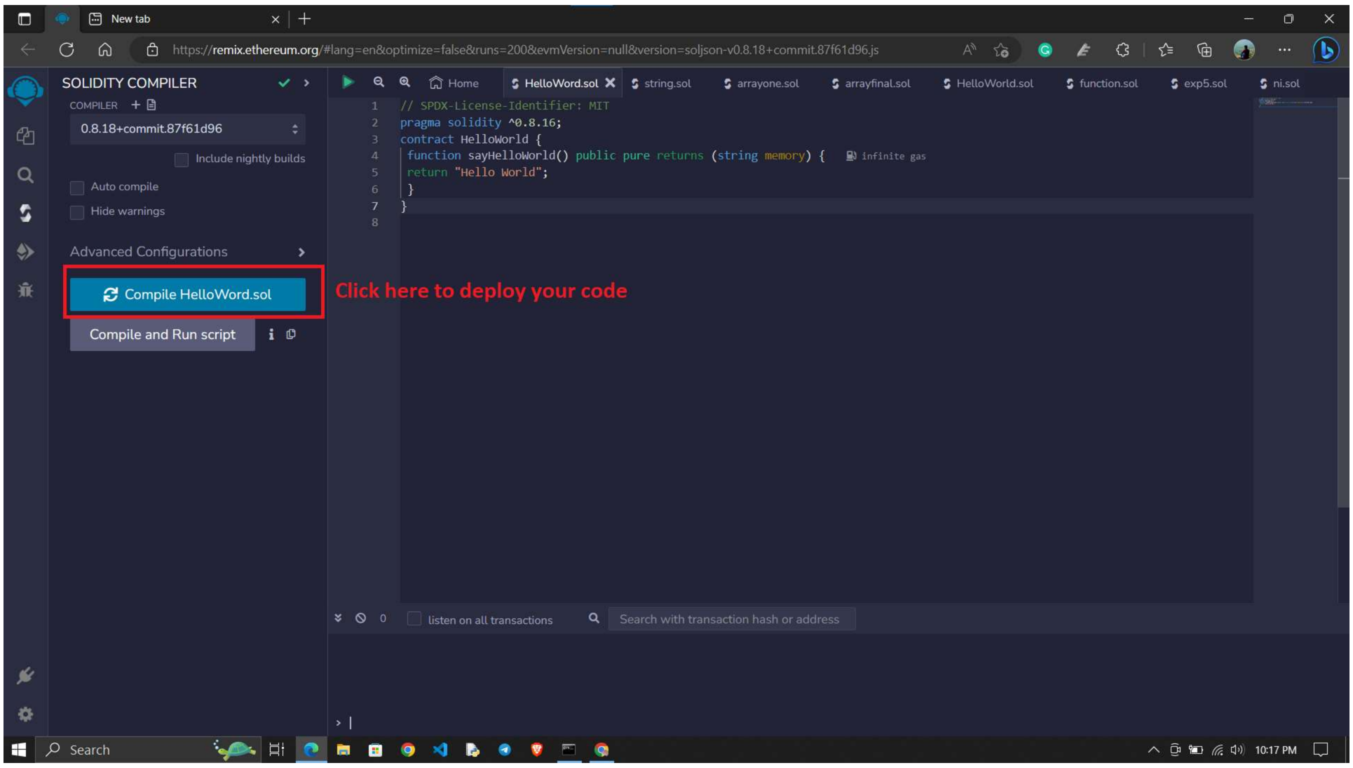
1. Open [https://remix.ethereum.org/ (https://remix.ethereum.org/)](https://remix.ethereum.org/)
2. click on create new file and and run this program
3. press ctrl+s to compile or click on green play icon

