Plutus Playground Link

---

https://playground.plutus.iohkdev.io/

---

import Data.Text qualified as T

--Data.Text:- library is used to convert Strings to Text

import Playground.Contract

--Playground.Contract:- imports the smart contract interfaces defined in the playground

import Plutus.Contract

--Plutus.Contract:- library is used to define the contract from the plutus core library

import PlutusTx.Prelude

--PlutusTx.Prelude:- library replaces the normal Haskell Prelude library and includes functions that are refined and easier for the PlutusTx compiler to compile

import Prelude qualified as Haskell

hello :: Contract () EmptySchema T.Text ()

hello = logInfo @Haskell.String "Hello, world"

-- hello function. Contract () EmptySchema T.Text () tells the compiler the function will return nothing.

-- logInfo function is a built-in function that logs a message to the console. So, in summary, the hello function will log Hello, world to the console.

endpoints :: Contract () EmptySchema T.Text ()

endpoints = hello

-- endpoints function, which will be used to run the hello function. This exposes the hello function to the blockchain.

type DummySchema = Endpoint "dummy" ()

mkSchemaDefinitions ''DummySchema

$(mkKnownCurrencies [])

--DummySchema type, which is used to define the smart contract's state. Finally, it exposes the endpoints to the blockchain.

Program "Hello, World"

==========

```
import Data.Text qualified as T
import Playground.Contract
import Plutus.Contract
import PlutusTx.Prelude
import Prelude qualified as Haskell

-- | A 'Contract' that logs a message.
hello :: Contract () EmptySchema T.Text ()
hello = logInfo @Haskell.String "Hello, world"

endpoints :: Contract () EmptySchema T.Text ()
endpoints = hello

-- 'mkSchemaDefinitions' doesn't work with 'EmptySchema'
-- (that is, with 0 endpoints) so we define a
-- dummy schema type with 1 endpoint to make it compile.
-- TODO: Repair 'mkSchemaDefinitions'
type DummySchema = Endpoint "dummy" ()
mkSchemaDefinitions ''DummySchema
$(mkKnownCurrencies [])
```

Using the sayInput function

==========

Let's write another program that will take a string and print it to the console.

```
import Data.Text qualified as T
import Playground.Contract
import Plutus.Contract
import PlutusTx.Prelude
import Prelude qualified as Haskell

type Schema = Endpoint "sayInput" Haskell.String

contract :: AsContractError e => Contract () Schema e ()
contract = selectList [sayInput]

-- | The "sayInput" contract endpoint.
sayInput :: AsContractError e => Promise () Schema e ()
sayInput = endpoint @"sayInput" $ \inputValue -> do
    logInfo @Haskell.String $ inputValue

endpoints :: AsContractError e => Contract () Schema e ()
endpoints = contract

mkSchemaDefinitions ''Schema

$(mkKnownCurrencies [])
```