# Protokoll
# Test Driven Development

Softwareentwicklung
4CHIT 2016/17

Martin Wölfer

**Version 0.2**
**Begonnen am 12. Oktober**
**Beendet am 23. Oktober**

# Inhaltsverzeichnis

# 1 Einführung

## 1.1 Ziele

- Eine Klasse Bruch erstellen
- Die Klasse anhand den Tests entwickeln
- Eine Coverage $> 95$
- Eine Sphinx Dokumentation erstellen

## 1.2 Voraussetzungen

- Python Grundwissen
- Sphinx installiert
- Test cases

## 1.3 Aufgabenstellung

Schreiben Sie zu die Klasse Bruch in einem Modul bruch

Nutzen Sie die Testklassen in PyCharm.

Ziel: Coverage $> 95$

Protokoll mit Testreports (inkl. Coverage) und Dokumentation (html)

Abgabe des Python-files

Achtung: Vergessen Sie nicht auf eine ausführliche Dokumentation mittels sphin

# 2 Ergebnisse

## 2.1 Klasse Bruch

Die Klasse Bruch soll Bruchteile darstellen können. Sie implementiert fast alle von Python bereitgestellten magischen Methoden, dadurch kann man sehr angenehm mit dieser Klasse arbeiten.

## 2.2 Beispiele

```
print(Bruch(5,4))
```

Output: `(5/4)`

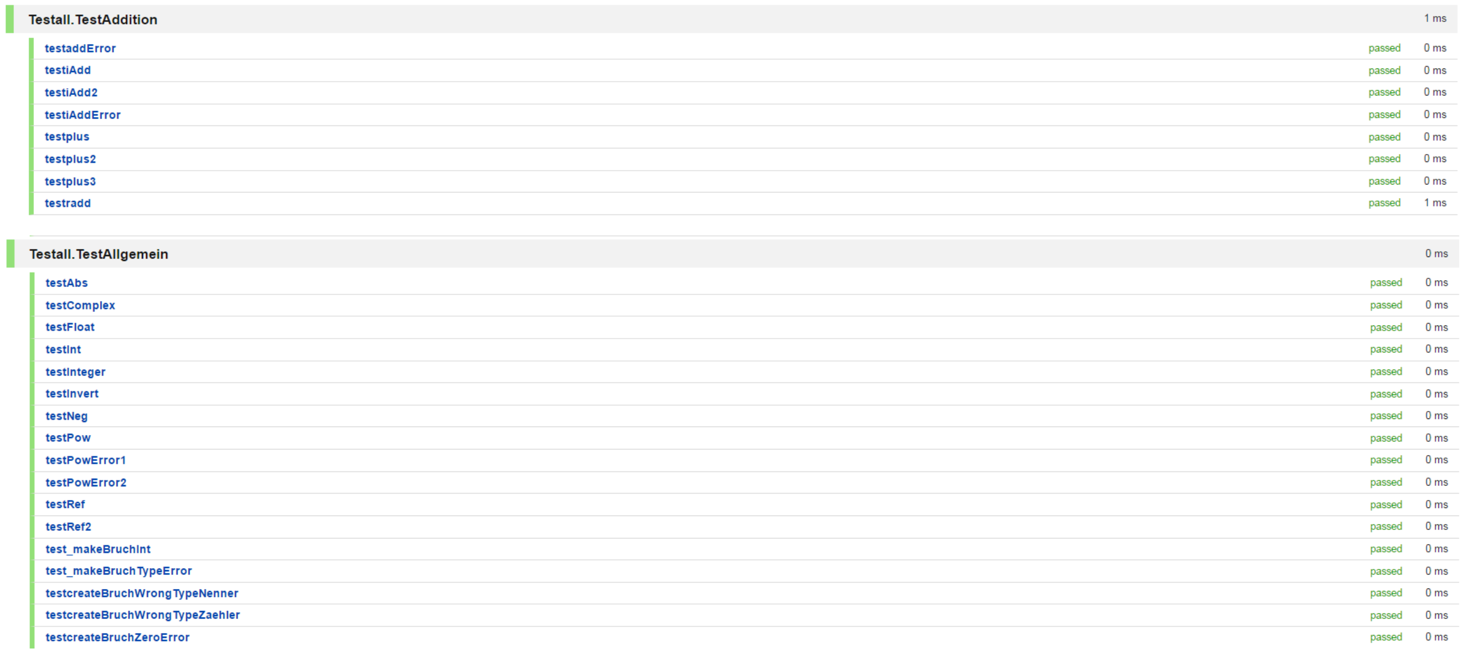
```
print(float(Bruch(5,4)))
```

Output: `1,25`

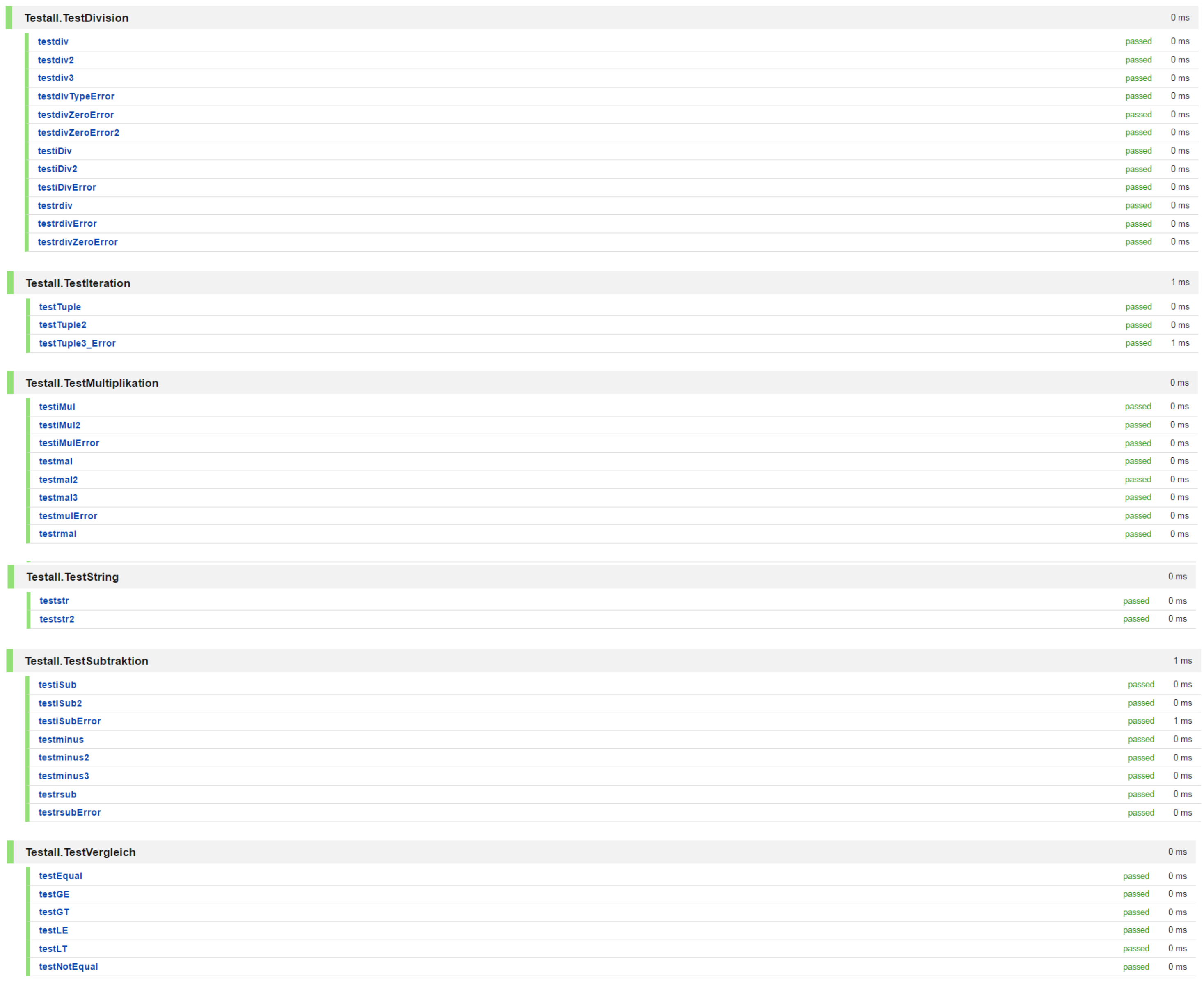
```
print(Bruch(3,2) + 1)
```

Output: `(5/4)`

## 2.3 Test reports

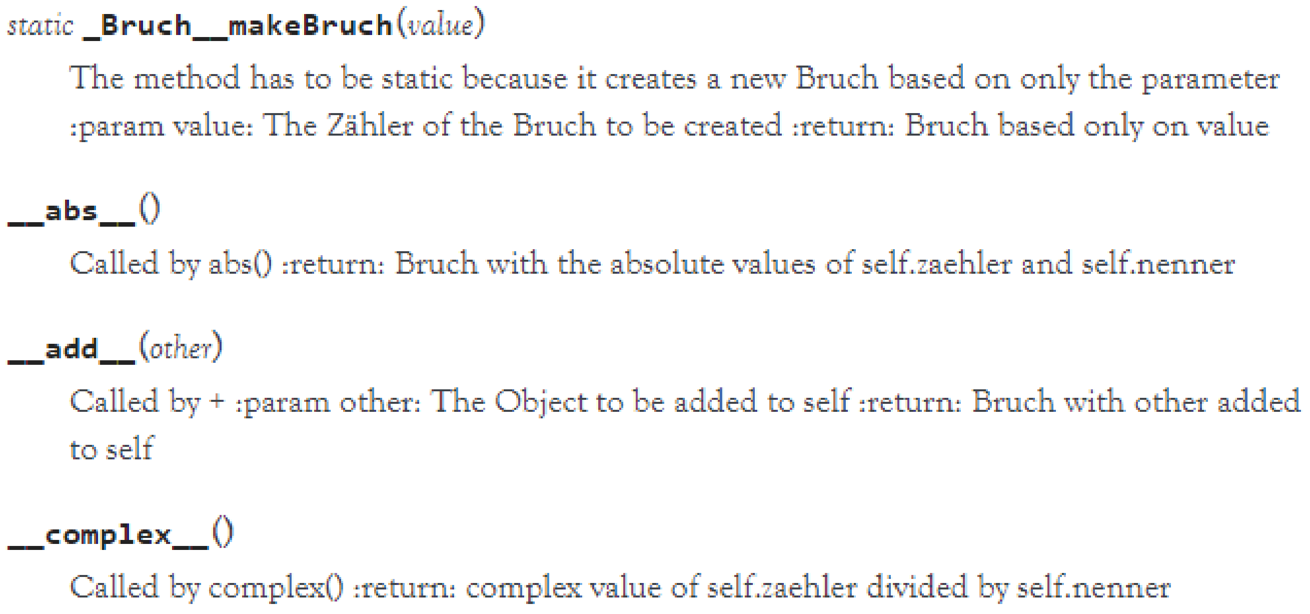


| | | |
|---|---|---|
| Testall.TestAddition | | 1 ms |
| testaddError | passed | 0 ms |
| testiAdd | passed | 0 ms |
| testiAdd2 | passed | 0 ms |
| testiAddError | passed | 0 ms |
| testplus | passed | 0 ms |
| testplus2 | passed | 0 ms |
