SOLIDITY PROGRAMMING

Karachi Institute of Technology and Entrepreneurship (KITE)

Session 1: Solidity Basics

Date: 29th October 2021

Instructor: Syed Faisal ur Rahman

Readings

https://www.tutorialspoint.com/solidity/index.htm

https://www.dappuniversity.com/articles/solidity-tutorial

https://101blockchains.com/solidity-tutorial/

https://www.geeksforgeeks.org/introduction-to-solidity/?ref=leftbar-rightbar

https://betterprogramming.pub/learn-solidity-functions-ddd8ea24c00d

https://www.bitdegree.org/learn/solidity-types

https://www.tutorialspoint.com/solidity/solidity_mappings.htm

https://medium.com/coinmonks/what-the-hack-is-memory-and-storage-in-solidity-6b9e62577305

https://www.ops.gov.ie/app/uploads/2021/01/Blockchain-Develop-Deploy-and-Test-Your-First-Smart-Contract.pdf

https://ethereumbuilders.gitbooks.io/guide/content/en/solidity_tutorials.html

DEVELOPMENT ENVIRONMENT

https://remix.ethereum.org/

TASK 1: BASIC APPLICATION WITH ONE FUNCTION

Step 1: Open remix.ethereum.org

Step 2: Create a code file (SolidityTest.sol) inside the contract folder.

Step 3: Write the following code in the file:

```
pragma solidity >=0.4.0 <0.6.0;  //pragma directive to tell the compiler about the Solidity version

contract SolidityTest {
  constructor() public{

  }

  function addNum() public view returns(int){

    int a = 3;

    int b = -2;

    int result = a + b;

    return result;

  }

}
```

Step 4: Click on the Solidity compiler button on the left side of your screen.

Step 5: After compiling the code, go to the “Deploy & run transactions” part by clicking the button on the left bar where the Solidity compiler button is.

Step 6: In the “Deploy & run transactions”, click the Deploy button.

Step 7: In the “Deploy & run transactions”, click on “SOLIDITYTEST AT 0x…..”.

Step 8: Click on addNum.

You will now see your output:

```
O:int256 1
```

TASK 2: BASIC APPLICATION WITH TWO FUNCTIONS

Step 1: Open remix.ethereum.org

Step 2: Create a code file (SolidityTest.sol) inside the contract folder.

Step 3: Write the following code in the file:

```
pragma solidity >=0.4.0 <0.6.0;  //pragma directive to tell the compiler about the Solidity version

contract SolidityTest {

  constructor() public{

  }

  function addNum() public view returns(int){
    int a = 3;

    int b = -2;

    int result = a + b;

    return result;

  }

  function subNum() public view returns(int){

    int a = 3;

    int b = -2;

    int result = a-b;

    return result;

  }

}
```

Step 4: Click on the Solidity compiler button on the left side of your screen.

Step 5: After compiling the code, go to the "Deploy & run transactions" part by clicking the button on the left bar where the Solidity compiler button is.

Step 6: In the "Deploy & run transactions", click the Deploy button.

Step 7: In the "Deploy & run transactions", click on "SOLIDITYTEST AT 0x…..".

Step 8: Click on addNum and subNum.

You will now see your output:

```
o:int256 1

o:int256 5
```

READING: DATA TYPES, VARIABLES, VARIABLE SCOPE, OPERATORS AND ARRAYS

https://www.tutorialspoint.com/solidity/solidity_types.htm

https://www.tutorialspoint.com/solidity/solidity_variables.htm

https://www.tutorialspoint.com/solidity/solidity_variable_scope.htm

https://www.tutorialspoint.com/solidity/solidity_operators.htm

https://www.tutorialspoint.com/solidity/solidity_arrays.htm

TASK 3: ARRAYS

Creating and accessing elements of an array:

```
// Solidity program to demonstrate accessing elements of an array

// Ref: https://www.geeksforgeeks.org/solidity-arrays/?ref=lbp

pragma solidity ^0.5.0;

// Creating a contract

contract Types {

   // Declaring an array

   uint[6] data;

   // Defining function to

   // assign values to array

   function array_example(

   ) public payable returns (uint[6] memory){

       data

         = [10, 20, 30, 40, 50, 60];

       return data;

  }

  // Defining function to access

  // values from the array

  // from a specific index

  function array_element(

  ) public payable returns (uint){

       uint x = data[0];

       return x;

  }

}
```

