumption. While a function using very large arrays as inputs might fail when the gas costs are too high, a function using a smaller array might still be able to execute.

[Watch a video tutorial on Function Outputs.](#)

★ **Assignment**

Create a new function called `returnTwo` that returns the values `-2` and `true` without using a return statement.

Check AnswerShow answer

Next

Well done! No errors.

```
74 uint[] public arr;
75
76 function arrayOutput() public view returns (uint[] memory) {
77     return arr;
78 }
79
80 function returnTwo() 479 gas
81     public
82     pure
83     returns (
84         int i,
85         bool j
86     )
87 {
88     i = -2;
89     j = true;
90 }
```

LearnEth is modifying .learneth/Solidity Beginner Course/5.4 Functions - Inputs and Outputs/inputsAndOutputs_test.sol

6. VISIBILITY

```
41 // This function will not compile since we're trying to call
42 // an external function here.
43 // function testExternalFunc() public pure returns (string memory) {
44 //     return externalFunc();
45 // }
46
47 // State variables
48 string private privateVar = "my private variable";
49 string internal internalVar = "my internal variable";
50 string public publicVar = "my public variable";
51 // State variables cannot be external so this code won't compile.
52 // string external externalVar = "my external variable";
53 }
54
55 contract Child is Base {
56     // Inherited contracts do not have access to private functions
57     // and state variables.
58     // function testPrivateFunc() public pure returns (string memory) {
59     //     return privateFunc();
60     // }
61
62     // Internal function call be called inside child contracts.
63     function testInternalFunc() public pure override returns (string memory) {
64         return internalFunc();
65     }
66 }
```

Base contract

If you compile and deploy the two contracts, you will not be able to call the functions `privateFunc` and `internalFunc` directly. You will only be able to call them via `testPrivateFunc` and `testInternalFunc`.

[Watch a video tutorial on Visibility.](#)

★ **Assignment**

Create a new function in the `Child` contract called `testInternalVar` that returns the values of all state variables from the `Base` contract that are possible to return.

Check Answer Show answer

Next

Well done! No errors.

```
59 //     return privateFunc();
60 // }
61
62 // Internal function call be called inside child contracts.
63 function testInternalFunc() public pure override returns (string memory) {
64     return internalFunc();
65 }
66 function testInternalVar() public view returns (string memory, string
67     return (internalVar, publicVar);
68 }
69 }
```

7.1 CONTROL FLOW - IF/ELSE

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract IfElse {
5     function foo(uint x) public pure returns (uint) { infinite gas
6         if (x < 10) {
7             return 0;
8         } else if (x < 20) {
9             return 1;
10        } else {
11            return 2;
12        }
13    }
14
15    function ternary(uint _x) public pure returns (uint) { infinite gas
16        // if (_x < 10) {
17        //     return 1;
18        // }
19        // return 2;
20
21        // shorthand way to write if / else statement
22        return _x < 10 ? 1 : 2;
23    }
24 }
```

• The function returns **true** if the argument is even, and **false** if the argument is odd.

• Use a ternary operator to return the result of the **evenCheck** function.

Tip: The modulo (%) operator produces the remainder of an integer division.

Check Answer

Show answer

Next

Well done! No errors.

```
14
15 function ternary(uint _x) public pure returns (uint) { infinite gas
16     // if (_x < 10) {
17     //     return 1;
18     // }
19     // return 2;
20
21     // shorthand way to write if / else statement
22     return _x < 10 ? 1 : 2;
23 }
24 function evenCheck(uint z) public pure returns (bool) { infinite gas
25     return z%2 == 0 ? true : false;
26 }
```

LearnEth is modifying Jearneth/Solidity Beginner Course/7.1 Control Flow - If/Else/IfElse_test.sol

7.2 CONTROL FLOW - LOOPS

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Loop {
5     function loop() public {  infinite gas
6         // for loop
7         for (uint i = 0; i < 10; i++) {
8             if (i == 3) {
9                 // Skip to next iteration with continue
10                continue;
11            }
12            if (i == 5) {
13                // Exit loop with break
14                break;
15            }
16        }
17
18        // while loop
19        uint j;
20        while (j < 10) {
21            j++;
22        }
23    }
24 }
25
```

LEARNETH

< 11/19 >

The **continue** statement is used to skip the remaining code block and start the next iteration of the loop. In this contract, the **continue** statement (line 10) will prevent the second if statement (line 12) from being executed.

break

The **break** statement is used to exit a loop. In this contract, the **break** statement (line 14) will cause the for loop to be terminated after the sixth iteration.

