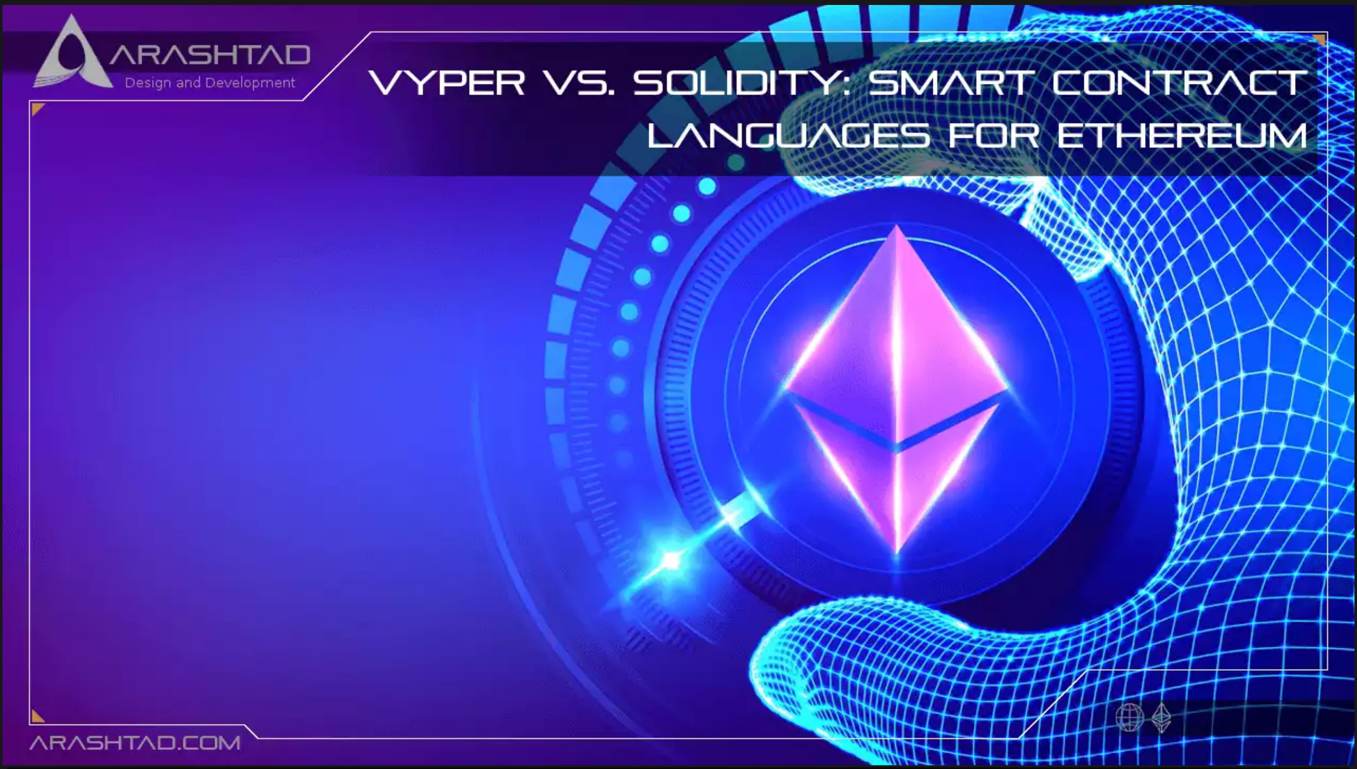
Vyper vs. Solidity: Smart Contract Languages for Ethereum

No comments



Smart contracts are the most critical aspects of the blockchain ecosystem. Smart contracts offer the ability to create self-executing code based on specific conditions. However, developers are constantly concerned with the Vyper vs. Solidity comparison and the similarities between their functions. In terms of creating smart contracts, which is a better choice? The older solution, Solidity, has been around for a long time. On the other hand, Vyper is relatively new but offers better performance for smart contract development. To find out which one is the best option, we will discuss the important differences between Solidity and Vyper.

What Is Smart Contracts?

Blockchain technology is characterized by smart contracts, which were introduced on Ethereum for the first time. Today, Ethereum supports the use of smart contracts in the largest ecosystem. It is important to note that without smart contracts, you wouldn't have popular apps like Uniswap or Aave. Essentially, smart contracts are programs that run forever on Ethereum. They collect data and code at a specific address on the Ethereum blockchain. They are also predictable and deterministic in executing the desired tasks. Vyper and Solidity differ in many ways, but understanding their differences only requires knowledge of high-level and low-level programming languages. High-level programming languages refer to human-readable languages that anyone can easily understand. It can help you figure out what the code can do for you.

In contrast, low-level programming languages or code are machine-readable languages that make it easier for machines to understand and execute code. Low-level code can be easier for machines to understand, whereas high-level code is easier for humans to comprehend. It is interesting to note that the compiler can bridge the gap by converting high-level code into low-level code. Since creating low-level bytecodes for smart contracts could take around months, high-level languages are better for designing smart contracts.

