**Vyper**

Vyper is a contract-oriented, pythonic programming language that targets the [Ethereum Virtual Machine (EVM)](https://ethereum.org/learn/#ethereum-basics).

**Principles and Goals**

* **Security**: It should be possible and natural to build secure smart-contracts in Vyper.
* **Language and compiler simplicity**: The language and the compiler implementation should strive to be simple.
* **Auditability**: Vyper code should be maximally human-readable. Furthermore, it should be maximally difficult to write misleading code. Simplicity for the reader is more important than simplicity for the writer, and simplicity for readers with low prior experience with Vyper (and low prior experience with programming in general) is particularly important.

Because of this Vyper provides the following features:

* **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.
* **Strong typing**
* **Small and understandable compiler code**
* **Limited support for pure functions**: Anything marked constant is not allowed to change the state.

Following the principles and goals, Vyper **does not** provide the following features:

* **Modifiers**: For example in Solidity you can define a function foo() mod1 { ... }, where mod1 can be defined elsewhere in the code to include a check that is done before execution, a check that is done after execution, some state changes, or possibly other things. Vyper does not have this, because it makes it too easy to write misleading code. mod1 just looks too innocuous for something that could add arbitrary pre-conditions, post-conditions or state changes. Also, it encourages people to write code where the execution jumps around the file, harming auditability. The usual use case for a modifier is something that performs a single check before execution of a program; our recommendation is to simply inline these checks as asserts.
* **Class inheritance**: Class inheritance requires people to jump between multiple files to understand what a program is doing, and requires people to understand the rules of precedence in case of conflicts (“Which class’s function X is the one that’s actually used?”). Hence, it makes code too complicated to understand which negatively impacts auditability.
* **Inline assembly**: Adding inline assembly would make it no longer possible to search for a variable name in order to find all instances where that variable is read or modified.
* **Function overloading**: This can cause lots of confusion on which function is called at any given time. Thus it’s easier to write missleading code (foo("hello") logs “hello” but foo("hello", "world") steals your funds). Another problem with function overloading is that it makes the code much harder to search through as you have to keep track on which call refers to which function.
* **Operator overloading**: Operator overloading makes writing misleading code possible. For example + could be overloaded so that it executes commands that are not visible at a first glance, such as sending funds the user did not want to send.
* **Recursive calling**: Recursive calling makes it impossible to set an upper bound on gas limits, opening the door for gas limit attacks.
* **Infinite-length loops**: Similar to recursive calling, infinite-length loops make it impossible to set an upper bound on gas limits, opening the door for gas limit attacks.
* **Binary fixed point**: Decimal fixed point is better, because any decimal fixed point value written as a literal in code has an exact representation, whereas with binary fixed point approximations are often required (e.g. (0.2)10 = (0.001100110011…)2, which needs to be truncated), leading to unintuitive results, e.g. in Python 0.3 + 0.3 + 0.3 + 0.1 != 1.

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.

# Installing Vyper

Take a deep breath, follow the instructions, and please [create an issue](https://github.com/vyperlang/vyper/issues) if you encounter any errors.

**Note**

The easiest way to experiment with the language is to use the [Remix online compiler](https://remix.ethereum.org/). (Activate the vyper-remix plugin in the Plugin manager.)

## Docker

Vyper can be downloaded as docker image from [dockerhub](https://hub.docker.com/r/vyperlang/vyper/tags?page=1&ordering=last_updated):

docker pull vyperlang/vyper

To run the compiler use the docker run command:

docker run -v $(pwd):/code vyperlang/vyper /code/<contract\_file.vy>

Alternatively you can log into the docker image and execute vyper on the prompt.

docker run -v $(pwd):/code/ -it --entrypoint /bin/bash vyperlang/vyper

root@d35252d1fb1b:/code# vyper <contract\_file.vy>

The normal parameters are also supported, for example:

docker run -v $(pwd):/code vyperlang/vyper -f abi /code/<contract\_file.vy>

[{'name': 'test1', 'outputs': [], 'inputs': [{'type': 'uint256', 'name': 'a'}, {'type': 'bytes', 'name': 'b'}], 'constant': False, 'payable': False, 'type': 'function', 'gas': 441}, {'name': 'test2', 'outputs': [], 'inputs': [{'type': 'uint256', 'name': 'a'}], 'constant': False, 'payable': False, 'type': 'function', 'gas': 316}]

**Note**

If you would like to know how to install Docker, please follow their [documentation](https://docs.docker.com/get-docker/).

## PIP

### Installing Python

Vyper can only be built using Python 3.6 and higher. If you need to know how to install the correct version of python, follow the instructions from the official [Python website](https://wiki.python.org/moin/BeginnersGuide/Download).

### Creating a 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. For easy virtualenv management, we recommend either [pyenv](https://github.com/pyenv/pyenv) or [Poetry](https://github.com/python-poetry/poetry).

**Note**

To find out more about virtual environments, check out: [virtualenv guide](https://docs.python.org/3/library/venv.html).

### Installing Vyper

Each tagged version of vyper is uploaded to [pypi](https://pypi.org/project/vyper/), and can be installed using pip:

pip install vyper

To install a specific version use:

pip install vyper==0.2.15

## nix

View the versions supported through nix at [nix package search](https://search.nixos.org/packages?channel=21.05&show=vyper&from=0&size=50&sort=relevance&query=vyper)

**Note**

The derivation for Vyper is located at [nixpkgs](https://github.com/NixOS/nixpkgs/blob/master/pkgs/development/compilers/vyper/default.nix)

### Installing Vyper

nix-env -iA nixpkgs.vyper