| testplus3 | passed | 0 ms |
| testradd | passed | 1 ms |
| Testall.TestAllgemein | | 0 ms |
| testAbs | passed | 0 ms |
| testComplex | passed | 0 ms |
| testFloat | passed | 0 ms |
| testInt | passed | 0 ms |
| testInteger | passed | 0 ms |
| testInvert | passed | 0 ms |
| testNeg | passed | 0 ms |
| testPow | passed | 0 ms |
| testPowError1 | passed | 0 ms |
| testPowError2 | passed | 0 ms |
| testRef | passed | 0 ms |
| testRef2 | passed | 0 ms |
| test_makeBruchInt | passed | 0 ms |
| test_makeBruchTypeError | passed | 0 ms |
| testcreateBruchWrongTypeNenner | passed | 0 ms |
| testcreateBruchWrongTypeZaehler | passed | 0 ms |
| testcreateBruchZeroError | passed | 0 ms |



```
Testall.TestDivision                          0 ms
    testdiv                          passed   0 ms
    testdiv2                         passed   0 ms
    testdiv3                         passed   0 ms
    testdivTypeError                 passed   0 ms
    testdivZeroError                 passed   0 ms
    testdivZeroError2                passed   0 ms
    testiDiv                         passed   0 ms
    testiDiv2                        passed   0 ms
    testiDivError                    passed   0 ms
    testrdiv                         passed   0 ms
    testrdivError                    passed   0 ms
    testrdivZeroError                passed   0 ms

Testall.TestIteration                         1 ms
    testTuple                        passed   0 ms
    testTuple2                       passed   0 ms
    testTuple3_Error                 passed   1 ms

Testall.TestMultiplikation                    0 ms
    testiMul                         passed   0 ms
    testiMul2                        passed   0 ms
    testiMulError                    passed   0 ms
    testmal                          passed   0 ms
    testmal2                         passed   0 ms
    testmal3                         passed   0 ms
    testmulError                     passed   0 ms
    testrmal                         passed   0 ms

Testall.TestString                            0 ms
    teststr                          passed   0 ms
    teststr2                         passed   0 ms

Testall.TestSubtraktion                       1 ms
    testiSub                         passed   0 ms
    testiSub2                        passed   0 ms
    testiSubError                    passed   1 ms
    testminus                        passed   0 ms
    testminus2                       passed   0 ms
    testminus3                       passed   0 ms
    testrsub                         passed   0 ms
    testrsubError                    passed   0 ms

Testall.TestVergleich                         0 ms
    testEqual                        passed   0 ms
    testGE                           passed   0 ms
    testGT                           passed   0 ms
    testLE                           passed   0 ms
    testLT                           passed   0 ms
    testNotEqual                     passed   0 ms
```

