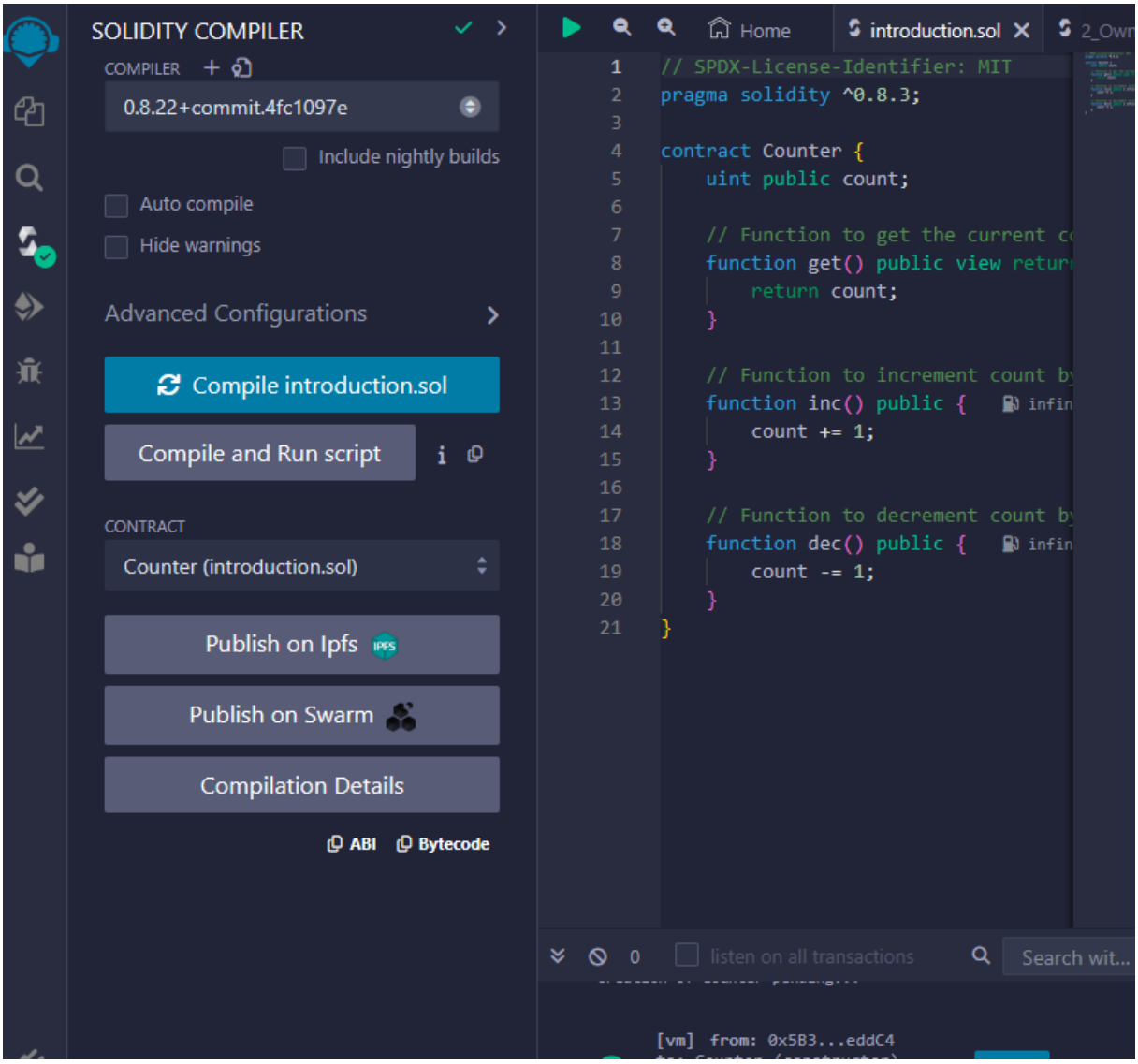
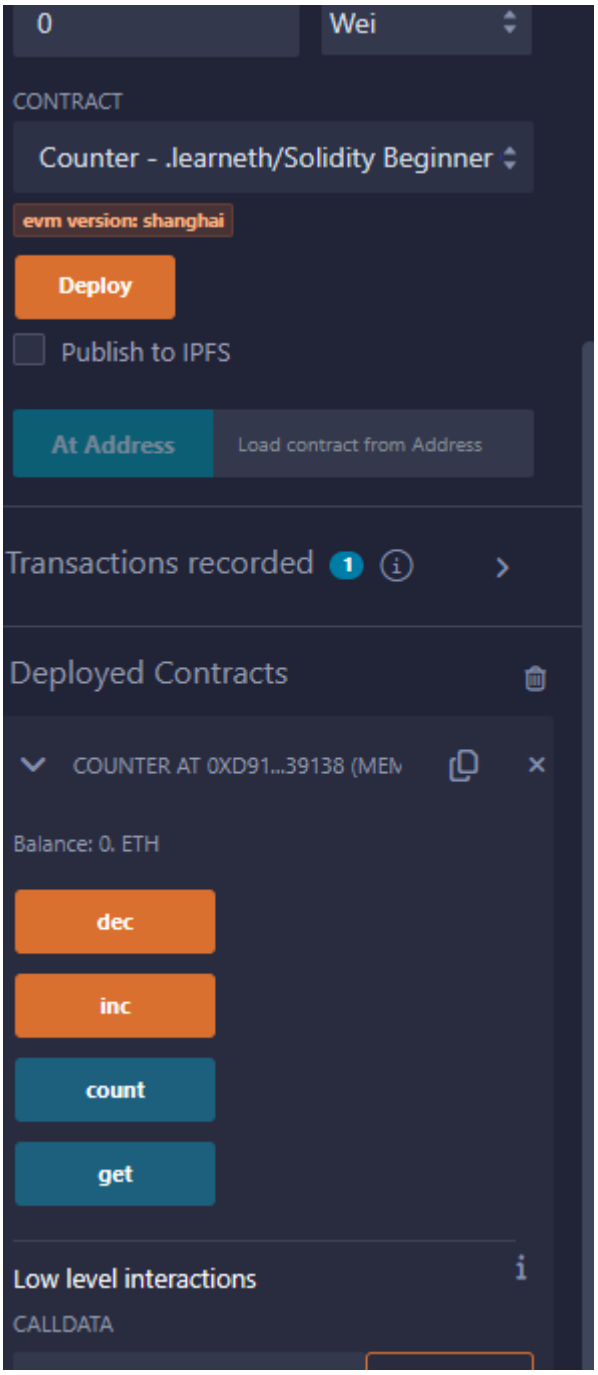
