**https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/**

**Solidity vs Vyper**

Updated on

Feb 28, 2023

[Vyper](https://www.quicknode.com/guides/tags/vyper/)

[Solidity](https://www.quicknode.com/guides/tags/solidity/)

*6 min read*

Overview[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#overview)

With the introduction to smart contracts on the Ethereum blockchain, it was only a matter of time until a language other than Solidity was made to write smart contract code. Vyper is one such language, looking to make smart contract source code easier to read. We have guides on both [Solidity](https://www.quicknode.com/guides/solidity/how-to-write-an-ethereum-smart-contract-using-solidity), and [Vyper](https://www.quicknode.com/guides/vyper/how-to-write-an-ethereum-smart-contract-using-vyper" \t "_blank), so this will not go over the basics in this guide. What we will do instead, is explore the pros and cons of each langauge.

**Prerequisites:**

* Basic understanding of Solidity.
* Basic understanding of Vyper

About Smart Contracts[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#about-smart-contracts)

When we think of blockchains, the first thing that comes to mind is the digital currency, and the second thing that comes to our mind is programmable software, i.e., smart contracts. Smart contracts were introduced by Ethereum, and other blockchains started rapidly appearing in the market, looking to implement this same functionality. Smart contracts enable the entire ecosystem of Ethereum. Without them, there would be no Aave, or UniSwap. There would be no DeFi at all. With a bit of context now, you may be wondering, what exactly is a smart contract?

According to the [Ethereum Docs:](https://ethereum.org/en/developers/docs/smart-contracts/)

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

In simple words, it is software that will live and run forever, and it will be performing the tasks in a deterministic, predictable manner. Before you get too excited, we need to talk about high-level and low-level languages. A high-level language is a human-readable language that helps us develop our code. It is reasonably easy to understand and parse what the code is going to accomplish. A low-level language is a machine-readable language that allows a machine to understand and execute code.

In the early days, developers wrote code in a low-level language. Now, this is not the best way to go about it, and it would take years for a new developer to understand and develop impactful software. That is why developers write in high-level languages. It allowed developers to write code in a human understandable format. But then, to make the machine understand the logic, we made the compiler. Compilers convert high-level language source code into machine-readable, low-level code.

Coming back to our previous question, you can write your smart contract in a low-level language called bytecode, which would take years for you to understand and months to implement. I know no one would opt for that, so we can go for the better solution, i.e., to use a high-level language. There are many languages that you can use to design your contract-

1. Solidity
2. Vyper
3. Yul
4. Yul+
5. FE

Here are some interesting facts to keep in mind-

1. Ethereum does not use a real machine to run your contract. It runs a virtual machine known as the Ethereum Virtual Machine(EVM) to execute the smart contract.
2. No matter which high-level language you go for, it will be compiled down to a machine-readable format such as bytecode.

While there are other languages you could a smart contract in, we will be focusing on Solidity and Vyper.

About Solidity[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#about-solidity)

Introduction[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#introduction)

If you have been in the blockchain space, you must have heard of Solidity. It is an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [high-level language](https://en.wikipedia.org/wiki/High-level_programming_language). It is strongly influenced by some of the existing high-level languages like JavaScript and C++. It is a statically typed language, meaning you will have to define the type of value you want to store in a variable so that the compiler knows what type of data to expect. This is essential when developing a deterministic application.

Code and Syntax[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#code-and-syntax)

Here you can see a simple contract written in Solidity.

*// SPDX-License-Identifier: GPL-3.0*  
  
pragma solidity >=0.7.0 <0.9.0;  
  
contract Promise {  
   
 string promiseMessage;  
   
 *//set function*  
 function setPromise( string memory \_promise) public{  
 promiseMessage=\_promise;  
 }  
 *//get function*  
 function getPromise()public view returns(string memory){  
 return promiseMessage;  
 }   
   
}

It's a simple smart contract that has a state variable named ***promiseMessage***. It has two functions -

1. ***setPromise*** sets the value for the state variable.
2. ***getPromise*** returns the value of the state variable.