READING: STRINGS, ENUMS, STRUCTS, MAPPINGS, CONVERSIONS, ETHER UNITS, SPECIAL VARIABLES

https://www.tutorialspoint.com/solidity/solidity_strings.htm

https://www.tutorialspoint.com/solidity/solidity_enums.htm

https://www.tutorialspoint.com/solidity/solidity_structs.htm

https://www.tutorialspoint.com/solidity/solidity_mappings.htm

https://www.tutorialspoint.com/solidity/solidity_conversions.htm

https://www.tutorialspoint.com/solidity/solidity_ether_units.htm

https://www.tutorialspoint.com/solidity/solidity_special_variables.htm

TASK 4: ENUMS

```
//Ref: https://www.tutorialspoint.com/solidity/solidity_enums.htm

pragma solidity ^0.5.0;

contract test {

  enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }

  FreshJuiceSize choice;

  FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;

  function setLarge() public {

    choice = FreshJuiceSize.LARGE;

  }

  function getChoice() public view returns (FreshJuiceSize) {

    return choice;

  }

  function getDefaultChoice() public pure returns (uint) {

    return uint(defaultChoice);

  }

}
```

TASK 5: STRUCTS

```
//Solidity struct demo

// Ref: https://www.tutorialspoint.com/solidity/solidity_structs.htm

pragma solidity ^0.5.0;

contract test {

  struct Book {

    string title;

    string author;

    uint book_id;

  }

  Book book;


  function setBook() public {

    book = Book('Learn Java', 'TP', 1);

  }

  function getBookId() public view returns (uint) {

    return book.book_id;

  }

}
```

READING: LOOPS AND DECISION MAKING

https://www.geeksforgeeks.org/solidity-while-do-while-and-for-loop/?ref=lbp

https://www.tutorialspoint.com/solidity/solidity_decision_making.htm

TASK 6: FOR AND WHILE LOOPS

FOR LOOP

```
// Solidity program to demonstrate the use of 'For loop'

//Ref: https://www.geeksforgeeks.org/solidity-while-do-while-and-for-loop/?ref=lbp
```

```
pragma solidity ^0.5.0;

// Creating a contract

contract Types {

    // Declaring a dynamic array

    uint[] data;

    // Defining a function

    // to demonstrate 'For loop'

    function loop(

    ) public returns(uint[] memory){

    for(uint i=0; i<5; i++){

        data.push(i);

     }

      return data;

    }

}
```

WHILE LOOP

```
// Solidity program to demonstrate the use of 'While loop'

// Ref: https://www.geeksforgeeks.org/solidity-while-do-while-and-for-loop/?ref=lbp

pragma solidity ^0.5.0;


// Creating a contract

contract Types {


    // Declaring a dynamic array

    uint[] data;
    // Declaring state variable
    uint8 j = 0;

    // Defining a function to
    // demonstrate While loop'
    function loop(
    ) public returns(uint[] memory){
    while(j < 10) {
        j++;
        data.push(j);
     }
      return data;
    }
}
```

READING: REFERENCE TYPES, STORAGE, MEMORY AND Mappings

https://www.c-sharpcorner.com/article/reference-types-in-solidity/

https://www.geeksforgeeks.org/storage-vs-memory-in-solidity/

https://www.bitdegree.org/learn/solidity-types#mapping-types

https://www.tutorialspoint.com/solidity/solidity_mappings.htm

TASK 7: STORAGE AND MEMORY

STORAGE

```
// Ref: https://www.geeksforgeeks.org/storage-vs-memory-in-solidity/
pragma solidity ^0.4.17;
// Creating a contract
```

```
contract helloGeeks

{

// Initialising array numbers

int[] public numbers;


// Function to insert values

// in the array numbers

function Numbers() public returns(int[] memory)

{

        numbers.push(1);

        numbers.push(2);


        //Creating a new instance

        int[] storage myArray = numbers;


        // Adding value to the

        // first index of the new Instance

        myArray[0] = 0;


        return numbers;

}
```

MEMORY

//Ref: https://www.geeksforgeeks.org/storage-vs-memory-in-solidity/

```
pragma solidity ^0.4.17;


// Creating a contract
```

```
contract helloGeeks

{

// Initialising array numbers

int[] public numbers;


// Function to insert

// values in the array

// numbers

function Numbers() public returns(int[] memory)

{

        numbers.push(1);

        numbers.push(2);


        //creating a new instance

        int[] memory myArray = numbers;


        // Adding value to the first

        // index of the array myArray

        myArray[0] = 0;

        return numbers;

}

}
```

TASK 8: MAPPINGS

```
//Ref: https://www.geeksforgeeks.org/solidity-mappings/?ref=leftbar-rightbar

// Solidity program to demonstrate adding values to mapping

pragma solidity ^0.4.18;

// Creating contract
```

```
contract mapping_example {

        //Defining structure
        struct student {

                //Declaring different
                // structure elements
                string name;
                string subject;
                uint8 marks;
        }

        // Creating mapping
        mapping (
        address => student) result;
        address[] public student_result;

        // Function adding values to
        // the mapping
        function adding_values() public returns(string memory) {
                var student
                = result[0xDEE7796E89C82C36BAdd1375076f39D69FafE252];

                student.name = "John";
                student.subject = "Chemistry";
                student.marks = 88;
                student_result.push(
                0xDEE7796E89C82C36BAdd1375076f39D69FafE252) -1;

                return result[student_result[0]].name;


        }
```