

[Deploy to Base Sepolia](#)[Contract Verification](#)[Exercise](#)[Control Structures](#) ▾[Standard Control Structures](#)[Loops](#)[Require, Revert, and Error](#)[Overview](#)[Exercise](#)[Storage in Solidity](#) ▾[Simple Storage](#)[Step by Step Guide](#)[How Storage Works](#)[Storage Overview](#)[Exercise](#)[Arrays in Solidity](#) ▾

Storage in Solidity

Storage Exercise

[Copy page](#)

Exercise - Demonstrate your knowledge of storage.

Create a contract that adheres to the following specifications:

Contract

Create a single contract called `EmployeeStorage`. It should not inherit from any other contracts. It should have the following functions:

State Variables

The contract should have the following state variables, optimized to minimize storage:

A private variable `shares` storing the employee's number of shares owned

Deploy to Base Sepolia

Contract Verification

Exercise



Standard Control Structures

Loops

Require, Revert, and Error

Overview

Exercise



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Employees with more than 5,000 shares count as directors and are stored in another contract

Public variable `name` which stores the employee's name

A private variable `salary` storing the employee's salary

Salaries range from 0 to 1,000,000 dollars

A public variable `idNumber` storing the employee's ID number

Employee numbers are not sequential, so this field should allow any number up to $2^{256}-1$

Constructor

When deploying the contract, utilize the `constructor` to set:

`shares`

`name`

`salary`

`idNumber`

For the purposes of the test, you **must** deploy the contract with the following values:

`shares` - 1000

`name` - Pat

`salary` - 50000

`idNumber` - 112358132134

Deploy to Base Sepolia

Contract Verification

Exercise

Standard Control Structures

Loops

Require, Revert, and Error

Overview

Exercise

Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise

View Salary and View Shares

! In the world of blockchain, nothing is ever secret!* `private` variables prevent other contracts from reading the value. You should use them as a part of clean programming practices, but marking a variable as `private` **does not hide the value**. All data is trivially available to anyone who knows how to fetch data from the chain.

*You can make clever use of encryption though!

Write a function called `viewSalary` that returns the value in `salary`.

Write a function called `viewShares` that returns the value in `shares`.

Grant Shares

Add a public function called `grantShares` that increases the number of shares allocated to an employee by `_newShares`. It should:

Add the provided number of shares to the `shares`

If this would result in more than 5000 shares, revert with a custom error called `TooManyShares` that returns the number of shares the employee would have with the new amount added

If the number of `_newShares` is greater than 5000, revert with a string message, "Too many shares"

Check for Packing and Debug Reset Shares

Deploy to Base Sepolia

Contract Verification



Search... Ctrl K

[GitHub](#)

[Support](#)

[Base Build](#)



[Get Started](#)

[Base Chain](#)

[Base Account](#)

[Base App](#)

[Mini Apps](#)

[OnchainKit](#)

[Cookbook](#)

[Showcase](#)

[Learn](#)

Standard Control Structures

Loops

Require, Revert, and Error

Overview

Exercise



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

[Exercise](#)





```
/**  
 * Do not modify this function. It is used to enable the unit test for this  
 * to check whether or not you have configured your storage variables to make  
 * use of packing.  
 *  
 * If you wish to do something, always return `0`  
 * I'm not your boss ^\_\(\^)\_/  
 *  
 * Fair warning though, if you do cheat, it will be on the blockchain having  
 * deployed by your wallet....FOREVER!  
 */  
  
function checkForPacking(uint _slot) public view returns (uint r) {  
    assembly {  
        r := sload (_slot)  
    }  
}  
  
/**  
 * Warning: Anyone can use this function at any time!  
 */  
function debugResetShares() public {  
    shares = 1000;  
}
```

Deploy to Base Sepolia

Contract Verification

Exercise



Standard Control Structures

Loops

Require, Revert, and Error

Overview

Exercise



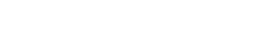
Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Submit your Contract and Earn an NFT Badge! (BETA)

⚠ Hey, where'd my NFT go!?

Testnets are not permanent! Base Goerli will soon be sunset, in favor of Base Sepolia.

As these are separate networks with separate data, your NFTs **will not** transfer over.

Don't worry! We've captured the addresses of all NFT owners on Base Goerli and will include them when we release the mechanism to transfer these NFTs to mainnet later this year! You can also redeploy on Sepolia and resubmit if you'd like!

Deploy to Base Sepolia

Contract Verification

Exercise



Standard Control Structures

Loops

Require, Revert, and Error

Overview

Exercise



Simple Storage

Step by Step Guide

How Storage Works

Storage Overview

Exercise



Was this page helpful?

👍 Yes

👎 No

✍ Suggest edits

⚠ Raise issue

◀ Storage Overview

Arrays >

e.org

Blog

Privacy Policy

Terms of Service

Cookie Policy

[Deploy to Base Sepolia](#)[Contract Verification](#)[Exercise](#)[Standard Control Structures](#)[Loops](#)[Require, Revert, and Error](#)[Overview](#)[Exercise](#)[Simple Storage](#)[Step by Step Guide](#)[How Storage Works](#)[Storage Overview](#)[Exercise](#)