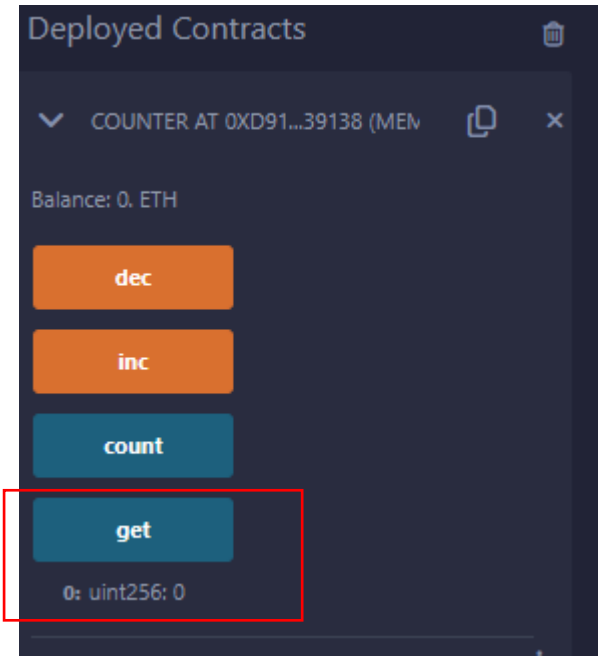
