# fuckery

*Release 0.6.0*

**Jan 22, 2023**

# Contents


**1 Overview** **1**
- 1.1 Installation . . . 1
- 1.2 Documentation . . . 1
- 1.3 Development . . . 2

**2 Installation** **3**

**3 Usage** **5**

**4 Reference** **7**
- 4.1 fuckery package . . . 7

**5 Contributing** **13**
- 5.1 Bug reports . . . 13
- 5.2 Documentation improvements . . . 13
- 5.3 Feature requests and feedback . . . 13
- 5.4 Development . . . 14

**6 Authors** **17**

**7 Changelog** **19**
- 7.1 0.6.0 (2023-01-22) . . . 19
- 7.2 0.5.3 (2020-08-19) . . . 19
- 7.3 0.5.2 (2020-08-19) . . . 19
- 7.4 0.5.1 (2020-08-19) . . . 19
- 7.5 0.5.0 (2020-03-13) . . . 19
- 7.6 0.4.1 (2018-04-28) . . . 20
- 7.7 0.4.0 (2018-04-28) . . . 20
- 7.8 0.3.9 (2018-04-28) . . . 20
- 7.9 0.3.8 (2018-04-03) . . . 20
- 7.10 0.3.7 (2018-03-26) . . . 20
- 7.11 0.3.6 (2018-03-26) . . . 20
- 7.12 0.3.5 (2018-03-26) . . . 20
- 7.13 0.3.4 (2018-03-26) . . . 20
- 7.14 0.3.3 (2018-03-26) . . . 20
- 7.15 0.3.2 (2018-03-26) . . . 21
- 7.16 0.3.1 (2018-03-26) . . . 21

7.17 0.3.0 (2018-03-26) . . . . . 21
7.18 0.2.3 (2017-03-27) . . . . . 21
7.19 0.2.2 (2017-03-01) . . . . . 21
7.20 0.2.1 (2017-03-01) . . . . . 21
7.21 0.2.0 (2017-03-01) . . . . . 21
7.22 0.1.0 (2017-02-12) . . . . . 21

**8 Indices and tables** **23**

**Python Module Index** **25**

**Index** **27**

# CHAPTER 1

# Overview

| | |
|---|---|
| docs | |
| tests | |
| package | **\|downloads\|** |

Python Brainfuck implemention.

- Free software: BSD license

## 1.1 Installation

```
pip install fuckery
```

## 1.2 Documentation

https://pyFuckery.readthedocs.io/

## 1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

| | |
|---|---|
| Windows | `set PYTEST_ADDOPTS=--cov-append`<br>`tox` |
| Other | `PYTEST_ADDOPTS=--cov-append tox` |

CHAPTER 2

# Installation

At the command line:

```
pip install fuckery
```

CHAPTER 3

# Usage

To use pyFuckery in a project simply import the package, instantiate a VM and execute your brainfuck program:

```
import fuckery
vm = fuckery.vm.VirtualMachine()
program = '++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..+++.>>.<-
↪.<.+++.------.--------.>>+.>++.'
vm.parse_and_run(program)
```

You can also swap out the input / output streams used by the VM, if stdin / stderr are not appropriate for your application:

```
import fuckery
import io
outstream = io.StringIO()
vm = fuckery.vm.VirtualMachine()
vm.stream_out = outstream
program = '++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..+++.>>.<-
↪.<.+++.------.--------.>>+.>++.'
vm.parse_and_run(program)
vm.out_stream.seek(0)
print(vm.out_stream.read())
```

CHAPTER 4

# Reference

## 4.1 fuckery package

### 4.1.1 Submodules

#### fuckery.cli module

Module that contains the command line app.

Why does this file exist, and why not put this in __main__?