[Watch a video tutorial on Loop statements.](#)

★ **Assignment**

1. Create a public **uint** state variable called **count** in the **Loop** contract.
2. At the end of the for loop, increment the count variable by 1.
3. Try to get the count variable to be equal to 9, but make sure you don't edit the **break** statement.

Check Answer Show answer

Next

Well done! No errors.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Loop {
5     uint public count;
6     function loop() public { 
7         // for loop
8         for (uint i = 0; i < 10; i++) {
9             if (i == 5) {
10                // Skip to next iteration with continue
11                continue;
12            }
13            if (i == 5) {
14                // Exit loop with break
15                break;
16            }
17            count ++;
18        }
19
20        // while loop
21        uint j;
22        while (j < 10) {
23            j++;
24        }
25    }
26 }
```

LearnEth is modifying .learneth/Solidity Beginner Course/7.2 Control Flow - Loops/loops_test.sol

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Array {
5     // Several ways to initialize an array
6     uint[] public arr;
7     uint[] public arr2 = [1, 2, 3];
8     // Fixed sized array, all elements initialize to 0
9     uint[10] public myFixedSizeArr;
10
11     function get(uint i) public view returns (uint) {    ⚡ infinite gas
12         return arr[i];
13     }
14
15     // Solidity can return the entire array.
16     // But this function should be avoided for
17     // arrays that can grow indefinitely in length.
18     function getArr() public view returns (uint[] memory) {    ⚡ infinite gas
19         return arr;
20     }
21
22     function push(uint i) public {    ⚡ 46829 gas
23         // Append to array
24         // This will increase the array length by 1.
25         arr.push(i);
26     }
27
28     function pop() public {    ⚡ 29467 gas

```

LEARNETH

same. This will create a gap in our array. If the order of the array is not important, then we can move the last element of the array to the place of the deleted element (line 46), or use a mapping. A mapping might be a better choice if we plan to remove elements in our data structure.

Array length

Using the length member, we can read the number of elements that are stored in an array (line 35).

Watch a video tutorial on Arrays.

★ **Assignment**

1. Initialize a public fixed-sized array called `arr3` with the values 0, 1, 2. Make the size as small as possible.
2. Change the `getArr()` function to return the value of `arr3`.

Check Answer

Show answer

Next

Well done! No errors.

arrays.sol

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Array {
5     // Several ways to initialize an array
6     uint[] public arr;
7     uint[] public arr2 = [1, 2, 3];
8     // Fixed sized array, all elements initialize to 0
9     uint[10] public myFixedSizeArr;
10    uint[3] public arr3 = [0, 1, 2];
11
12    function get(uint i) public view returns (uint) {    ⚡ infinite gas
13        return arr[i];
14    }
15
16
17    // Solidity can return the entire array.
18    // But this function should be avoided for
19    // arrays that can grow indefinitely in length.
20    function getArr() public view returns (uint[3] memory) {    ⚡ infinite gas
21        return arr3;
22    }
23
24    function push(uint i) public {    ⚡ 46829 gas
25        // Append to array
26        // This will increase the array length by 1.

```

LearnEthereum is modifying Learneth/Solidity Beginner Course/8.1 Data Structures - Arrays/arrays_test.sol

8.2 DATA STRUCTURES - MAPPINGS

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Mapping {
5     // Mapping from address to uint
6     mapping(address => uint) public myMap;
7
8     function get(address _addr) public view returns (uint) { 2885 gas
9         // Mapping always returns a value.
10        // If the value was never set, it will return the default value.
11        return myMap[_addr];
12    }
13
14    function set(address _addr, uint _i) public { 22854 gas
15        // Update the value at this address
16        myMap[_addr] = _i;
17    }
18
19    function remove(address _addr) public { 5554 gas
20        // Reset the value to the default value.
21        delete myMap[_addr];
22    }
23 }
24
25 contract NestedMapping {
26     // Nested mapping (mapping from address to another mapping)
27     mapping(address => mapping(uint => bool)) public nested;
28 }
```

LEARNETH

< 13/19 >

Removing values

We can use the delete operator to delete a value associated with a key, which will set it to the default value of 0. As we have seen in the arrays section.

[Watch a video tutorial on Mappings.](#)

★ **Assignment**

1. Create a public mapping `balances` that associates the key type `address` with the value type `uint`.
2. Change the functions `get` and `remove` to work with the mapping `balances`.
3. Change the function `set` to create a new entry to the `balances` mapping, where the key is the address of the parameter and the value is the balance associated with the address of the parameter.