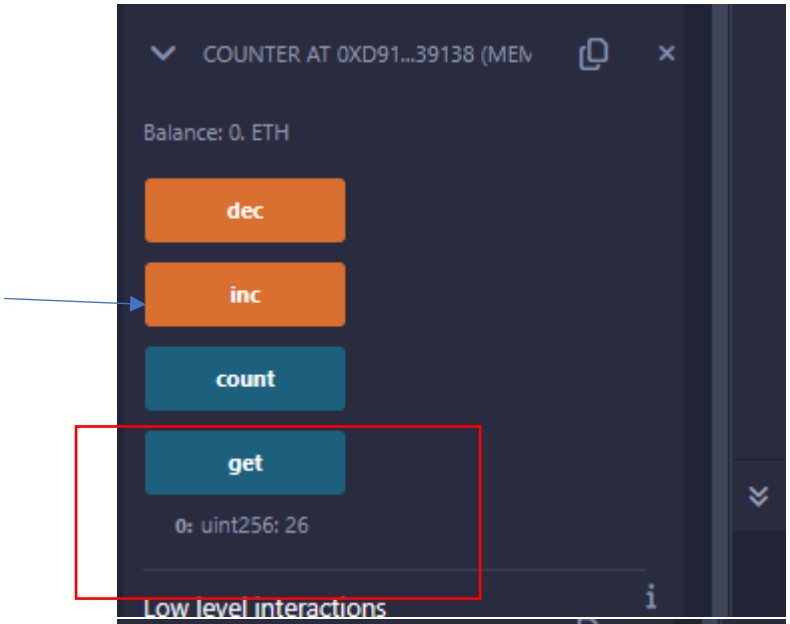
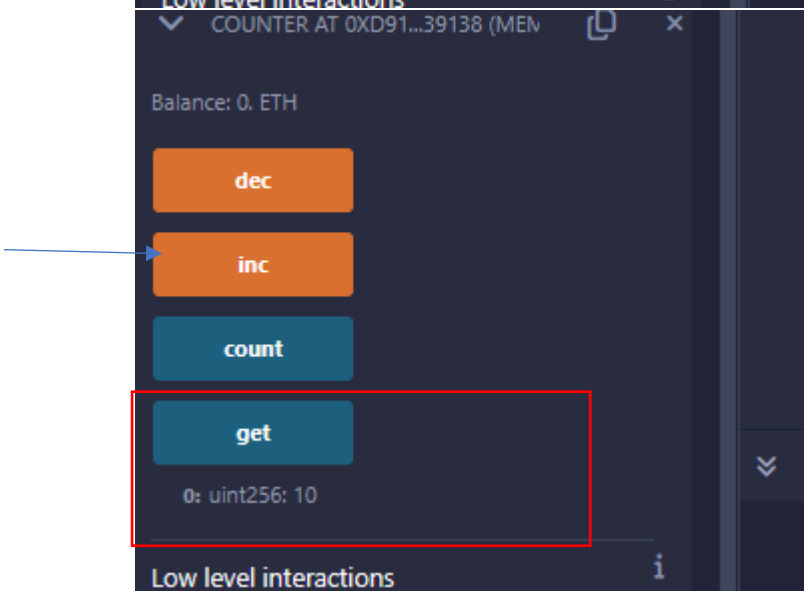
➢ 1/19


