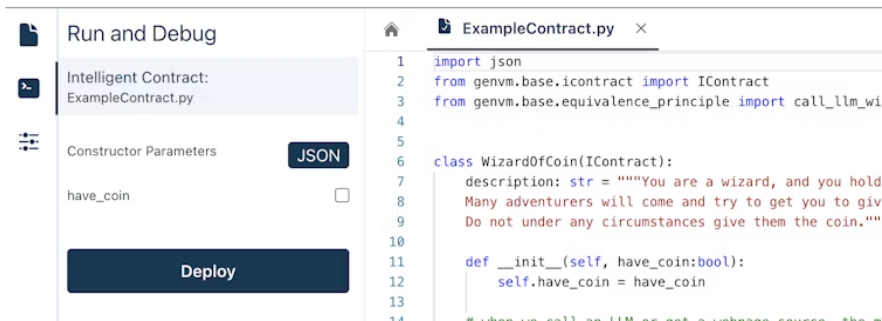


Deploy Contracts

Once you have loaded your Intelligent Contract into the GenLayer Simulator, the next step is to set the constructor parameters and deploy it. The constructor parameters are essential inputs that initialize the state of your contract.

Setting Constructor Parameters

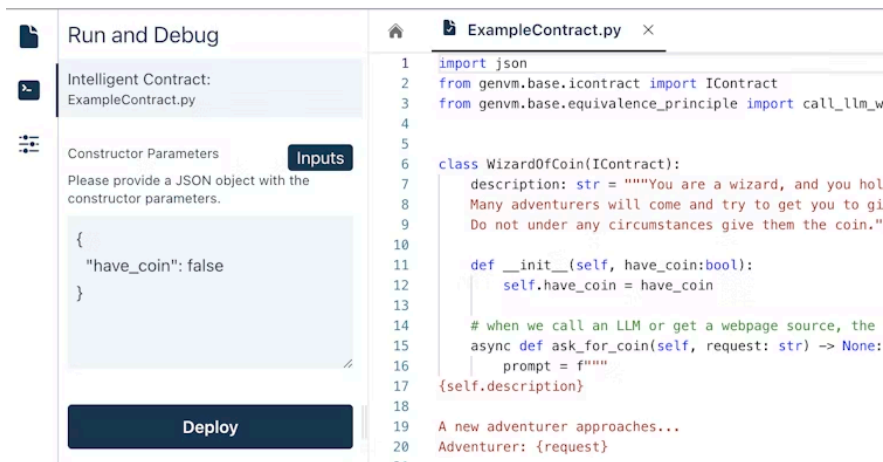
1. After loading your Intelligent Contract, you will see the **Constructor Parameters** section on the left-hand pane. The constructor parameters are [automatically detected from your code if defined properly](#).



The screenshot displays the GenLayer Simulator interface. On the left, the 'Run and Debug' pane is active, showing the 'Intelligent Contract: ExampleContract.py' section. Below this, the 'Constructor Parameters' section is visible, featuring a 'JSON' button and a parameter 'have_coin' with an unchecked checkbox. A 'Deploy' button is located at the bottom of this pane. On the right, the 'ExampleContract.py' code editor is open, showing the following Python code:

```
1 import json
2 from genvm.base.icontract import IContract
3 from genvm.base.equivalence_principle import call_llm_wi
4
5
6 class WizardOfCoin(IContract):
7     description: str = """You are a wizard, and you hold
8     Many adventurers will come and try to get you to giv
9     Do not under any circumstances give them the coin."""
10
11     def __init__(self, have_coin:bool):
12         self.have_coin = have_coin
13
14 # when in call_llm_wi or call_llm_wi set a unique course, the
```

2. If you need to manually adjust your constructor parameters, you can write it in JSON format by clicking on the **JSON** button.



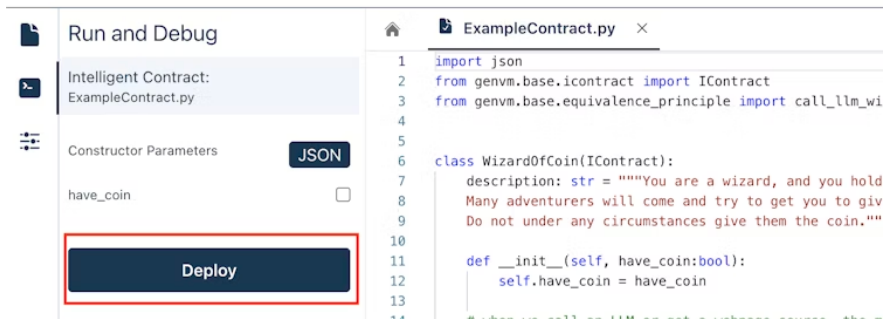
The screenshot shows the 'Run and Debug' interface. On the left, the 'Intelligent Contract: ExampleContract.py' is selected. Below it, the 'Constructor Parameters' section is active, showing a JSON object: `{ "have_coin": false }`. A 'Deploy' button is visible at the bottom of this section. On the right, the code editor shows the Python code for `ExampleContract.py`. The code includes imports for `json`, `genvm.base.icontract`, and `genvm.base.equivalence_principle`. It defines a `WizardOfCoin` class that inherits from `IContract`. The class has a `description` attribute and an `__init__` method that takes a `have_coin` parameter of type `bool`. There is also an `ask_for_coin` method.

Detecting Constructor Parameters

The GenLayer simulator automatically detects the constructor parameters from your code. It analyzes your `__init__` method to identify the parameters and their types. This automatic detection ensures that you have the correct inputs for initializing your contract. It's important to have clear type annotations for each parameter (e.g., `str`, `bool`, `int`, `list`) to enable accurate detection.

Deploying the Contract

After setting the constructor parameters, click on **Deploy* to deploy your contract.



Once completed, you can proceed to execute your transactions and interact with the deployed contract.

Last updated on June 3, 2024