## 2.4 Dokumentation



*static* **`_Bruch__makeBruch`**(*value*)

The method has to be static because it creates a new Bruch based on only the parameter :param value: The Zähler of the Bruch to be created :return: Bruch based only on value

**`__abs__`**()

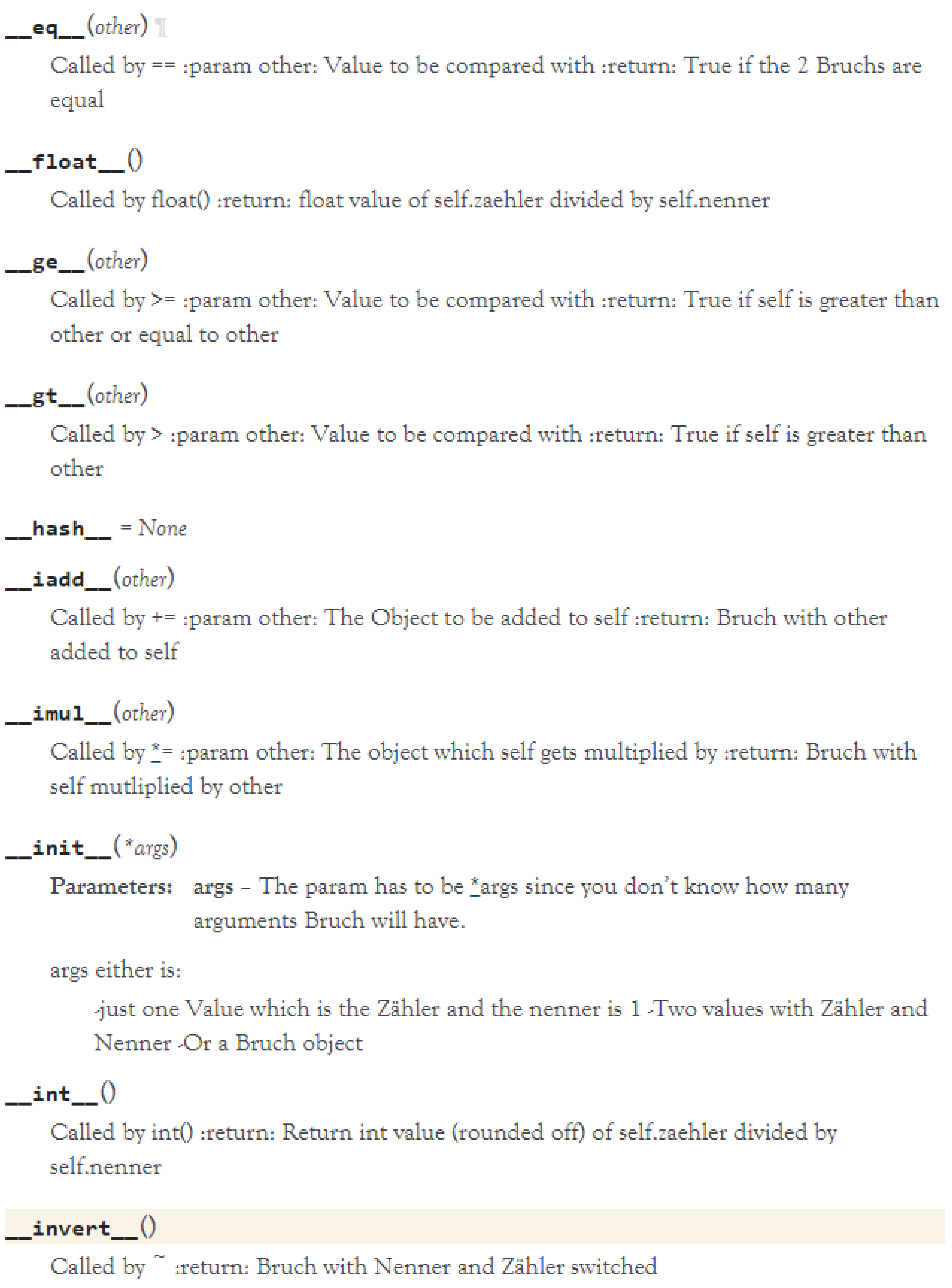
Called by abs() :return: Bruch with the absolute values of self.zaehler and self.nenner

**`__add__`**(*other*)

Called by + :param other: The Object to be added to self :return: Bruch with other added to self

**`__complex__`**()

Called by complex() :return: complex value of self.zaehler divided by self.nenner



**`__eq__`**(*other*) ¶

Called by == :param other: Value to be compared with :return: True if the 2 Bruchs are equal

**`__float__`**()

Called by float() :return: float value of self.zaehler divided by self.nenner

**`__ge__`**(*other*)

Called by >= :param other: Value to be compared with :return: True if self is greater than other or equal to other

**`__gt__`**(*other*)

Called by > :param other: Value to be compared with :return: True if self is greater than other

**`__hash__`** = *None*

**`__iadd__`**(*other*)

Called by += :param other: The Object to be added to self :return: Bruch with other added to self

**`__imul__`**(*other*)

Called by *= :param other: The object which self gets multiplied by :return: Bruch with self mutliplied by other

**`__init__`**(**args*)

**Parameters:** **args** – The param has to be *args since you don't know how many arguments Bruch will have.

args either is:

-just one Value which is the Zähler and the nenner is 1 -Two values with Zähler and Nenner -Or a Bruch object

**`__int__`**()

Called by int() :return: Return int value (rounded off) of self.zaehler divided by self.nenner

**`__invert__`**()

Called by ~ :return: Bruch with Nenner and Zähler switched

**`__isub__`**(*other*)

Called by -= :param other: The Object to be subtracted from self :return: Bruch with self subtracted from other

**`__iter__`**()

Called by z,n = Bruch(z,n) :return: Kind of a list which is iterable

**`__itruediv__`**(*other*)

Called by /= :param other: The object which self gets divided by :return: Bruch with self divided by other

**`__le__`**(*other*)

Called by <= :param other: Value to be compared with :return: True if self is less than other or equal to other

**`__lt__`**(*other*)

Called by < :param other: :return: True if self is less than other

**`__module__`** = *'bruch.Bruch'*

**`__mul__`**(*other*)

Called by * :param other: The object which self gets multiplied by :return: Bruch with self mulitplied by other

**`__neg__`**()

Called by - :return: Either Bruch with negative self.zaehler or Bruch with negative self.nenner => Double negative => Positive

**`__pow__`**(*power*)

Called by ** :param power: The exponent of the Bruch :return: Bruch with self.zaehler and self.nenner to the power of the exponent

**`__radd__`**(*other*)

Called by + :param other: The Object which gets self added to it :return: Bruch with self added to other

`__rmul__`(*other*)
: Called by * :param other: The object which gets multiplied by self :return: Bruch with other multiplied by self

`__rsub__`(*other*)
: Called by - :param other: The object which gets self subtracted from it :return: Bruch with other subtracted from self

`__rtruediv__`(*other*)
: Called by / :param other: The object which gets divided by self :return: Bruch with other divided by self

`__str__`()
: Called by str(), also called when printing :return: The Bruch in Parenthesis and with a Slash inbetween self.nenner and self.nenner If self.nenner is 1 then it'll only put out self.zaehler

`__sub__`(*other*)
: Called by - :param other: The Object to be subtracted from self :return: Bruch with subtracted from other

`__truediv__`(*other*)
: Called by / :param other: The object which self gets divided by :return: Bruch with self divided by other

`__weakref__`
: list of weak references to the object (if defined)

## 2.5 Github

Die Implementation und Genaue Dokumentation ist auf meinem Github-Repository online gestellt: https://github.com/mwoelfer-tgm/Bruch