SOLIDITY COMPILER
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
Home
introduction.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;
contract Counter {
uint public count;
// Function to get the current c
function get() public view retur
return count;
}
// Function to increment count b
function inc() public {
count += 1;
}
// Function to decrement count b
function dec() public {
count -= 1;
}
}
listen on all transactions
Search wit...
[vm] from: 0x5B3...eddC4


0
Wei
CONTRACT
Counter - .learneth/Solidity Beginner
evm version: shanghai
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
COUNTER AT 0XD91...39138 (MEM
Balance: 0. ETH
dec
inc
count
get
Low level interactions
CALLDATA


Deployed Contracts
COUNTER AT 0XD91...39138 (MEM
Balance: 0. ETH
dec
inc
count
get
0: uint256: 0


COUNTER AT 0XD91...39138 (MEM
Balance: 0. ETH
dec
inc
count
get
0: uint256: 26
Low level interactions


COUNTER AT 0XD91...39138 (MEM
Balance: 0. ETH
dec
inc
count
get
0: uint256: 10
Low level interactions

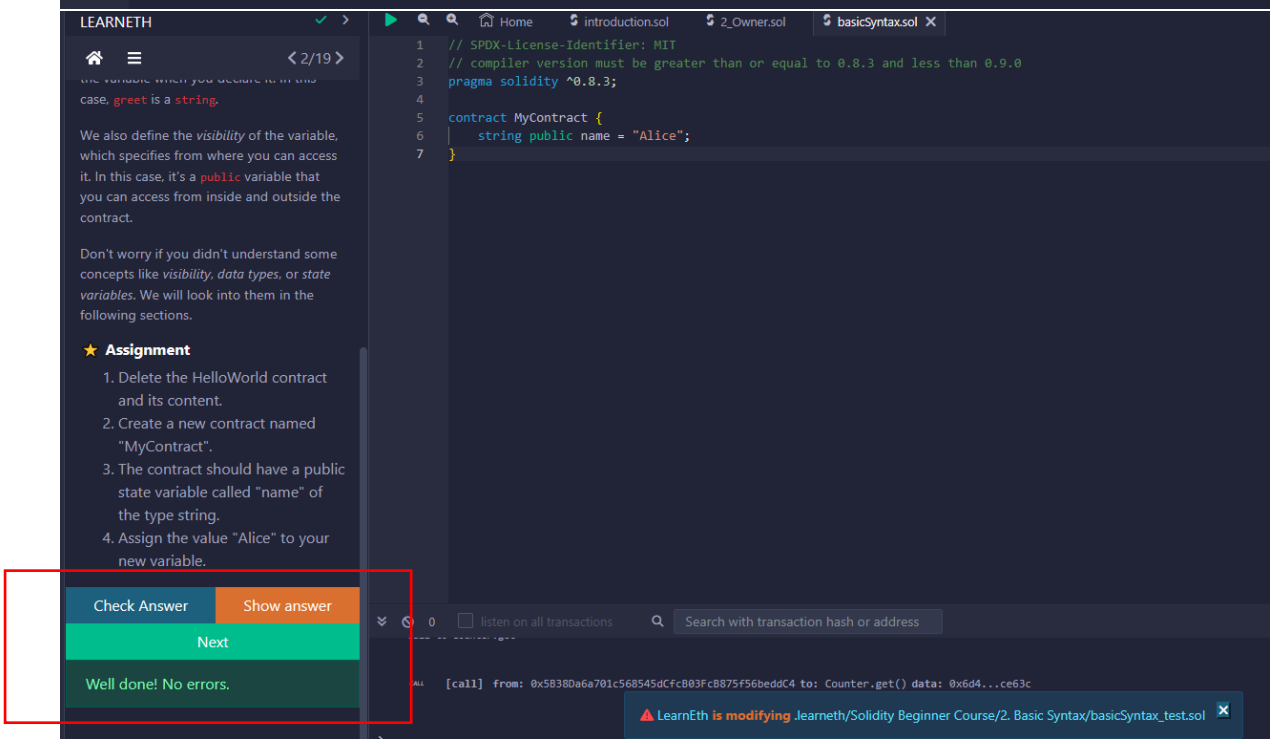
Le bytecode d'un contrat Solidity est stocké dans la mémoire de la blockchain Ethereum, plus précisément dans le compte du contrat.

- 2/19

```
contract HelloWorld {
    string public greet = "Hello World!";
}
```

```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
pragma solidity ^0.8.3;

contract MyContract {
    string public name = "Alice";
}
```



- 3/19

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.

```
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         blockNumber = block.number;
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 }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 4/19

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.

```
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     uint public newU = 0;
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     int public neg = -120;
27
28     address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;
29     address public  newAddr = 0x1111222233334444555566667777888899999aAaa;
30
31     // Default values
32     // Unassigned variables have a default value
33     bool public defaultBoo; // false
34     uint public defaultUint; // 0
35     int public defaultInt; // 0
36     address public defaultAddr; // 0x0000000000000000000000000000000000000000
37 }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c Debug

➢ 5/19

https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.22+commit.4fc1097e.js

LEARNETH 5/19

convention is to use an underscore as a prefix 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.