> You might be tempted to import things from __main__ later, but that will cause problems: the code will get executed twice:
>
> - When you run *python -mfuckery* python will execute `__main__.py` as a script. That means there won't be any `fuckery.__main__` in `sys.modules`.
> - When you import __main__ it will get executed again (as a module) because there's no `fuckery.__main__` in `sys.modules`.
>
> Also see (1) from http://click.pocoo.org/5/setuptools/#setuptools-integration

`fuckery.cli.`**`main`**(*options*)

`fuckery.cli.`**`makeargpaser`**()

#### fuckery.constants module

pyFuckery - constants.py Created on 2/12/17.

Constants used by fuckery.

`fuckery.constants.`**`DEFAULT_MEMORY_SIZE = 30000`**

Default memotry sizen

`fuckery.constants.SYM_DATA_DEC = '-'`
Brainfuck DATA INC token

`fuckery.constants.SYM_DATA_INC = '+'`
Brainfuck DATA INC token

`fuckery.constants.SYM_IO_INPUT = ','`
Braifuck IO input token

`fuckery.constants.SYM_IO_OUTPUT = '.'`
Brainfuck IO output token

`fuckery.constants.SYM_JMP_BACKWARD = ']'`
Brainfuck jump backward token

`fuckery.constants.SYM_JMP_FWD = '['`
Brainfuck jump forward token

`fuckery.constants.SYM_PTR_DEC = '<'`
Brainfuck PTR DEC token

`fuckery.constants.SYM_PTR_INC = '>'`
Brainfuck tokens Brainfuck PTR INC token

### fuckery.exc module

pyFuckery - exc.py Created on 2/12/17.

Exception definitions

**exception** `fuckery.exc.AddressError`
Bases: *fuckery.exc.StorageError*

Error related to address violations.

**exception** `fuckery.exc.ExitCondition`
Bases: *fuckery.exc.VMError*

Error raised during a exit condition.

**exception** `fuckery.exc.FuckeryError`
Bases: `Exception`

Base exception for pyFuckery errors.

**exception** `fuckery.exc.StorageError`
Bases: *fuckery.exc.FuckeryError*

Error doing a memory operation.

**exception** `fuckery.exc.VMError`
Bases: *fuckery.exc.FuckeryError*

Error related to the brainfuck VM

### fuckery.memory module

pyFuckery - memory.py Created on 2/12/17.

Memory object implementation. Provides memory bounds checking, as well as value enforcement.

**class** fuckery.memory.**Storage**(*n=30000*)

Bases: object

Provides an interface for storing memory values for the Brainfuck VM.

This provides for type safety & memory access checking.

Init function for Storage.

**Parameters** **n** – Number of memory cells to create.

**get**(*addr*)

Get the value of the memory at a location.

**Parameters** **addr** – Memory address to retrieve.

**Returns**

**mem_hash**

Returns a hash of the state of the memory.

Note - Computing this frequently can be expensive to do as the memory section is serialized via msgpack.dumps() prior to hashing.

**Returns**

**set**(*addr*, *value*)

Set the value of the memory at a locaiton.

**Parameters**

- **addr** – Memory address to set.
- **value** – Value to set.

**Returns**

fuckery.memory.**main**(*options*)

fuckery.memory.**makeargpaser**()

### fuckery.parser module

pyFuckery - parser.py Created on 2/19/17.

fuckery.parser.**main**(*options*)

fuckery.parser.**makeargpaser**()

fuckery.parser.**parse_program**(*s: str*) → lark.tree.Tree

Parser a program to generate the lark parse Tree.

**Parameters** **s** – Brainfuck program to parse.

**Returns**

### fuckery.vm module

pyFuckery - vm.py Created on 2/12/17.

VM Definition to execute brainfuck programs which have been parsed into lark.Tree objects.

**class** `fuckery.vm.`**`VirtualMachine`**(*memory_size: int = 30000*, *loop_detection: bool = False*)

Bases: `object`

This is the brainfuck VM. You can drop this into programs that need a brainfuck VM, such as a module you don't want a coworker to ever easily maintain, or a really cruel programming based game.

Init function for the VM.

