12.5. Überladen von Operatoren und magische Methoden

- die arithmetischen Operatoren sind für die elementaren Datentypen definiert, bewirken jedoch verschiedene Aktionen:

```
zahl1 = 5
zahl2 = 10

text1 = "Hallo"
text2 = "Welt"

print(zahl1+zahl2)
print(text1+text2)
```

```
15
HalloWelt
```

- falls der Operator **+** für die eigene Klasse benötigt wird, ist eine Methode zu erstellen, die den Additionsoperator für die Klasse definiert
- für den Additionsoperator ist in der Klasse die magische Methode **__add__()** zu definieren

- Erläuterung am Beispiel der Klasse Bruch
  - hat die Attribute Zähler und Nenner
  - hat die Konstruktor-Methode **__init__()**
  - hat eine Methode zum Ausgeben
  - enthält die die magische Methode **__add__()**, um die Addition abzubilden (es gelten die Regeln der Bruchrechnung: auf gleichem Nenner bringen und addieren)

- ```
  # Defintion der Klasse Bruch
  class Bruch:
      # Definition der Konstruktor-Methode __init__()
      def __init__(self, zaehler, nenner):
          self.zaehler = zaehler
          self.nenner = nenner


      # Definition der Methode zum Aufbereiten
      def __str__(self):
          print(f"Bruch: {self.zaehler} / {self.nenner}")


      # Definition der Methode zum Addieren für die Klasse Bruch
      # wird aufgerufen bei <obj> + <obj>
      # TypeError: unsupported operand type(s) for +: 'Bruch' and
  'Bruch'
      def __add__(self, other):
          z = self.zaehler*other.nenner+self.nenner*other.zaehler
          n = self.nenner*other.nenner
          return Bruch(z, n)
  ```
  ```

```
# Test der Klasse Bruch
# zaehler = int(input("Zähler: "))
# nenner = int(input("Nenner: "))
# oder
zaehler = 2
nenner = 3

# Erzeugen eines Objektes der Klasse Bruch mit Anfangswerten
# ein Bruch wird instanziiert und verkörpert hier den Bruch 2/3
b = Bruch(zaehler, nenner)
# Ausgeben des Objektes (der Klasse Bruch) b
b.ausgeben()

# Addition von zwei Objekten (der Klasse Bruch)
# Variable c verweist auf die Summe von b + b
c = b + b
# Ausgeben des Objektes (der Klasse Bruch) c
c.ausgeben()
```

```
Bruch: 2 / 3
Bruch: 12 / 9
```

12.5.1. Magische Methoden und ihre Operatoren (Auszug)

__add__()	+
__sub__()	-
__mul__()	*
__truediv__()	/
__floordiv__()	//
__pow__()	**

__lt__()	<
__le__()	<=
__gt__()	>
__ge__()	>=
__eq__()	==
__ne__()	!=

Aufgabe

1. Klasse Bruch
   die magischen Methoden für die Subtraktion (__sub__), Multiplikation (__mul__) und die Division (__truediv__), sowie die Operationen zur Prüfung auf Gleichheit (__eq__) und Ungleichheit (__ne__) sind zu erstellen und über entsprechende Tests zu überprüfen.

2. Es ist eine Benutzeroberfläche mit tkinter für die Klasse Bruch zu erstellen, um zwei Brüche addieren, subtrahieren, multiplizieren und dividieren zu können, sowie um auf Gleichheit und Ungleichheit prüfen zu können.