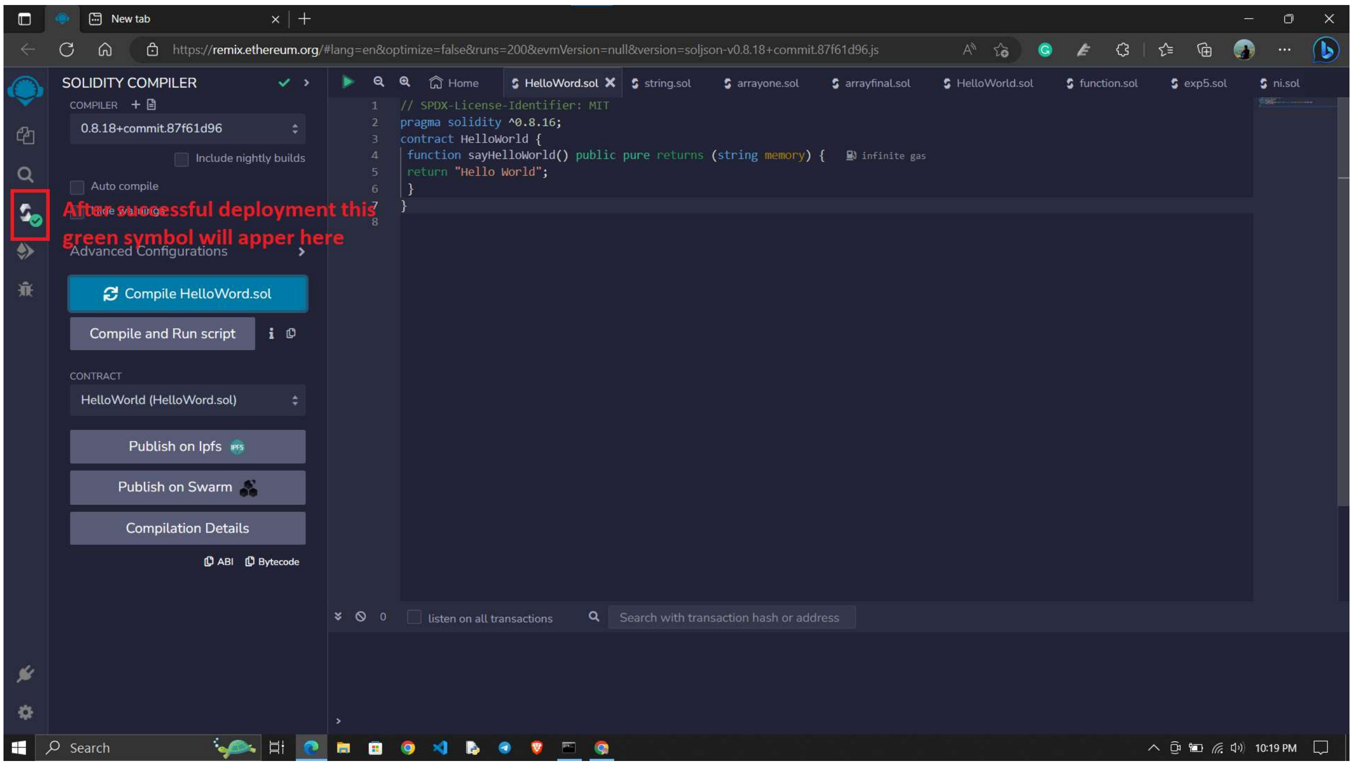


https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js

COMPILER: 0.8.18+commit.87f61d96

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {
 return "Hello World";
 }
}
```

Click here to deploy your code

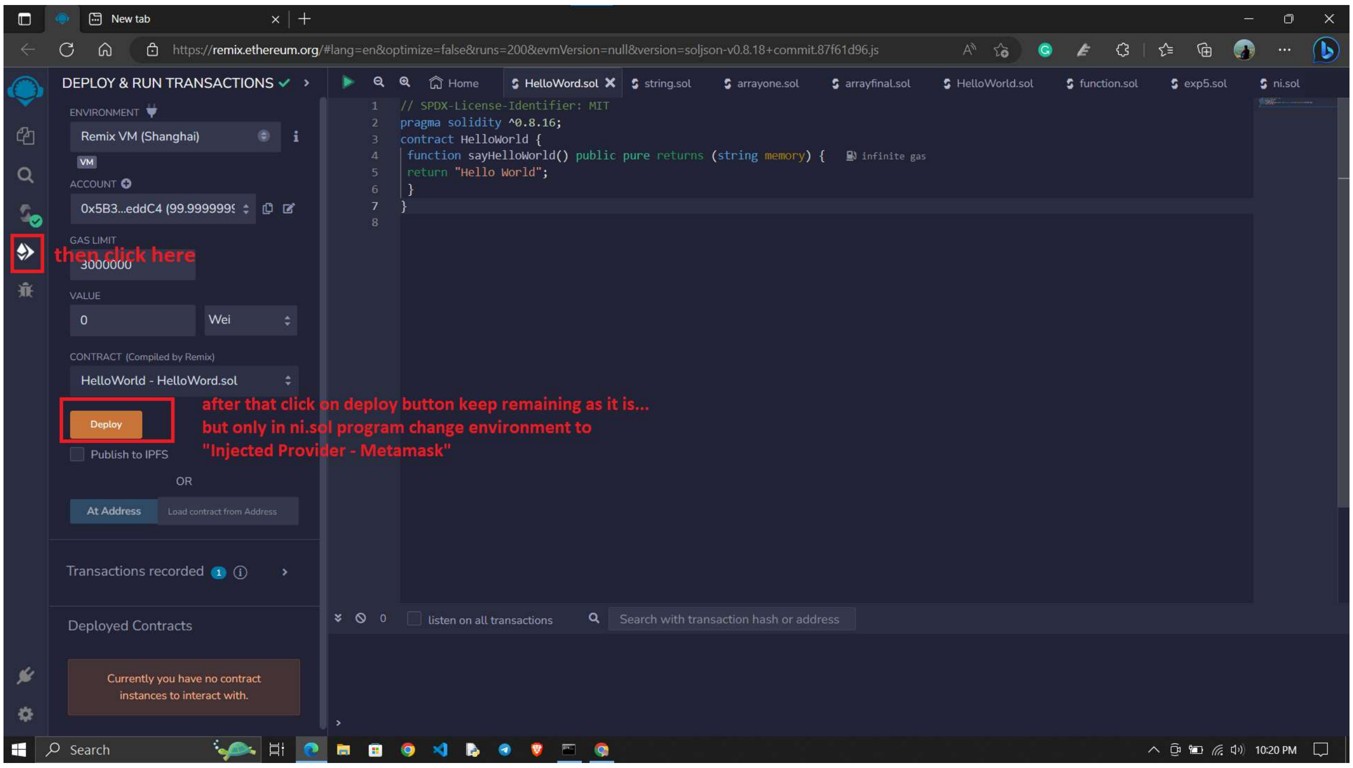