Home | introduction.sol | 2_Owner.sol | basicSyntax.sol | primitiveDataTypes.sol

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

contract SimpleStorage {
    // State variable to store a number
    uint public num;
    bool public b = true;

    // You need to send a transaction to write to a state variable.
    function set(uint _num) public {    22542 gas
        num = _num;
    }


    // You can read from a state variable without sending a transaction.
    function get() public view returns (uint) {    2481 gas
        return num;
    }
    function get_b() public view returns (bool) {    2545 gas
        return b;
    }
}
```

0 listen on all transactions Search with transaction hash or address

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

LearnEth is modifying learneth/Solidity Beginner Course/5.1 Functions - Reading an

- 6/19

6/19

You can declare a pure function using the keyword `pure`. In this contract, `add` (line 13) is a pure function. This function takes the parameters `i` and `j`, and returns the sum of them. It neither reads nor modifies the state variable `x`.

In Solidity development, you need to optimise your code for saving computation cost (gas cost). Declaring functions view and pure can save gas cost and make the code more readable and easier to maintain. Pure functions don't have any side effects and will always return the same result if you pass the same arguments.

Watch a video tutorial on View and Pure Functions.

Assignment

Create a function called `addToX2` that takes the parameter `y` and updates the state variable `x` with the sum of the parameter and the state variable `x`.

Check Answer | Show answer

Next

Well done! No errors.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.3;
3
4  contract ViewAndPure {
5      uint public x = 1;
6
7      // Promise not to modify the state.
8      function addToX(uint y) public view returns (uint) {    infinite gas
9          return x + y;
10     }
11     function addToX2(uint y) public {    infinite gas
12         x = x + y;
13     }
14
15     // Promise not to modify or read from the state.
16     function add(uint i, uint j) public pure returns (uint) {    infinite gas
17         return i + j;
18     }
19 }
```

0 listen on all transactions Search with transaction hash or address

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

- 7/19

deployment of the contract.

You declare a constructor using the `constructor` keyword. The constructor in this 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 Answer | Show answer

Next

Well done! No errors.

```
44          _;
45          locked = false;
46      }
47
48      function decrement(uint i) public noReentrancy {    infinite gas
49          x -= i;
50
51          if (i > 1) {
52              decrement(i - 1);
53          }
54      }
55      modifier sup(uint y) {
56          require(y > 0, "Not bigger than x");
57          _;
58
59      }
60      modifier incrementX(uint y) {
61          _;
62          x = x + y;
63      }
64      function increaseX(uint y) public onlyOwner sup(y) incrementX(y){    infinite gas
65      }
66  }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 8/19

8/19

Tutorial menu

...not be used as parameters or return parameters of contract functions that are publicly visible." From the Solidity documentation.

Arrays can be used as parameters, as shown in the function `arrayInput` (line 71). 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 Answer | Show answer

Next

Well done! No errors.

```
68      // Cannot use map for neither input nor output
69
70      // Can use array for input
71      function arrayInput(uint[] memory _arr) public {}    infinite gas
72
73      // Can use array for output
74      uint[] public arr;
75
76      function arrayOutput() public view returns (uint[] memory) {    infinite gas
77          return arr;
78      }
79      function returnTwo()    479 gas
80          public
81          pure
82          returns (
83              int i,
84              bool a
85          )
86      {
87          i = -2;
88          a = true;
89      }
90  }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 9/19

In this example, we have two contracts, the `Base` contract (line 4) and the `Child` contract (line 55) which inherits the functions and state variables from the `Base` contract.

When you uncomment the `testPrivateFunc` (lines 58-60) you get an error because the child contract doesn't have access to the private function `privateFunc` from the `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.

```
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) {    infinite gas
64          return internalFunc();
65      }
66      function testInternalVar() public view returns (string memory, string memory) {    infinite gas
67          return (internalVar, publicVar);
68      }
69  }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 10/19

10/19

If the first condition (line 6) of the foo function is not met, but the condition of the `else if` statement (line 8) becomes true, the function returns `1`.

Watch a video tutorial on the If/Else statement.

★ Assignment

Create a new function called `evenCheck` in the `IfElse` contract:

- That takes in a `uint` as an argument.
- The function returns `true` if the argument is even, and `false` if the argument is odd.
- Use a ternery 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.

