base docs

Search...   Ctrl K   GitHub   Support   Base Build

Get Started   Base Chain   Base Account   Base App   Mini Apps   OnchainKit   Cookbook   Showcase   **Learn**

Contracts and Basic Functions

Basic Functions Exercise

Exercise - Create and deploy a contract with simple math functions.

Copy page

Each module in this course will contain exercises in which you are given a specification for a contract **without** being given specific instructions on how to build the contract. You must use what you've learned to figure out the best solution on your own!

> ⓘ Once you've learned how to deploy your contracts to a test network, you'll be given the opportunity to submit your contract address for review by an onchain unit test. If it passes, you'll receive an NFT pin recognizing your accomplishment.
>
> **You'll deploy and submit this contract in the next module.**

The following exercise asks you to create a contract that adheres to the following stated specifications.

Ask a question...   Ctrl+I

Contract

Create a contract called `BasicMath`. It should not inherit from any other contracts and does not need a constructor. It should have the following two functions:

Adder

A function called `adder`. It must:

Accept two `uint` arguments, called `_a` and `_b`

Return a `uint` `sum` and a `bool` `error`

If `_a + _b` does not overflow, it should return the `sum` and an `error` of `false`

If `_a + _b` overflows, it should return `0` as the `sum`, and an `error` of `true`

Subtractor

A function called `subtractor`. It must:

Accept two `uint` arguments, called `_a` and `_b`

Return a `uint` `difference` and a `bool` `error`

If `_a - _b` does not underflow, it should return the `difference` and an `error` of `false`

If `_a - _b` underflows, it should return `0` as the `difference`, and an `error` of `true`

Introduction to Contracts

Hello World Guide

Basic Types

Basic Functions Exercise

■ base docs   Base.org   Blog   Privacy Policy   Terms of Service   Cookie Policy