COMPILER: 0.8.18+commit.87f61d96

CONTRACT: HelloWorld (HelloWord.sol)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {
 return "Hello World";
 }
}
```

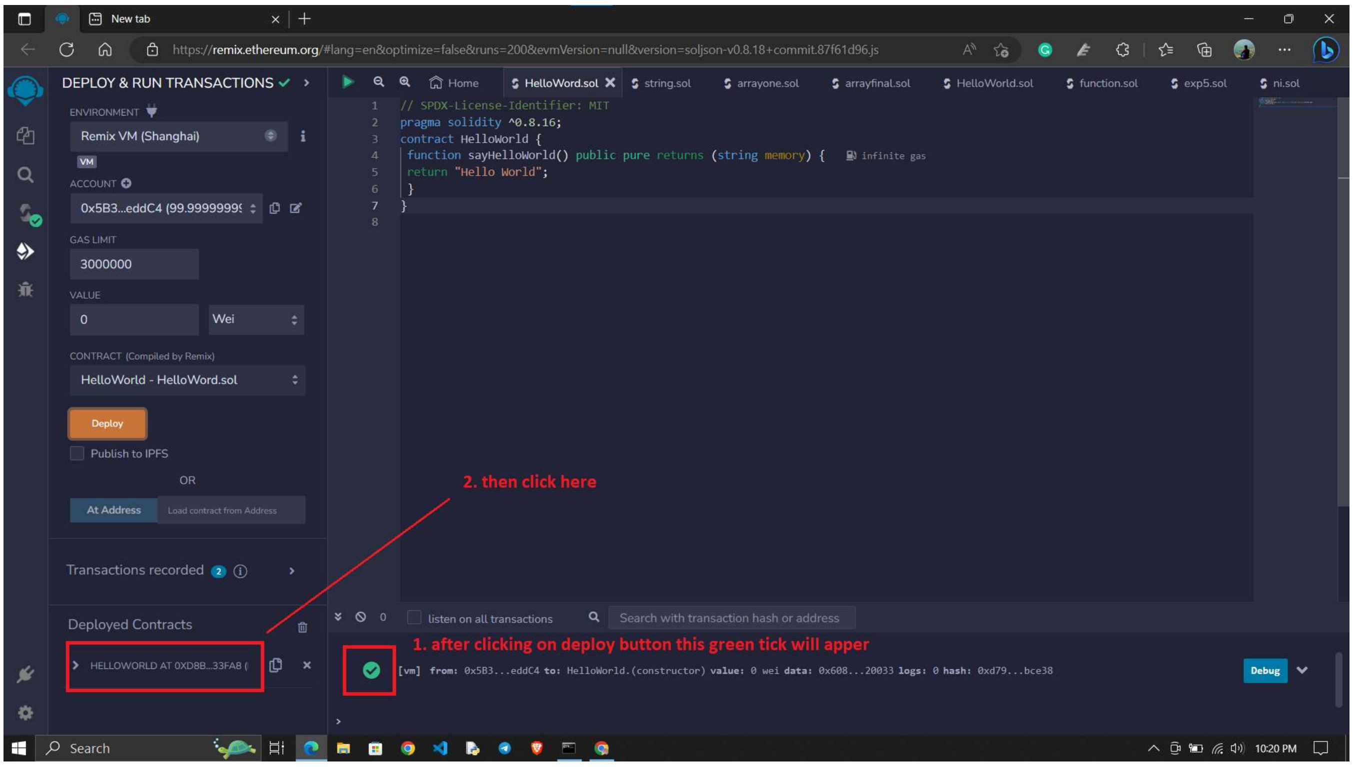
After successful deployment this green symbol will apper here