```
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      function evenCheck(uint y) public pure returns (bool) {    infinite gas
25          if (y%2 == 0){ return true ;}
26          else{
27              return false;
28          }
29      }
30  }
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 11/19

11/19

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 {    infinite gas
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 }
27
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce6

➢ 12/19

12/19

elements stay the same, which means that the length of the array will stay the 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.

```
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     // Solidity can return the entire array.
17     // But this function should be avoided for
18     // arrays that can grow indefinitely in length.
19     function getArr() public view returns (uint[3] memory) {    infinite gas
20         return arr3;
21     }
22
23     function push(uint i) public {    46829 gas
24         // Append to array
25         // This will increase the array length by 1.
26         arr.push(i);
27     }
28
29     function pop() public {    29467 gas
30         // Remove last element from array
31         // This will decrease the array length by 1
```

0 listen on all transactions Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 13/19

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.

```
pragma solidity ^0.8.3;

contract Mapping {
    // Mapping from address to uint
    mapping(address => uint) public balances;

    function get(address _addr) public view returns (uint) {    2885 gas
        // Mapping always returns a value.
        // If the value was never set, it will return the default value.
        return balances[_addr];
    }

    function set(address _addr) public {    25265 gas
        // Update the value at this address
        balances[_addr] = _addr.balance;
    }

    function remove(address _addr) public {    5576 gas
        // Reset the value to the default value.
        delete balances[_addr];
    }
}


contract NestedMapping {
    // Nested mapping (mapping from address to another mapping)
    mapping(address => mapping(uint => bool)) public nested;

    function get(address _addr1, uint _i) public view returns (bool) {    3178 gas
        // You can get values from a nested mapping
```

0 listen on all transactions Search with transaction hash or address

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 14/19

Tutorial menu

the keys and values as a mapping inside curly braces (line 19).

Initialize and update a struct: We initialize an empty struct first and then update its member by assigning it a new value (line 23).

Accessing structs

To access a member of a struct we can use the dot operator (line 33).

Updating structs

To update a structs' member we also use the dot operator and assign it a new value (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

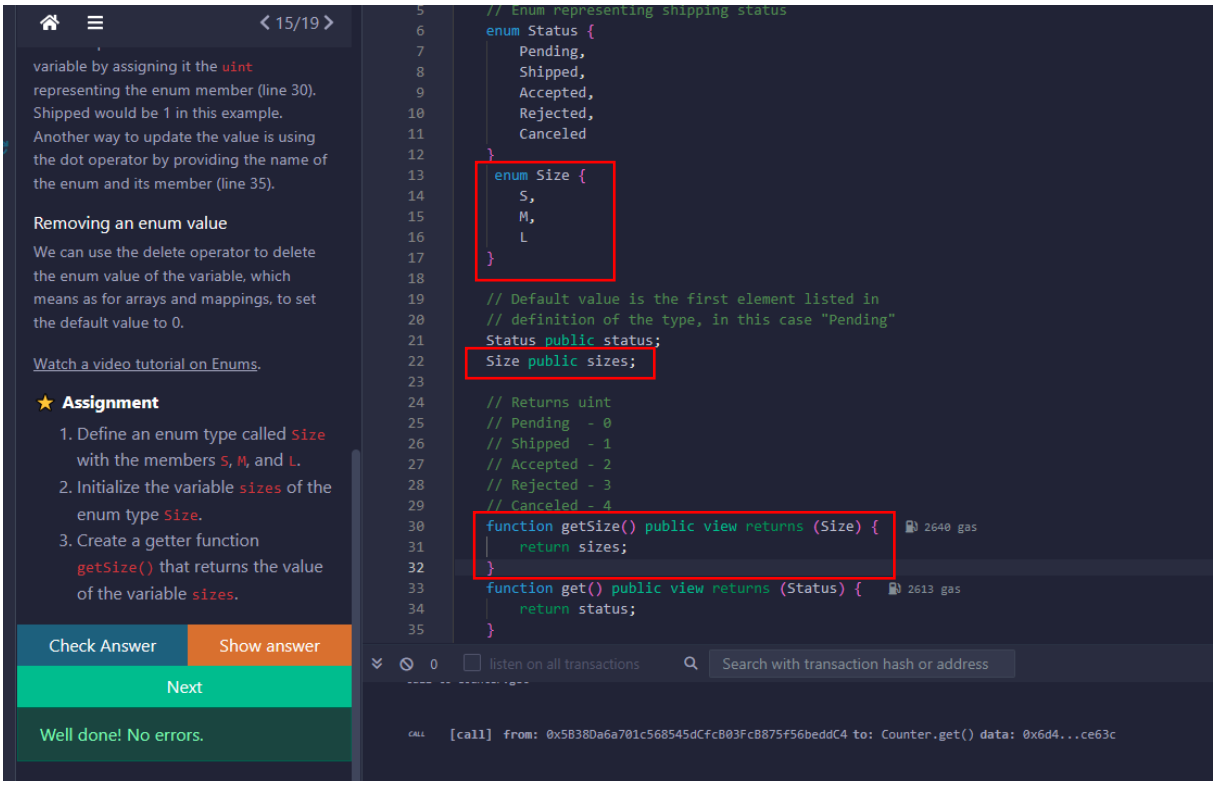
Well done! No errors.

```
25
26          todos.push(todo);
27      }
28
29      // Solidity automatically created a getter for 'todos' so
30      // you don't actually need this function.
31      function get(uint _index) public view returns (string memory text, bool completed) {    infinite gas
32          Todo storage todo = todos[_index];
33          return (todo.text, todo.completed);
34      }
35
36      // update text
37      function update(uint _index, string memory _text) public {    infinite gas
38          Todo storage todo = todos[_index];
39          todo.text = _text;
40      }
41
42      // update completed
43      function toggleCompleted(uint _index) public {    29006 gas
44          Todo storage todo = todos[_index];
45          todo.completed = !todo.completed;
46      }
47      function remove(uint _index) public {    infinite gas
48          delete todos[_index];
49      }
50  }
```

