

## Code

### 1. Array element Access

```
1 // Solidity program to demonstrate
2 // accessing elements of an array
3 // SPDX-License-Identifier: MIT
4 pragma solidity ^0.8.0;
5
6 // Creating a contract
7 contract Types {
8     // Declaring an array
9     uint[6] data;
10    // Defining function to
11    // assign values to array
12    function array_example() public payable returns (uint[6] memory){ infinite gas
13        data = [uint(10), 20, 30, 40, 50, 60];
14        return data;
15    }
16    // Defining function to access
17    // values from the array
18    // from a specific index
19    function array_element() public payable returns (uint){ 2433 gas
20        uint x = data[2];
21        return x;
22    }
23 }
```

### 2. Array length check

```
3 // SPDX-License-Identifier: MIT
4 pragma solidity ^0.8.0;
5 // Creating a contract
6 contract Types {
7     // Declaring an array
8     uint[6] data;
9     // Defining a function to
10    // assign values to an array
11    function array_example( infinite gas
12    ) public payable returns (uint[6] memory){
13        data = [uint(10), 20, 30, 40, 50, 60];
14        return data;
15    }
16    // Defining a function to
17    // find the length of the array
18    function array_length() public returns(uint) { 328 gas
19        uint x = data.length;
20        return x;
21    }
22 }
```

### 3. Array data push

```
1 // Solidity program to demonstrate
2 // Push operation
3 // SPDX-License-Identifier: MIT
4 pragma solidity ^0.8.0;
5 // Creating a contract
6 contract Types {
7     // Defining the array
8     uint[] data = [10, 20, 30, 40, 50];
9     // Defining the function to push
10    // values to the array
11    function array_push() public returns(uint[] memory){ infinite gas
12        data.push(60);
13        data.push(70);
14        data.push(80);
15        return data;
16    }
17 }
```



## Output













### 1. Deploying a contract

The screenshot shows the Remix IDE interface. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which includes fields for 'ENVIRONMENT' (Remix VM (Merge)), 'ACCOUNT' (0x5B3...ddC4), 'GAS LIMIT' (3000000), 'VALUE' (0), and 'CONTRACT' (Types - contracts/assignment3A.sol). The 'Deploy' button is highlighted. The main editor displays the Solidity code for the 'Types' contract, which includes an array 'data' and an 'array\_push' function. The bottom console shows the deployment output: '[vm] from: 0x5B3...ddC4 to: Types.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x3a9...58967'.

## 2. Executing code

### a. Accessing Array Elements

 0 ☐ listen on all transactions  Search with transaction hash or address

transaction hash	0xbecfa8182076e9ed14d55bf660c265326fd74034c3a512f557f85a3b5f15d0cc 
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 
to	Types.array_example() 0xd9145CCE52D386f254917e481eB44e9943F39138 
gas	43944 gas 
transaction cost	38212 gas 
execution cost	17148 gas 
input	0x672...27992 
decoded input	{ } 
decoded output	<pre>{   "0": "uint256[6]: 10,20,30,40,50,60" }</pre> 
logs	[ ]  
val	0 wei 

>

### b. Checking length of Array

transaction hash	0xc922d8c297caf9c43f44ba93f970365f4a4575803181fadc9d046e2f312605ab
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	Types.array_length() 0xDA0bab807633f07f013f94D00E6A4F96F8742B53
gas	24601 gas
transaction cost	21392 gas
execution cost	328 gas
input	0x0cc...008bd
decoded input	{}
decoded output	{           "0": "uint256: 6"         }
logs	[]
val	0 wei

### c. Pushing data to Array

transaction hash	0x5369052c7baf9f94ee5d522427ae31e2d4b02b90dccf76223de7953d6307b1f
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	Types.array_push() 0xd2a5bC10698FD955D1Fe6cb468a17809A08fd005
gas	123506 gas
transaction cost	107396 gas
execution cost	86332 gas
input	0x7d6...e3dd0
decoded input	{}
decoded output	{           "0": "uint256[]: 10,20,30,40,50,60,70,80"         }
logs	[]
val	0 wei