Hands On

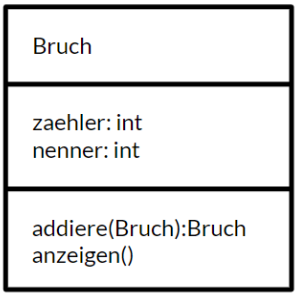
Rechnen mit Brüchen - Bruchklasse

In dieser Aufgabe soll aus dem UML-Klassendiagramm zur Klasse „Bruch“ ein Python-Skript entstehen, welches diese Klasse mit den zwei Methoden `anzeigen` und `addiere` implementiert. Zähler und Nenner sind dabei nicht-negative natürliche Zahlen.

Erstelle in Thonny zuerst eine neue Datei `bruch.py` und übernimm das vorgegebene Grundgerüst:

```
class Bruch:
      def __init__(self, …):
            …
      def anzeigen(self):
            …

…
```



1.

Ergänze den Kopf der Konstruktor-Methode `__init__()` um zwei Parameter `zaehler` und `nenner`.

Diese sollen in der Konstruktor-Methode genutzt werden, um die zwei Attribute `zaehler` und `nenner` zu initialisieren. Ergänze die Methode dementsprechend um diese zwei Attribute.

Vergiss nicht, dass Attribute über das Schlüsselwort `self` referenziert werden, also bspw. `self.attribut`.

2.

Ergänze die Methode `anzeigen()` so, dass diese den Bruch im Format `1/2` mithilfe von `print()` ausgibt.

Du kannst einen Integer (int) mithilfe von `str(zahl)` in einen String konvertieren.

Teste deine Methode, indem du einen Bruch a mit Beispielwerten initialisierst und dann `a.anzeigen()` aufrufst.

3.

Ergänze die Klasse um die Methode `addiere()` so, dass der Bruch mit einem zweiten addiert wird. Dieser zweite Bruch erhält die Methode über einen Parameter.

Das Ergebnis der Addition der beiden Brüche soll über ein `return`-Statement zurückgegeben werden.

🐞 Vergiss nicht, dass zum Addieren von Brüchen der Nenner zuerst gleichgesetzt werden muss.

🧪 Teste deine Methode, indem du drei Bruch-Objekte `a,b,c` erstellst. Dabei soll c die Summe von a und b sein, also: `c = a.addiere(b)`. Lass dir alle drei Brüche anzeigen, a und b sollten sich nicht verändert haben.

Wenn du es bis hierhin geschafft hast, well done! 🎉 Schau dir gerne noch die Bonusaufgabe an und löse sie, wenn du schon so weit gekommen bist.

Bonus

Erstelle für die Klasse `Bruch` eine neue Methode `kuerzen()`, welche sicherstellen soll, dass ein Bruch immer in gekürzter Form ausgegeben wird.

- Das Kürzen eines Bruches kann man leicht durchführen, indem der größte gemeinsame Teiler (ggT) von Zähler und Nenner berechnet wird. Eine mögliche Berechnung des ggT ist anhand zweier Beispiele im Folgenden veranschaulicht:

  Beispiel 1: ggT von 544 und 391
  544 / 391 = 1 mit Rest 153
  391 / 153 = 2 mit Rest 85
  153 / 85 = 1 mit Rest 68
  85 / 68 = 1 mit Rest 17
  68 / 17 = 4 mit Rest 0

  Die Division geht auf (Rest 0), der ggT von 544 und 391 ist somit 17.

  Beispiel 2: ggT von 13 und 7
  Es ist nun der ggT von 13 und 7 gesucht.
  13 / 7 = 1 mit Rest 6
  7 / 6 = 1 mit Rest 1
  6 / 1 = 6 mit Rest 0

  Die Division geht auf (Rest 0), der ggT von 13 und 7 ist 1 (da 13 und 7 teilerfremd sind).

- Um den Rest einer ganzzahligen Division zu erhalten kannst du den Modulo-Operator (%) verwenden.

  Beispiel in Python: `13%5 == 3`, da 13 / 5 = 2 mit Rest 3 ist.