Vyper

A new python-like smart contract language, now in Beta!

https://github.com/ethereum/vyper
https://github.com/status-im/vyper-debug
https://vyper.readthedocs.io
https://remix.vyper.live

@jacqueswww | Jacques Wagener

October 2018

Vyper

Seat belts included.

History

- Vyper (or Viper) was started when Serpent was abandoned.
- Serpent had various issues that Vyper addresses.
- Some examples of the list of issues with Serpent are the following:*
  - Too low level language (such as LLL)
  - Lack of type checking
  - Lack of overflow checks and safety measures
- Vyper is a python-like language (uses Python 3.6 type annotations).

*https://blog.zeppelin.solutions/serpent-compiler-audit-3095d1257929

Vyper Features

- Auditing and readability are prioritised over code-reuse and programming ease
  - No operator or function overloading
  - Writing misleading code should be difficult
- No modifiers, as they can hide their true intent
  - Often just a single if statement
  - Forces jumping around when reading a contract
- Uses finite data structures to prevent gas limitation attacks
  - Only unlimited length data structure is the map type
- Infinite loops [e.g. range(<variable>)] are disallowed
- Force visibility of function to be set (1st Parity Multisignature Hack 2017)
- Try to minimise the possibility of reentrancy (DAO Hack 2016)
  - Recursion disabled on internal calls completely
  - Tries to disable it with an external call (but can be chained to bypass the check).

Vyper Features

- Decimal fixed-point arithmetic and storage
  - Python devs know to use “from decimal import Decimal”
  - For a financial system, fixed-point decimal makes sense
  - Being able to take any written literal and get an exact representation (no 0.9999 occurrences)
- Overflow protection with clamps (ie. SafeMath.sol is built in)
  - Overflow can cause all sorts of hell
- No in-line assembly
  - Serious way to hide intent as well as outcome
  - Can’t read the state of a contract easily with inline assembly (variables, memory, stack state is lost)
- No inheritance
  - Inheritance can hide method or functionality from an auditor
  - Encouraged to write everything in one file
  - Composition over inheritance for external replaceable code

Types

- All types in vyper use 32 bytes base word
    - Uint256
        - Unsigned integer
    - Int128
        - Signed integer
    - Byte array
        - Store the length of allocated bytes
        - Annotates the max length of bytes
    - Bytes32
        - Fixed size, pads right with zero bytes
        - Max size of 32 bytes
    - Address
        - 32 bytes in memory
        - 20 bytes actually used

```
@public
def test() -> bytes[132]:
    a: uint256 = 1
    b: int128 = -3
    c: bytes[100] = "hello world"
    d: bytes32 = convert("hello galaxy", bytes32)

    return concat(d, c)
```

Maps

- Maps
  - Key value lookup
  - Only available as a global
  - Has infinite length

```
registry: address[bytes[100]]

@public
def register(name: bytes[100], owner: address):
    assert self.registry[name] == ZERO_ADDRESS, "already set!"
    self.registry[name] = owner

@public
@constant
def lookup(name: bytes[100]) -> address:
    return self.registry[name]
```

Types

- Lists
  - Only allow single type to be stored in a list (unlike python)

```
@public
def foo():
    x: int128[3] = [1, 2, 3]
    x = [4, 5, 6]
    x[1] = 1
```

Constants

Custom constants
- Uses constant()
- Defined in global scope only

Built In constants
- Commonly used constants
  - ZERO_ADDRESS
  - MAX_INT128
  - MIN_INT128
  - MAX_DECIMAL
  - MIN_DECIMAL
  - MAX_UINT256

```
# Custom constant
OWNER: constant(address) =
0x0000000000000000000000000000000000000012

@public
def get_owner() -> address:
    return OWNER

# builtin constant
@public
def test_zaddress(a: address) -> bool:
    return a == ZERO_ADDRESS
```

Assertions

- **Assert**
  - Fails if not condition is not met
  - User REVERT opcode
- **assert** with a message / reason
  - Tell the world why it failed!
  - Cost ~110 bytes per message on contract creation (see EIP838)


```
@public
def foo():
    assert 1 == 2
```

```
@public
def test(a: int128) -> int128:
    assert a > 1, "larger than one please"
    return 1 + a
```