Check Answer

Show answer

Next

Well done! No errors.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Mapping {
5     // Mapping from address to uint
6     mapping(address => uint) public balances;
7
8     function get(address _addr) public view returns (uint) { 2885 gas
9         // Mapping always returns a value.
10        // If the value was never set, it will return the default value.
11        return balances[_addr];
12    }
13
14    function set(address _addr) public { 25265 gas
15        // Update the value at this address
16        balances[_addr] = _addr.balance;
17    }
18
19    function remove(address _addr) public { 5576 gas
20        // Reset the value to the default value.
21        delete balances[_addr];
22    }
23 }
24
25 contract NestedMapping {
26     // Nested mapping (mapping from address to another mapping)
27     mapping(address => mapping(uint => bool)) public nested;
28 }
```

8.3 DATA STRUCTURES - STRUCTS

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Todos {
    struct Todo {
        string text;
        bool completed;
    }

    // An array of 'Todo' structs
    Todo[] public todos;

    function create(string memory _text) public {
        // 3 ways to initialize a struct
        // - calling it like a function
        todos.push(Todo(_text, false));

        // key value mapping
        todos.push(Todo({text: _text, completed: false}));

        // initialize an empty struct and then update it
        Todo memory todo;
        todo.text = _text;
        // todo.completed initialized to false

        todos.push(todo);
    }
}
```

(lines 39 and 45).

Watch a video tutorial on Structs.

★ **Assignment**

Create a function `remove` that takes a `uint` as a parameter and deletes a struct member with the given index in the `todos` mapping.

Check Answer

Show answer

Next

Well done! No errors.

```
44     Todo storage todo = todos[_index];
45     todo.completed = !todo.completed;
46 }
47
48 function remove(uint _index) public {
49     delete todos[_index];
50 }
51 }
```

LearnEth is modifying Learneth/Solidity Beginner Course/8.3 Data Structures - Structs/structs_test.sol

8.4 DATA STRUCTURES - ENUMS

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Enum {
5     // Enum representing shipping status
6     enum Status {
7         Pending,
8         Shipped,
9         Accepted,
10        Rejected,
11        Canceled
12    }
13
14    // Default value is the first element listed in
15    // definition of the type, in this case "Pending"
16    Status public status;
17
18    // Returns uint
19    // Pending - 0
20    // Shipped - 1
21    // Accepted - 2
22    // Rejected - 3
23    // Canceled - 4
24    function get() public view returns (Status) { 2590 gas
25        return status;
26    }
27
28    // Update status by passing uint into input
```

the enum member (line 30). Shipped would be 1 in this example. Another way to update the value is using the dot operator by providing the name of the enum and its member (line 35).

Removing an enum value

We can use the delete operator to delete the enum value of the variable, which means as for arrays and mappings, to set the default value to 0.

[Watch a video tutorial on Enums.](#)

★ Assignment

1. Define an enum type called `Size` with the members `S`, `M`, and `L`.
2. Initialize the variable `sizes` of the enum type `Size`.
3. Create a getter function `getSize()` that returns the value of the variable `sizes`.

Check Answer

Show answer

Next

Well done! No errors.

```
13
14
15     enum Size {
16         S,
17         M,
18         L
19     }
20
21    // Default value is the first element listed in
22    // definition of the type, in this case "Pending"
23    Status public status;
24    Size public sizes;
25
26    // Returns uint
27    // Pending - 0
28    // Shipped - 1
29    // Accepted - 2
30    // Rejected - 3
31    // Canceled - 4
32    function get() public view returns (Status) { 2613 gas
33        return status;
34    }
35    function getSize() public view returns (Size) { 2640 gas
36        return sizes;
37    }
```

LearnEth is modifying learneth/Solidity Beginner Course/8.4 Data Structures - Enums/enums_test.sol

9. DATA LOCATIONS

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract DataLocations {
5     uint[] public arr;
6     mapping(uint => address) map;
7     struct MyStruct {
8         uint foo;
9     }
10    mapping(uint => MyStruct) myStructs;
11
12    function f() public { 381 gas
13        // call _f with state variables
14        _f(arr, map, myStructs[1]);
15
16        // get a struct from a mapping
17        MyStruct storage myStruct = myStructs[1];
18        // create a struct in memory
19        MyStruct memory myMemStruct = MyStruct(0);
20    }
21
22    function _f(  undefined gas
23        uint[] storage _arr,
24        mapping(uint => address) storage _map,
25        MyStruct storage _myStruct
26    ) internal {
27        // do something with storage variables
28    }
```

LEARNETH

gas possible.