**Parameters**

- **`memory_size`** – Number of memory cells to instantiate.
- **`loop_detection`** – Enables loop detection if this evaluates to True. This is very costly from a

computation perspective, so use it wisely.

**`current_value`**

Property which represents the data value the current memory address points too.

**Returns**

**`dec_data_ptr`**() → None

Decrements the data pointer by 1.

**Returns** None

**`dec_data_value`**() → None

Decrements the value pointed to by the data pointer by 1. This wraps at zero, back to 255.

**Returns** None

**`inc_data_ptr`**() → None

Increments the data pointer by 1.

**Returns** None

**`inc_data_value`**() → None

Increments the value pointed to by the data pointer by 1. This wraps at 255, back to zero.

**Returns** None

**`io_input`**() → None

Reads a single character from self.stream_in.

If self.stream_in is sys.stdin (default value), it will prompt the user for a string and record the FIRST byte of that string. Otherwise, it will attempt to read 1 byte from the stream_in buffer.

Empty inputs have no effect on the state of the system.

**Returns** None

**`io_output`**() → None

Writes the current value, after casting it via chr(), to self.stream_out.

**Returns** None

**`parse_and_run`**(*program: str*) → None

Parse and run a brainfuck program.

**Parameters** **`program`** – String representing a brainfuck program.

**Returns** None

**`run`**(*tree: lark.tree.Tree*) → None

Walk a Brainfuck AST and execute the program contained in the AST.

This function is recursive, so its possible for a deeply nested program to hit the Python interpreter recursion limit, but if your brainfuck does that, kudos to you.

**Parameters** **tree** – Parsed brainfuck program.

**Returns**

**state_hash**

MD5 representing the state of the system. It is a hash of the memory and the current data pointer.

Note - Computing this frequently can be expensive to do with a large memory section, as the memory section is serialized via msgpack.dumps() prior to hashing.

**Returns**

fuckery.vm.**main**(*options*)

fuckery.vm.**makeargpaser**()

CHAPTER 5

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

## 5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

## 5.2 Documentation improvements

fuckery could always use more documentation, whether as part of the official fuckery docs, in docstrings, or even on the web in blog posts, articles, and such.

## 5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/williamgibb/pyFuckery/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

## 5.4 Development

To set up *fuckery* for local development:

1. Fork pyFuckery (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/pyFuckery.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

### 5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

### 5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run a specific test, with coverage reporting, you can also use the packaged testrunner.sh script.

./testrunner.sh test_myfeature

To run all the test environments in *parallel* (you need to `pip install detox`):

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

```
detox
```

# CHAPTER 6

# Authors

- William Gibb - https://github.com/williamgibb/

CHAPTER 7

# Changelog

## 7.1 0.6.0 (2023-01-22)

- Drop 3.6 and 3.7 support. Add 3.8 and 3.10 as build / release targets.
- Move docker image to using python:3.10.

## 7.2 0.5.3 (2020-08-19)

- Another CI tweak to validate pypi packages.

## 7.3 0.5.2 (2020-08-19)

- Another CI tweak to validate pypi packages.

## 7.4 0.5.1 (2020-08-19)

- CI tweak to validate pypi packages.

## 7.5 0.5.0 (2020-03-13)

- Bump lark
- CI tweaks.

## 7.6 0.4.1 (2018-04-28)

- Use msgpack to serialize the memory blob for use in loop detection. Makes fuckery vroom vroom fast.

## 7.7 0.4.0 (2018-04-28)

- Update lark to a modern version
- Rearrange test code layout a bit

## 7.8 0.3.9 (2018-04-28)

- Use fstrings everywhere!
- Change circleci config from 2.0 to 2.1.
- Fix DeprecationWarning

## 7.9 0.3.8 (2018-04-03)

- Add daily CI builds.

## 7.10 0.3.7 (2018-03-26)

- Fix comment line.