Units

- Unit system allows you much finer control of ensuring you are working with the correct type
- Units are something we are familiar with, and reads quite well
- No conversion between units… yet!

```
units: {
    cm: "centimeter",
    km: "kilometer"
}

a: int128(cm) # global storage

@public
def test() -> int128(km):
    b: int128(km)
    b = 100
    return b
```

Vyper Functions

Function visibility

- Either private or public
- @private
    - Can only be called from within current contract
    - Equivalent of solidity's *internal*
- @public
    - Can be called from another contract or address

Constant decorator

- @constant
    - Can not alter storage of the contract

Payable decorator

- @payable
    - Function can accept payment in the form of eth.

Built in methods

- Constructor **__init__**
  - Only gets run on create of the contract
  - Constructor code is not stored on chain

- Default function **__default__**
  - Defaults to REVERT / transaction failure
  - If added to contract, allows unspecified
  - Allow receiving of funds with @payable

```
Sent: event({sender: indexed(address)})

@public
@payable
def __default__():
    log.Sent(msg.sender
```

Builtin functions

- convert
- as_wei
- concat
- send
- as_unitless_number
- RLPList
- Math ones
    - floor / ceil
- Crypto
    - keccak256
    - ecrecover
    - ecmul
    - Ecadd
- https://vyper.readthedocs.io/en/latest/built-in-functions.html

Events

- *event* keyword to describe event format
- *log* to dispatch an event

```
Assigned: event({variable: int128})

@public
def foo(i: int128) -> int128:
    variable : int128 = i
    log.Assigned(variable)
    return variable
```

Contract Calls

- Use *self* to call another method
- External contract calls uses the *contract* keyword
- Each method specifies either
  - *modifying* (CALL)
    - Can modify storage
  - *constant* (STATICCALL)
    - Will not modify storage

```
class Foo():
        def foo(arg1: int128) -> int128: constant

@public
def bar(arg1: address, arg2: int128) -> int128:
    return Foo(arg1).foo(arg2)



class Bar():
    def foo() -> int128: modifying
    def array() -> bytes[3]: constant

@public
def bar(arg1: address) -> int128:
    return Bar(arg1).foo()
```

Let’s look at some code

```
registry: address[bytes[100]]


@public
def register(name: bytes[100], owner: address):
    assert self.registry[name] == ZERO_ADDRESS  # check name has not been set yet.
    self.registry[name] = owner


@public
@constant
def lookup(name: bytes[100]) -> address:
    return self.registry[name]
```

Let’s look at some code

Tooling

Getting there!

- Syntax highlighting for popular editors
  - Vim, emacs, sublime, atom, visual code
- Framework integrations
  - Truffle, embark, populus

Vyper Debugger

- Attempt to resemble the python debugger as close as possible
- (Should be) easy to pip install and run on a contract

Vyper Remix plugin

- https://remix.vyper.live/
- Shout out to @nrryuya !

Vyper Debugger

<DEMO>

- Better transaction failure
- Inspect scope

remix.vyper.live

<DEMO>

- Compile & Deploy to javascript VM
- Remote & Local compile

Python tests

- Roll your own *conftest.py* (or copy).
    See example conftest in the workshop repo.

- Pytest ethereum
    - https://github.com/fubuloubu/pytest-ethereum

- Predecessor to populus: twig
    - https://github.com/ethereum/twig
-

Other frameworks

- Embark
- Truffle (truper)
- All listed at https://github.com/ethereum/vyper/wiki/Vyper-tools-and-resources

Try it: Raffle example

https://github.com/jacqueswww/devcon4-workshop

Come and join us!

- Make some tools!
  - Opportunity to start from the ground up
- Opinions on code styling?
- Learn a new language by writing some tutorials
- https://github.com/ethereum/vyper/wiki/Vyper-tools-and-resources
- Head over to our repo and come look at the “discussion” issues.

Questions?

Contributions: https://github.com/ethereum/vyper
Chatter: https://gitter.im/ethereum/vyper

October 2018