bftools

Release 0.4.0

BobDotCom

Sep 21, 2022

CONTENTS

A brainfuck toolbox for python.

PyPI: https://pypi.org/project/bftools/

Docs: https://bftools.readthedocs.io/en/latest/

CHAPTER

ONE

INSTALLATION

You can install released versions of bftools from the Python Package Index via pip or a similar tool:

Stable Release: `pip install bftools`

Working Version: `pip install git+https://github.com/BobDotCom/bftools.git`

CHAPTER

TWO

USAGE

```
import bftools
comp = bftools.BrainfuckTools()
py = comp.compile("++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>>++++++++++++++++.-------
↪--------.+++++++++++++.+.")
print(py.code)
```

2.1 Manuals

2.1.1 API Reference

The following section outlines the API of bftools.

Core Utilities

These provide the main functionality of bftools.

BrainfuckTools Class

class bftools.**BrainfuckTools**

The BrainfuckTools class is a wrapper for the compiler, decoder and encoder methods.

It comes with some tools to make it easier to use, such as caching the last compiled code, the last decoded code and the last encoded text.

last_compiled

The last compiled code.

Type
Optional[*CompiledBrainfuck*]

last_decoded

The last decoded code.

Type
Optional[*DecodedBrainfuck*]

`last_encoded`

The last encoded text.

Type
Optional[*EncodedBrainfuck*]

`compile`(*code: str*) → *CompiledBrainfuck*

Compiles a brainfuck code into python code.

Parameters
code (*str*) – The brainfuck code to compile.

Returns
The compiled code.

Return type
CompiledBrainfuck

`decode`(*code: str*) → *DecodedBrainfuck*

Decodes brainfuck code into text.

Parameters
code (*str*) – The brainfuck code to decode.

Returns
The decoded code.

Return type
DecodedBrainfuck

`encode`(*text: str*) → *EncodedBrainfuck*

Encodes text into brainfuck code.

Parameters
text (*str*) – The text to encode.

Returns
The encoded text.

Return type
EncodedBrainfuck

Shortcut Functions

bftools.`compile`(*code: str*) → *CompiledBrainfuck*

Shortcut for *BrainfuckTools.compile()*.

This is equivalent to `BrainfuckTools().compile(code)`.

Parameters
code (*str*) – The brainfuck code to compile.

Returns
The compiled code.

Return type
CompiledBrainfuck

bftools.**decode**(*code: str*) → *DecodedBrainfuck*

Shortcut for *BrainfuckTools.decode()*.

This is equivalent to `BrainfuckTools().decode(code)`.

Parameters
code (*str*) – The brainfuck code to decode.

Returns
The decoded text.

Return type
DecodedBrainfuck

bftools.**encode**(*text: str*) → *EncodedBrainfuck*

Shortcut for *BrainfuckTools.encode()*.

This is equivalent to `BrainfuckTools().encode(text)`.

Parameters
text (*str*) – The text to encode.

Returns
The encoded text.

Return type
EncodedBrainfuck

Converted Classes

These classes are returned by various methods from the *Core Utilities*.

class bftools.**CompiledBrainfuck**

An object to represent python compiled from Brainfuck.

To receive the decoded text, use *result* or str(*DecodedBrainfuck*).

> **Warning:** This class is not intended to be instantiated directly. Use *decode()* or *BrainfuckTools.decode()* instead.

result

The result code. This will never be `None` unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Type
Optional[str]

property code: Optional[str]

The compiled code.

Deprecated since version 0.3.0: The code property is deprecated and will be removed in 0.5.0. Use *result* or str(*CompiledBrainfuck*) instead.

Returns
The compiled code. This will never be `None` unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Return type
Optional[str]

property raw_parsed: Optional[Tuple[Symbol, ...]]

Raw parsed code

This will never be None unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Note: This is meant to be used internally and you should not need to use it.

Changed in version 0.3.0: Now returns None instead of raising a ValueError.

Returns
The raw parsed code.

Return type
Optional[Tuple[Symbol]]

parse(*code: str*) → None

Parse the given code.

Note: You should not need to use this method. It is intended for internal use only, so you should only need to use it if you override the functionality of the library. This method is not dangerous like *DecodedBrainfuck.parse()* is.

Parameters
code (*str*) – The code to parse.

class bftools.DecodedBrainfuck

An object to represent text decoded from Brainfuck.

To receive the decoded text, use *result* or str(*DecodedBrainfuck*).

Warning: This class is not intended to be instantiated directly. Use *decode()* or *BrainfuckTools.decode()* instead.

result

The result text. This will never be None unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Type
Optional[str]

property text: Optional[str]

The decoded text.

Deprecated since version 0.3.0: The text property is deprecated and will be removed in 0.5.0. Use *result* or str(*DecodedBrainfuck*) instead.

Returns
The result text. This will never be None unless *parse()* has not been called. Since the

library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Return type
Optional[str]

parse(*code: str*) → None

Parse the given code.

Note: You should not need to use this method. It is intended for internal use only, so you should only need to use it if you override the functionality of the library. See the warning below for more information.

Warning: This method uses the `exec()` function. It is therefore not safe to use this method with untrusted code. The library uses this internally and will ensure that user input is not directly passed to this method, unless you override the functionality of the library. If you invoke this method directly, you need to ensure that the code you pass is safe.

Parameters
code (*str*) – The code to parse.

Raises
SyntaxError – If the code is not syntactically correct.

class bftools.**EncodedBrainfuck**

An object to represent text encoded into Brainfuck.

To receive the encoded Brainfuck, use *result* or str(*EncodedBrainfuck*).

Warning: This class is not intended to be instantiated directly. Use *encode()* or *BrainfuckTools.encode()* instead.

result

The result code. This will never be `None` unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Type
Optional[str]

property code: Optional[str]

The encoded code.

Deprecated since version 0.3.0: The code property is deprecated and will be removed in 0.4.0. Use *result* or str(*EncodedBrainfuck*) instead.

Returns
The encoded text. This will never be `None` unless *parse()* has not been called. Since the library always calls *parse()* before returning the object, this should never happen unless you override the functionality of the library.

Return type
Optional[str]

parse(*text: str*) → None

Parse the given text.

Note: You should not need to use this method. It is intended for internal use only, so you should only need to use it if you override the functionality of the library. This method is not dangerous like *DecodedBrainfuck.parse()* is.

Parameters
text (*str*) – The text to parse.

Tools

These are tools that are used internally by the *Core Utilities*.

bftools.**factor**(*x: int*) → Tuple[int, int]

Factors **x** into 2 numbers, a and b, such that a + b is as small as possible.

Parameters
x (int) – The number to factor

Returns
A Tuple of 2 integers that factor into **x**.

Return type
Tuple[int, int]

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX