

# Vyper: The Slick new Ethereum Language!

## Programming Language Survey

Presented by G H Hem Sagar

CB.EN.U4CYS21016

TIFAC-CORE in Cyber Security

Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 26, 2023



**AMRITA**  
VISHWA VIDYAPEETHAM



- 1 What is Vyper?
- 2 Why is Vyper required?
- 3 Getting started with Vyper!
- 4 Conclusion



# VYPER

- **Vyper** is a **secure, open source** programming language for writing Ethereum smart contracts.
- It focuses on the **Ethereum Virtual Machine** or **EVM**, which is a simulated global computer running in parallel to the Ethereum blockchain.
- Vyper was created by **Vitalik Buterin**, the founder of **Ethereum**, in 2014. He was inspired by Python and wanted to create a language that was more suitable for writing smart contracts and decentralized applications.
- Vyper is a pythonic programming language. It is strongly typed and belongs to contract - oriented programming paradigm.



## Principles and Goals:

- *Security* - Vyper emphasizes on ensuring natural abilities for security of smart contracts.
- *Simplicity* - Vyper focuses on simplicity of the language and compiler.
- *Auditability* - Vyper strives the code to be maximally human-readable. It believes "Simplicity for the reader is more important than simplicity for the writer."

## Features Vyper Offers:

- Bounds and overflow checking: On array accesses and arithmetic.
- Support for signed integers and decimal fixed point numbers
- Decidability: It is possible to compute a precise upper bound for the gas consumption of any Vyper function call.
- Small and understandable compiler code.
- Limited support for pure functions: Anything marked constant is not allowed to change the state.



# Disadvantages? nah Precaution for security!

The limited feature set of Vyper is one of the foremost highlights on the programming language.

A limited set of features ensures better security and auditability of smart contract development. However, it is also important to learn about the features which Vyper does not include.

## Features Excluded:

- Modifiers
- Class inheritance
- Inline assembly
- Function overloading
- Operator overloading
- Recursive calling
- Infinite-length loops
- Binary fixed point



# Why is Vyper important?

- The extended set of features in **Solidity** and its flexibility have been called to question for vulnerabilities. **For example**, smart contracts with bugs and malicious design patterns can be one of the formidable concerns on Solidity.
- However, **Vyper** comes with a unique design for avoiding malicious design patterns.
- Though it does not guarantee complete immunity from suspicious design patterns, Vyper can make sure that it is impossible to get most of them through the verification

## Trivia

Vyper does not strive to be a 100% replacement for everything that can be done in Solidity; it will deliberately forbid things or make things harder if it deems fit to do so for the goal of increasing security.

- It provides a simple, intuitive, and powerful way to create decentralized applications (DApps) on the Ethereum platform.
- By offering a simpler, more secure coding language, Vyper makes it easier to create secure, reliable smart contracts on the Ethereum network.



# Getting Started with Vyper

## 1 Installing Python:

- Vyper can be installed using Python.

### Note:

Vyper can only be built using Python 3.6 and higher.

## 2 Creating an Virtual Environment:

- It is strongly recommended to install Vyper in a virtual Python environment, so that new packages installed and dependencies built are strictly contained in your Vyper project and will not alter or affect your other development environment set-up.
- Pyenv or Poetry are recommended environments.

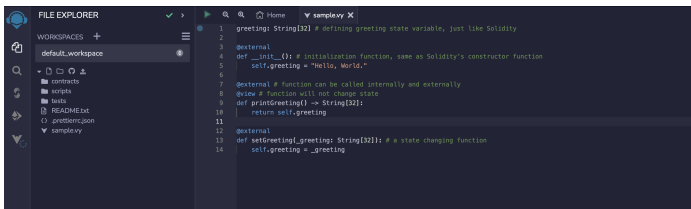
## 3 Installing Vyper:

- Each tagged version of vyper is uploaded to *pypi*, and can be installed using **pip**.
- **Command:** `pip install vyper`



# Sample Program!

**Remix IDE** can be used to compile the code online and it also provides facility to test deploy the contract.



The screenshot shows the Remix IDE interface. On the left is the 'FILE EXPLORER' panel with a 'WORKSPACES' section containing 'default.workspace'. Below it is a file tree with folders 'contracts', 'scripts', and 'tests', and files 'README.txt', '.prettierrc.json', and 'sample.vy'. The main editor area displays a Vyper program named 'sample.vy' with the following code:

```
1 greeting: String[32] # defining greeting state variable, just like Solidity
2
3 @external
4 def __init__(): # initialization function, same as Solidity's constructor function
5     self.greeting = "Hello, World."
6
7 @external # function can be called internally and externally
8 @view # function will not change state
9 def printGreeting() -> String[32]:
10     return self.greeting
11
12 @external
13 def setGreeting(_greeting: String[32]): # a state changing function
14     self.greeting = _greeting
```

**Figure:** Remix IDE online

After compilation,

- The *bytecode* generated by the vyper compiler can be taken and the contract is deployed using mist or geth
- Or Remote compiler such as **RemixIDE** can be used to compile and can deploy the contract on any net of choice.





Thank you!

**Thank You!!!**



- Vyper Documentation
- A Beginners guide to Vyper
- Understanding Vyper
- Remix IDE online