## 7.11 0.3.6 (2018-03-26)

- Tweak tag build rules

## 7.12 0.3.5 (2018-03-26)

- Tweak tag build rules

## 7.13 0.3.4 (2018-03-26)

- Tweak tag build rules

## 7.14 0.3.3 (2018-03-26)

- Tweak tag build rules

## 7.15 0.3.2 (2018-03-26)

- Tweak tag build rules

## 7.16 0.3.1 (2018-03-26)

- Tweak tag build rules

## 7.17 0.3.0 (2018-03-26)

- Remove TravisCI build support.
- Add CircleCI support for CI testing.
- Add CircleCI support for PyPi publishing and Docker container building.

## 7.18 0.2.3 (2017-03-27)

- Add a parse_and_run() function to the VirtualMachine class, to allow it to execute arbitrary brainfuck programs.
- Update docstrings considerably, and improve sphinx based autodoc usage.
- Add CircleCI testing

## 7.19 0.2.2 (2017-03-01)

- Fix issue with doc generation.

## 7.20 0.2.1 (2017-03-01)

- Fix issue with wheel's and trove classifiers on pypi.

## 7.21 0.2.0 (2017-03-01)

- Working brainfuck interpreter available.
- Renamed package from pyfuckery to fuckery.

## 7.22 0.1.0 (2017-02-12)

- First release on PyPI.

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## f

fuckery, 7
fuckery.cli, 7
fuckery.constants, 7
fuckery.exc, 8
fuckery.memory, 8
fuckery.parser, 9
fuckery.vm, 9

# Index


## A

AddressError, 8

## C

current_value (*fuckery.vm.VirtualMachine attribute*), 10

## D

dec_data_ptr() (*fuckery.vm.VirtualMachine method*), 10
dec_data_value() (*fuckery.vm.VirtualMachine method*), 10
DEFAULT_MEMORY_SIZE (*in module fuckery.constants*), 7

## E

ExitCondition, 8

## F

fuckery (*module*), 7
fuckery.cli (*module*), 7
fuckery.constants (*module*), 7
fuckery.exc (*module*), 8
fuckery.memory (*module*), 8
fuckery.parser (*module*), 9
fuckery.vm (*module*), 9
FuckeryError, 8

## G

get() (*fuckery.memory.Storage method*), 9

## I

inc_data_ptr() (*fuckery.vm.VirtualMachine method*), 10
inc_data_value() (*fuckery.vm.VirtualMachine method*), 10
io_input() (*fuckery.vm.VirtualMachine method*), 10
io_output() (*fuckery.vm.VirtualMachine method*), 10

## M

main() (*in module fuckery.cli*), 7
main() (*in module fuckery.memory*), 9
main() (*in module fuckery.parser*), 9
main() (*in module fuckery.vm*), 11
makeargpaser() (*in module fuckery.cli*), 7
makeargpaser() (*in module fuckery.memory*), 9
makeargpaser() (*in module fuckery.parser*), 9
makeargpaser() (*in module fuckery.vm*), 11
mem_hash (*fuckery.memory.Storage attribute*), 9

## P

parse_and_run() (*fuckery.vm.VirtualMachine method*), 10
parse_program() (*in module fuckery.parser*), 9

## R

run() (*fuckery.vm.VirtualMachine method*), 10

## S

set() (*fuckery.memory.Storage method*), 9
state_hash (*fuckery.vm.VirtualMachine attribute*), 11
Storage (*class in fuckery.memory*), 8
StorageError, 8
SYM_DATA_DEC (*in module fuckery.constants*), 7
SYM_DATA_INC (*in module fuckery.constants*), 8
SYM_IO_INPUT (*in module fuckery.constants*), 8
SYM_IO_OUTPUT (*in module fuckery.constants*), 8
SYM_JMP_BACKWARD (*in module fuckery.constants*), 8
SYM_JMP_FWD (*in module fuckery.constants*), 8
SYM_PTR_DEC (*in module fuckery.constants*), 8
SYM_PTR_INC (*in module fuckery.constants*), 8

## V

VirtualMachine (*class in fuckery.vm*), 9
VMError, 8