Smart Contract Programming Languages

Developers need to know the best programming languages to develop smart contracts since smart contracts are an integral part of the emerging crypto ecosystem. Understanding the differences between Ethereum smart contract languages could help you choose between Solidity and Vyper. Yul, FE, and Yul+ are important languages for smart contract development. Despite the numerous options available among smart contract programming languages, Solidity and Vyper have marked a significant influence. To choose the right one, let us learn the basics of both smart contract programming languages.

What is Vyper?

A contract-oriented, pythonic programming language for smart contracts that are intended to work with the EVM, Vyper is designed to improve Solidity in terms of security and readability. In addition to its emphasis on audibility, one of its principles is that the Vyper code should be as easy to read as possible. A key goal of Vyper is to make it harder for anyone to write misleading code. Simplicity for the reader (i.e., the auditor) is more important than simplicity for the writer (i.e., the developer). Thus, malicious code within smart contracts or decentralized applications (DApps) will be easier to identify).

What is Solidity?

The Solidity programming language was developed in 2014 and shares similarities with programming languages like JavaScript, C++, and Python. It is an object-oriented, high-level programming language for creating smart contracts for Ethereum, BNB Smart Chain, and Avalanche. In Solidity, computer code is typed in a way that is easily readable and understandable instead of ones and zeros. For example, Solidity code will include words and phrases like "function" and "contract," along with curly brackets. Because Solidity is an object-oriented coding language, it uses "objects," which are pieces of code that can be reused to create similar pieces of code without having to rewrite the original. In addition to the compiler, Solidity executes the human-readable, high-level code on the Ethereum Virtual Machine (EVM).

Vyper vs Solidity Differences

The following are some of the most notable differences between Solidity and Vyper for developers of smart contracts.

Image not found or type unknown Solidity VS Vyper Vyper vs. Solidity: Smart Contract Languages for Ethereum

1- Flexible Learning

At present, Solidity stands out as one of Ethereum's most popular smart contract programming languages. Therefore, you can access several learning resources and tools for beginners on Solidity. Vyper is a relatively new language and does not have extensive documentation as Solidity does. Although developers can learn about Vyper with basic Python skills, it has a lighter and simpler design that makes auditing easier.

2- Syntax Ease

Solidity and Vyper are high-level languages, so their syntax is quite straightforward and easy to understand. Ethereum smart contract languages differ considerably in syntax, which can make it quite confusing. Despite this, Solidity shines because of its simple syntax, which is similar to C++ and JavaScript, two of the most commonly used programming languages. Vyper, on the other hand, is as easy to use as Solidity. Due to the similarities between Vyper and Python, anyone with Python knowledge can easily grasp Vyper syntax.

3- Community Support

As a result of Solidity's extensive use, a large community of developers, enthusiasts, experts, and professionals has developed. Therefore, Solidity developers can take advantage of this massive community's backing. In contrast, Vyper is still undergoing development, as are its community members. Vyper has limited functionalities compared to Solidity and is aimed at a small group of developers. Vyper, like Solidity, is less likely to receive community support in the future.

4- Definition of Contracts and Error Handling

It is possible to define Solidity licenses and versions for a contract using Brownie and VSCode. However, only the version of Vyper that you use for determining a contract must be listed. Solidity also appears to have a bit diminished power due to the requirement to compile a contract before identifying typos. Vyper offers a credible advantage with its ability to immediately identify even the slightest typo. As a result, Vyper simplifies contract definition and improves debugging efficiency.

5- Creating Auctions

Creating auctions in Vyper should be straightforward, and an external decorator would help make it possible to call it from other contracts. You can also specify all the other elements according to your needs. In contrast to Solidity errors, Vyper asserts itself as the reliable alternative. The auction will remain open for bidding as long as the 'block.timestamp' value remains lower than the auction run time.

6- Withdrawal

The Solidity vs. Vyper debate is evident in the simplicity of the withdraw function in Vyper. Solidity lets you set the amount along with the hassle of writing 'if' statements. Vyper keeps everything streamlined.

7- Ending the Contract

The final stage in developing smart contracts also specifies differences between Ethereum's smart contract languages. Vyper requires to assert to verify that time has run out. When using Solidity, you will have to write 'if' statements before switching the variable to "True." At the same time, you should emit the auction's end and transfer funds rather than send funds in Vyper.

Final Words

As a result of the differences between Vyper and Solidity, Vyper stands out as a favorable alternative. It is much simpler than Solidity, especially in terms of simplicity. If you are familiar with Python, Vyper is easier to understand. However, Solidity has a strong community and access to tools and resources for developers. This means that the smart contract programming language that is most appropriate for you will depend on your use case. With the rise of blockchain-based applications and smart contracts, Solidity and Vyper will become increasingly important.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.

Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP NEWSLETTER RSS FEED