If you have worked with JavaScript or C++ before, you may find the syntax somewhat familiar.

Pros[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#pros)

1. A lot of learning resources and tutorials are available on the internet.
2. It has vast community support. If you are stuck somewhere, there is a huge community that can help you resolve the issue.
3. Excellent developer tools are available. Some examples of them are -
   1. Remix: An online editor to write your smart contract.
   2. Truffle and Hardhat: Development suite to write, test and deploy your contract.
4. Many major players in the space such as Aave and UniSwap are written in Solidity.
5. It supports dynamic size arrays and strings. In the above code, we can vary the length of our string.
6. In the most recent solidity version [exception handling](https://en.wikipedia.org/wiki/Exception_handling) was added to the platform.

Cons[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#cons)

1. There are chances of overflow, i.e., the value to be stored is too large.
2. According to the survey [conducted in 2018](https://arxiv.org/pdf/1802.06038.pdf), many solidity contracts had a severe vulnerability that was missed during development/testing.

About Vyper[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/" \l "about-vyper" \o "Direct link to About Vyper)

Introduction[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#introduction-1)

Vyper is the second most popular choice to develop your contract after Solidity. A brief overview of Vyper according to [official docs](https://vyper.readthedocs.io/en/stable/):

Vyper is a contract-oriented, pythonic programming language that targets the Ethereum Virtual Machine (EVM).

It was specifically developed to address the security issues which were there in Solidity. It is strongly influenced by python, as you might have guessed from the definition. Unlike Solidity, Vyper has dropped some Object-Oriented concepts such as Inheritance, which is popularly referred to as contract-oriented or transactional programming. The main motive was to make contracts auditable and more secure, making less error-prone contracts. It is a strongly typed language, which means it does not allow you to use one type of data type as another. This ensures that there are no mishaps when dealing with various data types.

Code and Synatx[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/" \l "code-and-synatx" \o "Direct link to Code and Synatx)

Here, we are writing the previously shown contract using Vyper. For a deeper understanding, you can refer our guide on [how to write an Ethereum smart contract using Vyper](https://www.quicknode.com/guides/vyper/how-to-write-an-ethereum-smart-contract-using-vyper)

We are doing the same thing as we did with Solidity. I hope the Python developers are seeing some familiar syntax. Unlike in Solidity, here we define an upper limit on the string with 100 characters.

Fun Fact: Uniswap V1 was written in Vyper!

Pros[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#pros-1)

1. It is easier for Python developers to get started with it.
2. It aims to be transparent for security and readability purposes.
3. It has seen a rise in development tools such as:
   1. [Brownie:](https://vyper.readthedocs.io/en/stable/testing-contracts-brownie.html#:~:text=Brownie%20is%20a%20Python%2Dbased,overview%20of%20testing%20with%20Brownie.)😃 A Python-based development framework to test smart contracts.
   2. [Etherscan](https://etherscan.io/vyper) provides an online editor to compile your contracts.
4. DeFi protocols like Curve use it for developing their contracts.
5. There is a limitation on the string and array. It makes the contract less prone to attacks.
6. It is possible to compute a precise upper bound for any function call's gas consumption.

Cons[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#cons-1)

1. Community support is not quite there.
2. It lacks modifiers, class inheritance, recursive calls, and dynamic data types.
3. It is still under development. Many features are available in Solidity but not in Vyper.

Conclusion[​](https://www.quicknode.com/guides/ethereum-development/smart-contracts/solidity-vs-vyper/#conclusion)

Now, the million dollar question is, which one is better?

The main motive of developing Vyper was not to replace Solidity but to be used alongside it. Everyhing comes down to the developer. It is recommended to start with Solidity for beginners, but for secure contracts, it is good to use Vyper. Now, that is not to say that the inverse is not true as well. It all boils down to your personal preference. Each has its pros and cons. You can find a detailed list of syntactical differences [here](https://reference.auditless.com/cheatsheet/).

Having made it to the end of the guide, we hope you have learned a thing or two; and can make a more educated guess on which language to pick for your next Ethereum project!