**Web 3**

**What is the difference between ERC20, 721,777, 1155?**

ERC-20 introduces a standard for fungible tokens. That is, each token (type and value) has properties that make it exactly the same as another token. For example, ERC-20 tokens behave similarly to ETH. That is, one token is always equal to all other tokens.

ERC-721 introduces a standard for NFT. This means that this type of token is unique and can have a different value than another token of the same smart contract.

ERC-777 is a fungible token standard that improves on the existing ERC-20 standard.

ERC-1155 is a standard interface for contracts that manage multiple token types. A single deployed contract may contain any combination of fungible tokens, non-fungible tokens, or other constructs (e.g., semi-fungible tokens).

**Vyper**

1. **How to write secure smart contracts in vyper.**

### Step 1: Install Vyper

Download Python 3.6 or higher versions from Python’s official website.

The pip package manager is essential to install Vyper using the following command:

pip install vyper

### Step 2: Write the code

Let’s build a smart contract to store and display a user’s address.

Note: You will need to register with MetaMask to interact with Ethereum.

#### Vyper code to create a smart contract

Create a folder, vyper\_contract, and a file, contract.vy, inside the folder. Contract.vy is the Vyper code file to create the necessary Vyper smart contract.

Step 3: Compile the Vyper smart contract

This step checks to make sure the contract works as desired.

vyper<filename>.vy, or in this case, vypercontract.vy, is the command to compile the contract.

The output is a hexadecimal string that is the bytecode of the contract.

* Compile the contract remotely using Remix IDE. In the IDE, create a new file called contract.vy.
* Include the same Vyper code mentioned above in this remote file.
* Install the Vyper plugin in the Remix IDE. Click on the socket icon in the sidebar and type “vyper” to search for the same.
* Click Activate on the item that appears.
* Click on the Vyper icon and press the Compile contract.vy button.

### Step 4: Deploy the contract

Click the Ethereum icon in the sidebar.

From the ENVIRONMENT dropdown, select Injected Web3. A small UI beneath will change to the RopstenTestnet you are connected to.

Click on Deploy and input a word beside the Deploy button. This word is the user’s address - the desired input.

Clicking the Deploy button will make the MetaMask UI appear.

Click Confirm.

The Vyper smart contract will be deployed to the RopstenTestnet.

### Step 5: Test the deployed contract

Now, interact with the contract deployed in the previous step to call the methods in the Remix browser IDE or view the deployed contract on the Ropsten Ethereum Faucet.

Click on the CONTRACT AT 0x26… dropdown. You will see the functions and state variables, getAddress and address. Choose getAddress.

It will return the word passed to the constructor during deployment in step 4. The action called the getAddress function in the contract.

A Vyper smart contract that can store and display user addresses has successfully been tested and deployed! Try your hand at it by using the codes above. You can also try it with different variables.

1. **Why is vyper preferred over solidity in building smart contract?**

The following are the reasons why vyper is preferred over solidity in building smart contracts:

* Simple syntax
* Learning flexibility
* Contract definition and error handling is simple to do
* Ease o defining variables
* Withdrawal is easy
* Ending the contract is less cumbersome