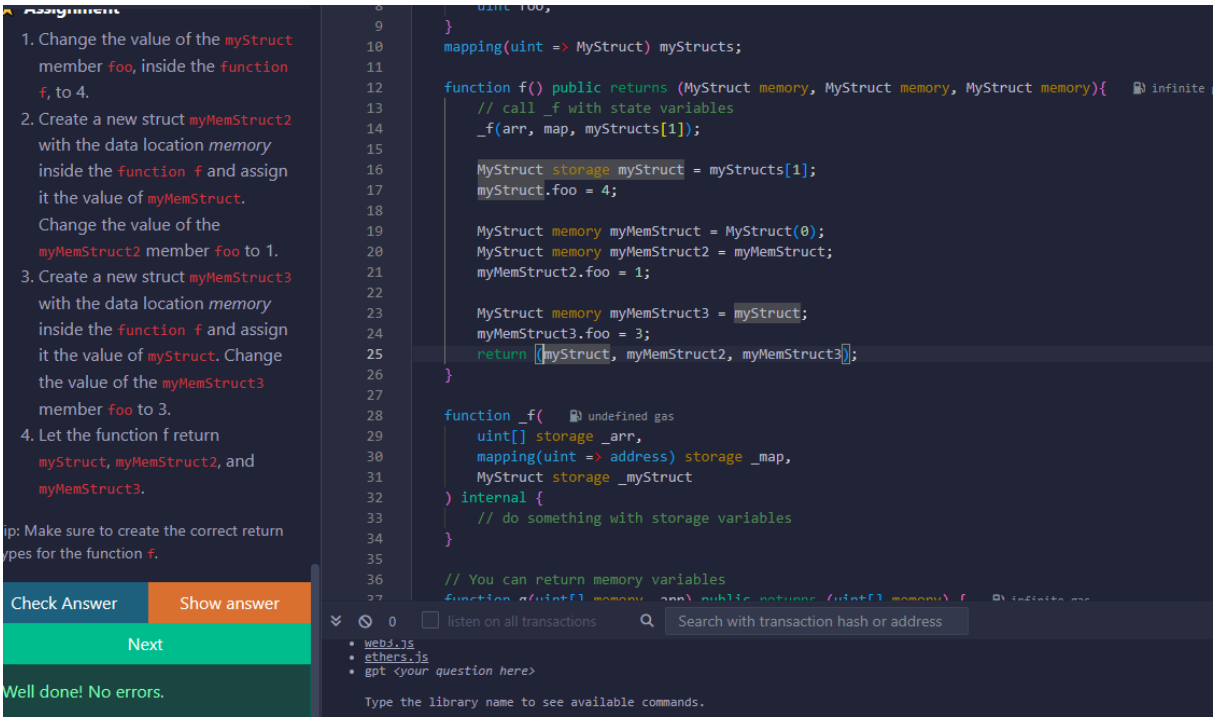
0 listen on all transactions Search with transaction hash or address

