



HUMBOLDT UNIVERSITY OF BERLIN

EINFÜHRUNG IN DAS WISSENSCHAFTLICHE RECHNEN

# Documentation of CLI Fraction Calculator

*Christian Parpart & Kei Thoma*

May 17, 2019

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Euclidean Algorithm Library</b>	<b>3</b>
2.1	euclidean_algorithm(a, b) . . . . .	3
2.2	least_common_multiple(a, b) . . . . .	3
2.3	main() . . . . .	4
<b>3</b>	<b>Fraction API</b>	<b>4</b>
3.1	__init__(numerator, denominator) . . . . .	4
3.2	Get Attribute Functions . . . . .	4
3.3	__pos__() . . . . .	5
3.4	__neg__() . . . . .	5
3.5	__abs__() . . . . .	5
3.6	__add__() . . . . .	5
3.7	__sub__() . . . . .	5
3.8	__mul__() . . . . .	5
3.9	__truediv__() . . . . .	5
3.10	__str__() . . . . .	5
3.11	__main__() . . . . .	5
<b>4</b>	<b>Fraction Calculator CLI</b>	<b>5</b>
<b>5</b>	<b>Improvement Horizon</b>	<b>5</b>
5.1	Project Complexity One . . . . .	5

## 1 Introduction

## 2 Euclidean Algorithm Library

In this module, we first implemented the well known euclidean algorithm which finds the greatest common divisor given two integers. From there, we use the result of the above mentioned algorithm to calculate the least common multiple.

### 2.1 `euclidean_algorithm(a, b)`

#### Arguments

1. `first_number` (int): the first integer,  $a$ ; negative values are accepted
2. `second_number` (int): the second integer,  $b$ ; negative values are accepted

#### Returns

- (int): the greatest common divisor found via the recursive euclidean algorithm

#### Description

This function finds the greatest common divisor with the help of recursive euclidean algorithm.

#### Example

### 2.2 `least_common_multiple(a, b)`

#### Arguments

1. `first_number` (int): the first integer,  $a$ ; negative values are accepted
2. `second_number` (int): the second integer,  $b$ ; negative values are accepted

#### Returns

- (int): the least common multiple calculated with the help of the euclidean algorithm and the formula

#### Description

This finds the least common multiple.

**Example****2.3 `main()`****3 Fraction API**

The `Fraction` class in `fraction.py` implements a fraction, i.e. concepts such as  $\frac{1}{2}$  or  $-\frac{2}{3}$ , mathematically correctly. For this endeavor, `Fraction` saves three pseudo-private attributes representing the unsigned numerator, the unsigned denominator and finally the sign of the fraction.

After an instance of `Fraction` is initialized, it is automatically reduced properly to the most minimal form, e.g.  $\frac{8}{12}$  naturally becomes  $\frac{2}{3}$ , with the help of the euclidean algorithm.

Finally, to allow some easy way to handle this class, few build-in operators such as the absolute function and binary addition were overloaded.

**3.1 `__init__(numerator, denominator)`****Arguments**

1. **numerator** (int): the numerator; negative values are allowed, but is then saved as a positive integer at `numerator_`
2. **denominator** (int): the denominator; negative values are allowed, but is then saved as a positive integer at `denominator_`; if no argument is passed, it defaults to 1

Note that even though `numerator_` and `denominator_` are always positive, the sign of the `Fraction` is determined at the point of initialization and is saved under the boolean attribute `sign_`.

**Raises**

- `ZeroDivisionError`: if 0 is passed as the parameter for the denominator

**Description****3.2 Get Attribute Functions**

Fortunately or unfortunately depending on one's perspective about dynamic languages, Python does not allow private attributes or methods. However, we don't want that the three attributes, `numerator_`, `denominator_`, and `sign_`, are modifiable from the outside of the `Fraction` class. Therefore, this class provides three methods, `get_numerator()`, `get_denominator()`, and `get_sign()`, which simply returns the respective attribute.

### 3.3 `--pos--()`

#### Return

- `(self)`: returns the unchanged self

#### Description

Overloading the unitary plus operator is not very exciting. The fraction object is unchanged and returned immediately.

### 3.4 `--neg--()`

### 3.5 `--abs--()`

### 3.6 `--add--()`

### 3.7 `--sub--()`

### 3.8 `--mul--()`

### 3.9 `--truediv--()`

### 3.10 `--str--()`

### 3.11 `--main--()`

## 4 Fraction Calculator CLI

## 5 Improvement Horizon

### 5.1 Project Complexity One

### 5.2