★ Assignment

1. Change the value of the `myStruct` member `foo`, inside the function `f`, to 4.
2. Create a new struct `myMemStruct2` with the data location `memory` inside the function `f` and assign it the value of `myMemStruct`. Change the value of the `myMemStruct2` member `foo` to 1.
3. Create a new struct `myMemStruct3` with the data location `memory` inside the function `f` and assign it the value of `myStruct`. Change the value of the `myMemStruct3` member `foo` to 3.
4. Let the function `f` return `myStruct`, `myMemStruct2`, and `myMemStruct3`.

Tip: Make sure to create the correct return types for the function `f`.

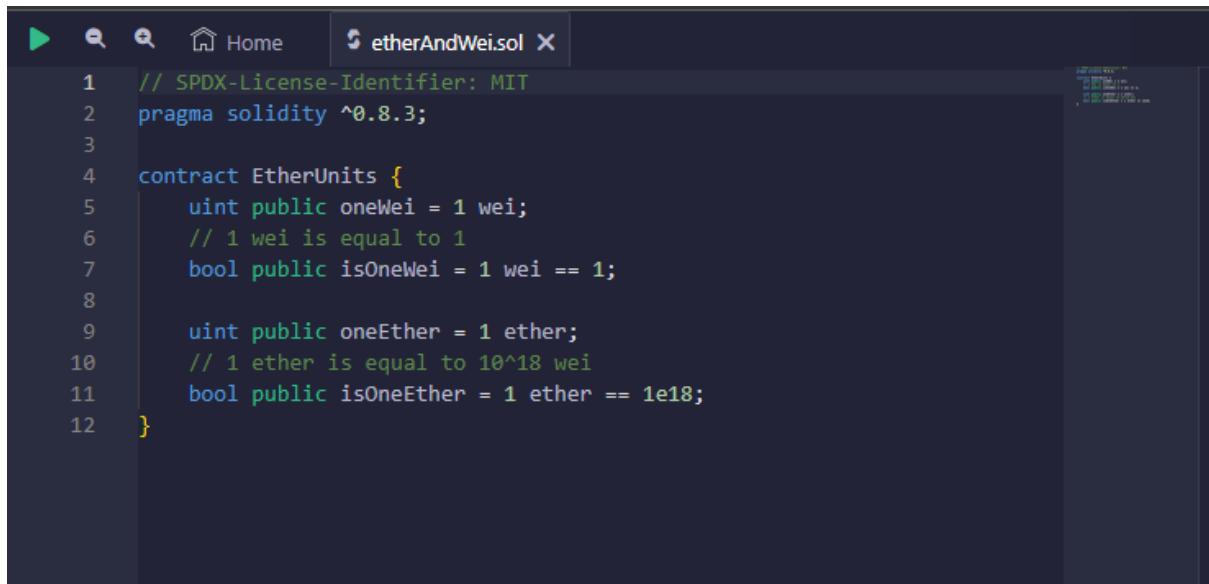
Check AnswerShow answer

Next

Well done! No errors.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract DataLocations {
5     uint[] public arr;
6     mapping(uint => address) map;
7     struct MyStruct {
8         uint foo;
9     }
10    mapping(uint => MyStruct) public myStructs;
11
12    function f() public returns (MyStruct memory, MyStruct memory, MyStruct memory) {
13        // call _f with state variables
14        _f(arr, map, myStructs[1]);
15        // get a struct from a mapping
16        MyStruct storage myStruct = myStructs[1];
17        myStruct.foo = 4;
18        // create a struct in memory
19        MyStruct memory myMemStruct = MyStruct(0);
20        MyStruct memory myMemStruct2 = myMemStruct;
21        myMemStruct2.foo = 1;
22
23        MyStruct memory myMemStruct3 = myStruct;
24        myMemStruct3.foo = 3;
25        return (myStruct, myMemStruct2, myMemStruct3);
26    }
27 }
```


10.1 TRANSACTIONS - ETHER AND WEI



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract EtherUnits {
5     uint public oneWei = 1 wei;
6     // 1 wei is equal to 1
7     bool public isOneWei = 1 wei == 1;
8
9     uint public oneEther = 1 ether;
10    // 1 ether is equal to 10^18 wei
11    bool public isOneEther = 1 ether == 1e18;
12 }
```



Assignment

1. Create a **public uint** called **oneGwei** and set it to 1 **gwei**.
2. Create a **public bool** called **isOneGwei** and set it to the result of a comparison operation between 1 **gwei** and 10^9 .

Tip: Look at how this is written for **gwei** and **ether** in the contract.

[Check Answer](#) [Show answer](#)

[Next](#)

Well done! No errors.

LearnEthereum is modifying Learneth/Solidity Beginner Course/10.1 Transactions - Ether and Wei/etherAndWei_test.sol