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Shanghai)
ACCOUNT
0x5B3...eddC4 (99.9999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled by Remix)
HelloWorld - HelloWord.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
Currently you have no contract instances to interact with.
listen on all transactions
Search with transaction hash or address

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {   infinite gas
 return "Hello World";
 }
}
```

then click here

after that click on deploy button keep remaining as it is...
but only in ni.sol program change environment to
"Injected Provider - Metamask"

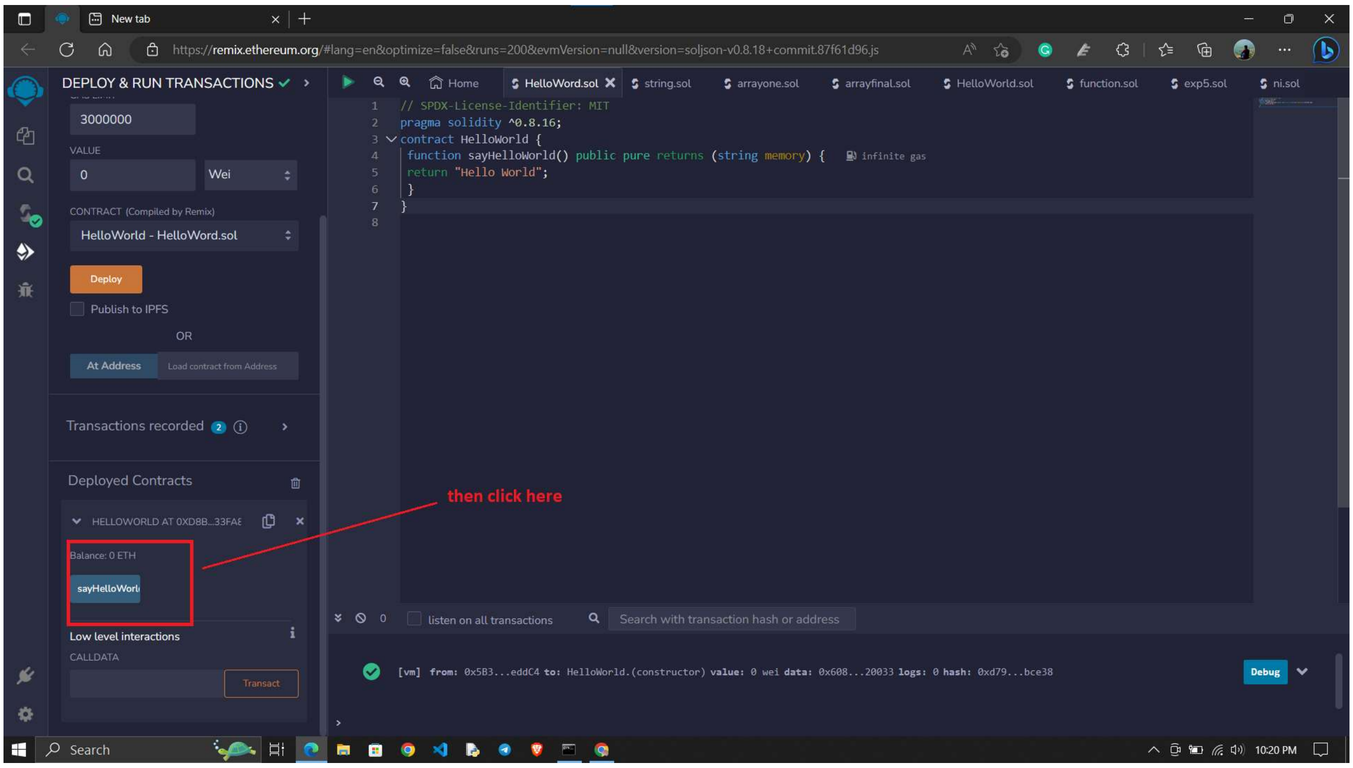

DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Shanghai)
ACCOUNT
0x5B3...eddC4 (99.9999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled by Remix)
HelloWorld - HelloWord.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 2
Deployed Contracts
HELLOWORLD AT 0XD8B...33FA8 (
listen on all transactions
Search with transaction hash or address
Debug

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {   infinite gas
 return "Hello World";
 }
}
```