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 15/19


15/19
variable by assigning it the uint representing 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.
// Enum representing shipping status
enum Status {
Pending,
Shipped,
Accepted,
Rejected,
Canceled
}
enum Size {
S,
M,
L
}
// Default value is the first element listed in
// definition of the type, in this case "Pending"
Status public status;
Size public sizes;
// Returns uint
// Pending - 0
// Shipped - 1
// Accepted - 2
// Rejected - 3
// Canceled - 4
function getSize() public view returns (Size) {
return sizes;
}
function get() public view returns (Status) {
return status;
}
2640 gas
2613 gas
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c

➢ 16/19


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.
ip: Make sure to create the correct return ypes for the function f.
Check Answer
Show answer
Next
Well done! No errors.
}
mapping(uint => MyStruct) myStructs;
function f() public returns (MyStruct memory, MyStruct memory, MyStruct memory){
// call _f with state variables
_f(arr, map, myStructs[1]);
MyStruct storage myStruct = myStructs[1];
myStruct.foo = 4;
MyStruct memory myMemStruct = MyStruct(0);
MyStruct memory myMemStruct2 = myMemStruct;
myMemStruct2.foo = 1;
MyStruct memory myMemStruct3 = myStruct;
myMemStruct3.foo = 3;
return (myStruct, myMemStruct2, myMemStruct3);
}
function _f(
uint[] storage _arr,
mapping(uint => address) storage _map,
MyStruct storage _myStruct
) internal {
// do something with storage variables
}
// You can return memory variables
undefined gas
listen on all transactions
Search with transaction hash or address
web3.js
ethers.js
gpt <your question here>
Type the library name to see available commands.

➢ 17/19

One ether is equal to 1,000,000,000,000,000,000 (10^18) wei (line 11).

Watch a video tutorial on Ether and Wei.

★ **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.

```
11      bool public isOneEther = 1 ether == 1e18;
12
13      uint public oneGwei = 1 gwei;
14      bool public isOneGwei = 1 gwei == 1e9;
15  }
```

0 listen on all transactions Search with tra

- web3.js
- ethers.js
- gpt <your question here>

Type the library name to see availa

➢ 19/19

Watch a video tutorial on Sending Ether.

★ **Assignment**

Build a charity contract that receives Ether that can be withdrawn by a beneficiary.

1. Create a contract called Charity.
2. Add a public state variable called owner of the type address.
3. Create a donate function that is public and payable without any parameters or function code.
4. Create a withdraw function that is public and sends the total balance of the contract to the owner address.

Tip: Test your contract by deploying it from one account and then sending Ether to it from another account. Then execute the withdraw function.

Check Answer | Show answer

Next

Well done! No errors.

```
    }

    function sendViaCall(address payable _to) public payable {    undefined gas
        // Call returns a boolean value indicating success or failure.
        // This is the current recommended method to use.
        (bool sent, bytes memory data) = _to.call{value: msg.value}("");
        require(sent, "Failed to send Ether");
    }
}

contract Charity {
    address public owner;

    constructor() {    170056 gas 145600 gas
        owner = msg.sender;
    }

    function donate() public payable {}    142 gas

    function withdraw() public {    infinite gas
        uint amount = address(this).balance;

        (bool sent, bytes memory data) = owner.call{value: amount}("");
        require(sent, "erreur lors de l'envoie d'Eth");
    }
}
```

0 listen on all transactions Search with transaction hash or address

- Execute JavaScript scripts:
  - *Input a script directly in the command line interface*
  - *Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()`*
  - *Right click on a JavaScript file in the file explorer and then click `Run`*