2. then click here

1. after clicking on deploy button this green tick will apper

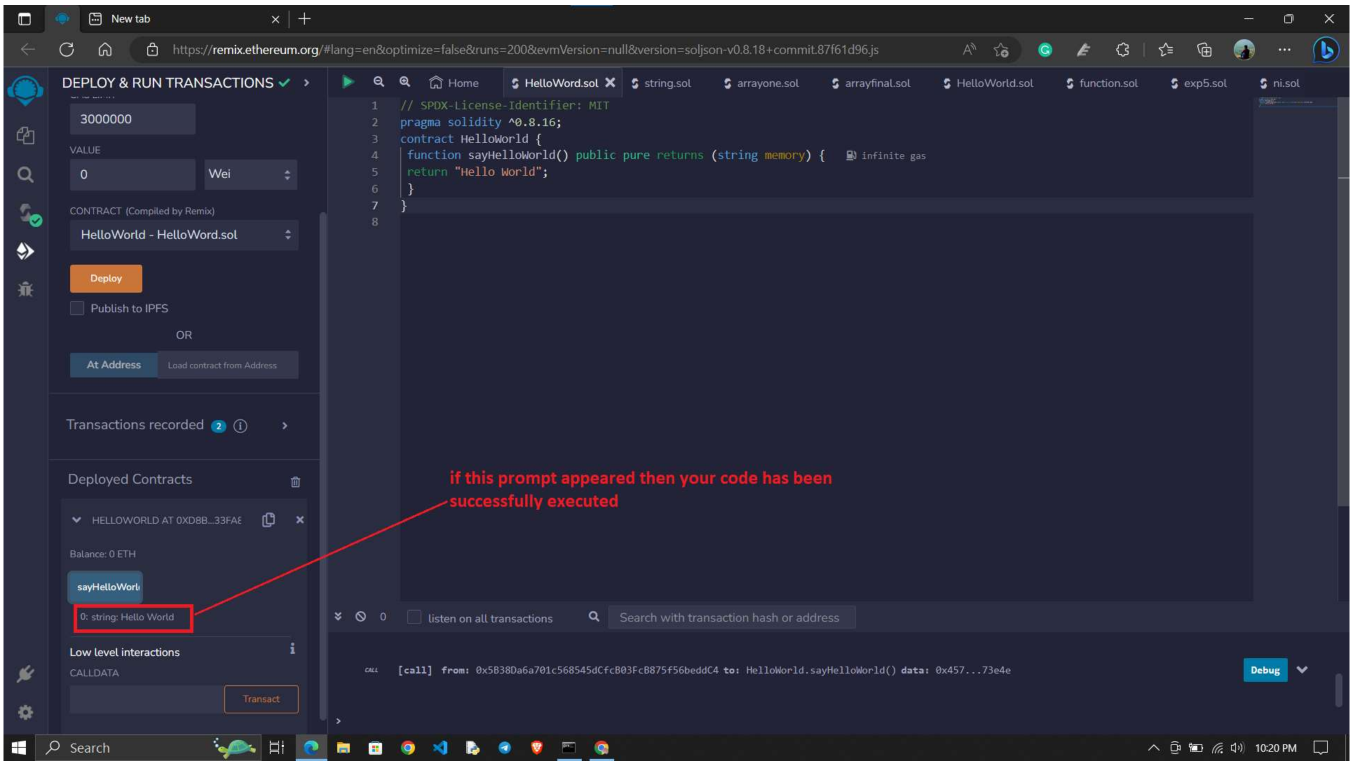
```
[vm] from: 0x5B3...eddC4 to: HelloWorld.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xd79...bce38
```



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {
 return "Hello World";
 }
}
```

```
[vm] from: 0x5B3...eddC4 to: HelloWorld.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xd79...bce38
```

DEPLOY & RUN TRANSACTIONS
3000000
VALUE
0
Wei
CONTRACT (Compiled by Remix)
HelloWorld - HelloWord.sol
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 2
Deployed Contracts
HELLOWORLD AT 0XDBB...33FAE
Balance: 0 ETH
sayHelloWorl
Low level interactions
CALLDATA
Transact
then click here
listen on all transactions
Search with transaction hash or address
Debug



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {
 return "Hello World";
 }
}
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: HelloWorld.sayHelloWorld() data: 0x457...73e4e
```

DEPLOY & RUN TRANSACTIONS
Deployed Contracts
HELLOWORLD AT 0XDBB...33FAE
Balance: 0 ETH
sayHelloWorl
0: string: Hello World
Low level interactions
Transact
if this prompt appeared then your code has been successfully executed
Debug

ni.sol

for this program open your `MetaMask` wallet first and then in `Deploy and run transaction` change environment to `"Injected Provider -MetaMask"` and then connect to Metamask wallet. After that follow all steps as it is

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;
// Creating a contract
contract shreyansh_05
```

```
{
 // Defining a function
 function get_output() public pure returns (string memory){
 return ("Hi, your contract ran successfully");
 }
}
```

HelloWorld.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.16;
contract HelloWorld {
 function sayHelloWorld() public pure returns (string memory) {
 return "Hello World";
 }
}
```

Another different program of Helloworld.sol

```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equal to 0.8.17 and less than 0.9.0
pragma solidity ^0.8.17;
contract HelloWorld {
 string public greet = "Hello World!";
}
```

function.sol

```
                                          // SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;
contract Function {
 // Functions can return multiple values.
 function returnMany() public pure returns (uint, bool, uint) {
 return (1, true, 2);
 }
 // Return values can be named.
 function named() public pure returns (uint x, bool b, uint y)
 {
 return (1, true, 2);
 }
 // Return values can be assigned to their name.
 // In this case the return statement can be omitted.
 function assigned() public pure returns (uint x, bool b, uint y) {
 x = 1;
 b = true;
 y = 2;
 }
 // Use destructuring assignment when calling another
 // function that returns multiple values.
 function destructuringAssignments()
 public
 pure
 returns (uint, bool, uint, uint, uint)
 {
 (uint i, bool b, uint j) = returnMany();
 // Values can be left out.
 (uint x, , uint y) = (4, 5, 6);
 return (i, b, j, x, y);
 }
 // Cannot use map for either input or output
 // Can use array for input
 function arrayInput(uint[] memory _arr) public {}
 // Can use array for output
 uint[] public arr;
 function arrayOutput() public view returns (uint[] memory) {
 return arr;
 }
}
// Call function with key-value inputs
contract XYZ {
 function someFuncWithManyInputs(
 uint x,
 uint y,
 uint z,
 address a,
 bool b,
 string memory c
 ) public pure returns (uint) {}
 function callFunc() external pure returns (uint) {
 return someFuncWithManyInputs(1, 2, 3, address(0), true, "c");
 }
 function callFuncWithKeyValue() external pure returns (uint) {
 return
 someFuncWithManyInputs({a: address(0), b: true, c: "c", x: 1, y: 2, z: 3});
 }
}
```

In [ ]: