Mathehelfer Bruchrechnen

Ein Hackathon für Kinder und Jugendliche

Norbert Seulberger

September 2022

Inhaltsverzeichnis

1 Aufgabenstellung und Planung

1.1 Die Aufgabenstellung

Der Mathehelfer Bruchrechnen kann in der Grundversion Brüche und Kommazahlen ineinander umrechnen. Außerdem können Brüche gekürzt werden.

Die Darstellung der Umrechnungen erfolgt in einem Browser. Dabei werden die Brüche grafisch schön mit Zähler, Nenner und Bruchstrich dargestellt wie in einem Mathematikbuch.

Es sind also beispielsweise folgende Operationen möglich:

$$\frac{6}{8} = \frac{3}{4}$$

$$\frac{3}{4} = 0.75$$

$$0.8 = \frac{4}{5}$$

Dabei geben wir die linke Seite fest vor und berechnen die rechte Seite jeweils mit einem Programm. Wir schreiben jeweils eine eigene Funktion, also drei Funktionen, die

- einen Bruch kürzt,
- einen Bruch in eine Kommazahl umrechnet oder
- eine Kommazahl in einen Bruch umwandelt.

Als Programmiersprache verwenden wir Python.

Die Formelzeile tragen wir automatisch in eine kleine Web-Seite ein, die wir uns im Browser ansehen. Dabei beschreiben wir die Formel in einer speziellen HTML-Sprache für mathematische Formeln: MathML.

Die Sprache MathML ist sehr einfach. Allerdings muss man sehr viel Text eingeben, um eine Formel mit MathML zu beschreiben. Daher schreiben wir ein Programm, das die Formel automatisch in MathML ausdrückt.

2 Erklärungen und Aufgaben

2.1 MathML

MathML ist eine einfache Markup-Sprache, die in einem HTML-Dokument dazu verwendet werden kann, um mathematische Formeln darzustellen.

Genauso wie in HTML werden die Daten in sogenannte Tags eingeschlossen, d.h. ein Element beginnt mit einen Tag, dann stehen die Daten und ein schließendes Tag beendet den Ausdruck.

2.1.1 Operanden und Operatoren

Für die verschiedenen Teile einer mathematischen Formel gibt es unterschiedliche Elemente.

- `<mn>42</mn>` stellt die Zahl 42 dar.
- `<mo>=</mo>` beschreibt ein Gleichheitszeichen.
- `<mo>+</mo>` schreibt ein Pluszeichen.

Die Formel $2 + 3 = 5$ lautet also in MathML:

```
<mn>2</mn>
<mo>+</mo>
<mn>3</mn>
<mo>=</mo>
<mn>5</mn>
```

1.1 Tipp
Der Browser ignoriert die Zeilenumbrüche in MathML. Wir können die Formel also auch in eine Zeile schreiben.

```
<mn>2</mn><mo>+</mo><mn>3</mn><mo>=</mo><mn>5</mn>
```

2.1.2 Brüche

2.1.3 Eine Formelzeile

Eine Formel wird wie folgt geschrieben: `<math>` Hier steht die Formel. `</math>`

Damit die Formel in eine eigene Zeile geschrieben wird, verwenden wir das Absatz-Tag von HTML: `<p>` ... `</p>`

Also zusammen:

```
<p>
    <math>
        Hier steht die Formel.
    </math>
</p>
```

2.2 Aufgabenblatt MathML

2.2.1 Aufgabe: Anzeige der HTML-Datei

Schau dir im Firefox-Browser die Datei `manuell.html` an. Die Anzeige sollte etwa so aussehen:

$$\frac{6}{8} = \frac{3}{4}$$

Hier kannst du die Formeln für das Umrechnen eines Bruchs in eine Dezimalzahl und umgekehrt einfügen.

2.2.2 Aufgabe: Ansicht der HTML-Datei im Editor

Schau dir jetzt dieselbe Datei im Editor der Entwicklungsumgebung an. Der Text sieht etwa so aus:

```
<!doctype html>
<html lang="de">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Bruchrechnen</title>
    </head>
    <body>
        <p>
            <math>
                <mfrac>
                    <mi>6</mi>
                    <mi>8</mi>
                </mfrac>
                <mo>=</mo>
                <mfrac>
                    <mi>3</mi>
                    <mi>4</mi>
                </mfrac>
            </math>
        </p>
        <p>
            Hier kannst du die Formeln für das Umrechnen ...
        </p>
    </body>
```

```
</html>
```

Kannst du die Element entdecken, die für die Darstellung des Kürzens der beiden Brüche stehen?

2.2.3 Aufgabe: Die Beschreibung der beiden Umrechungen

Füge in die Datei `manuell.html` den Text in der MathML-Sprache ein, so dass die beiden Umrechungen dargestellt werden (zusätzlich zur Formel für das Kürzen). Der Text „Hier kannst du ...“ darf überschrieben werden.

$$\frac{6}{8} = \frac{3}{4}$$

$$\frac{3}{4} = 0.75$$

$$0.8 = \frac{4}{5}$$

Überprüfe das Ergebnis im Firefox-Browser.

2.3 Algorithmen

Ein Algorithmus ist die eindeutige Beschreibung eines Verfahrens, um eine bestimmte Aufgabe zu lösen.

2.3.1 Beispiel: Das Erweitern eines Bruches

Ein Bruch wird erweitert, indem der Zähler und der Nenner mit demselben Faktor multipliziert werden. Wird beispielsweise der Bruchs $\frac{2}{5}$ mit dem Faktor 3 erweitert, so ergibt sich diese Gleichung:

$$\frac{2}{5} = \frac{2 \cdot 3}{5 \cdot 3} = \frac{6}{15}$$

2.3.2 Der Algorithmus für das Erweitern

- Nimm einen Bruch und den Faktor, mit dem du den Bruch erweitern willst.
- Merke dir den Zähler und den Nenner des Bruchs jeweils in einer Variablen.
- Multipliziere die Variable für den Zähler mit dem Erweiterungsfaktor.
- Verfahre genauso mit dem Nenner.
- Erstelle einen neuen Bruch mit dem neuen Zähler und dem neuen Nenner.

2.4 Funktionen

Eine Funktion ist ein kleines Teilprogramm, das eine spezifische Aufgabe löst. Eine Funktion arbeitet mit folgenden Schritten:

- Die Funktion nimmt einen oder mehrere Eingabewerte entgegen.
- Die Funktion führt eine Berechnung oder andere Aktivität durch.
- Die Funktion gibt das Ergebnis der Berechnung zurück.

Für bestimmte Eingabewerte erhält man immer dasselbe Ergebnis. Die Eingabewerte heißen Parameter. Die Anzahl und der Typ der Parameter ist festgelegt.

2.4.1 Der Aufbau einer Funktion in Python

Der Aufbau einer Funktion orientiert sich an den drei Schritte, die wir eben kennen gelernt haben:

- Die Signaturzeile nennt den Namen und beschreibt die Parameter.
- Der Rumpf der Funktion ist eingerückt. Hier wird die Berechnung durchgeführt.
- Die letzte Zeile im Rumpf gibt das Ergebnis zurück. Sie beginnt mit dem Wort `return`.

2.4.2 Eine Beispielfunktion: Das Erweitern eines Bruchs

In 2.3 haben wir den Algorithmus zum Erweitern eines Bruchs formuliert:

- Nimm einen Bruch und den Faktor, mit dem du den Bruch erweitern willst.
- Merke dir den Zähler und den Nenner des Bruchs jeweils in einer Variablen.
- Multipliziere die Variable für den Zähler mit dem Erweiterungsfaktor.
- Verfahre genauso mit dem Nenner.
- Erstelle einen neuen Bruch mit dem neuen Zähler und dem neuen Nenner.

In Python lautet die Funktion für das Erweitern:

```
def erweitereBruch(bruch, faktor):
    alterZaehler = bruch.zaehler
    alterNenner = bruch.nenner
    neuerZaehler = alterZaehler * faktor
    neuerNenner = alterNenner * faktor
    neuerBruch = Bruch(neuerZaehler, neuerNenner)
    return neuerBruch
```

4.1 Tipp: Datentypen angeben

Wer möchte, kann die Typen der Parameter und des Ergebnisses angeben. Es hilft, Fehler zu vermeiden. Richtige Programmierer tun das.

Die Signaturzeile für die Funktion zum Erweitern lautet dann:

```
def erweitereBruch(bruch: Bruch, faktor: int) -> Bruch:
```

Der Rumpf bleibt unverändert.

2.5 Zeichenketten

2.6 Beispiel: Der Kehrwert eines Bruches

Lies dieses Beispiel sorgfältig durch. Es dient als Vorlage für die Aufgaben, die du danach selbstständig lösen sollst.

2.6.1 Der Kehrwert eines Bruches

Der Kehrwert eines Bruches ist ein anderer Bruch, dessen Zähler der Nenner des ursprünglichen Bruchs und dessen Nenner der ursprüngliche Zähler ist. Beispielsweise lautet der Kehrwert des Bruchs $\frac{2}{3}$ demzufolge $\frac{3}{2}$.

Es gibt eine bekannte Rechenregel zum Dividieren von Brüchen: Ein Bruch wird durch einen anderen Bruch dividiert, indem man den ersten Bruch mit dem Kehrwert des zweiten Bruchs multipliziert:

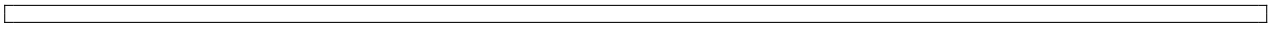
$$\frac{2}{5} : \frac{1}{2} = \frac{2}{5} \cdot \frac{2}{1} = \frac{4}{5}$$

2.6.2 Der Algorithmus

Das Bilden des Kehrwerts funktioniert so:

- Nimm einen Bruch.
- Benenne eine Variable mit dem Namen `neuerZaehler` und weise dieser Variablen den Nenner des übergebenen Bruchs zu.
- Verfahre genauso mit dem neuen Nenner.
- Erstelle einen neuen Bruch aus dem neuen Zähler und dem neuen Nenner.

2.6.3 Die Funktion zur Berechung des Kehrwerts



2.6.4 Die Funktion zur Darstellung der Rechnung zum Kürzen

Der Algorithmus lautet:

- Nimm einen Bruch.
- Erzeuge aus diesem Bruch einen gekürzten Bruch. Verwende dazu die Funktion, die du eben gesehen hast.

- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion, die du im vorherigen Beispiel gesehen hast.
- Erzeuge aus dem gekürzten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion aus dem vorherigen Beispiel.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen.
- Die Zeichenkette beginnt mit <math>.
- Es folgt der MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folge der MathML-Text für den gekürzten Bruch.
- Die Zeichenkette beginnt mit </math>

Die Funktion lautet dann:

```
def schreibeKuerzen(bruch: Bruch) -> str:
    bruchGekuerzt = kuerzeBruch(bruch)
    textUngekuerzt = schreibeBruch(bruch)
    textGekuerzt = schreibeBruch(bruchGekuerzt)
    return "<math>" + textUngekuerzt + "<mo>=</mo>" + textGekuerzt + "
       </math>"
```

2.6.5 Überprüfung des Ergebnisses

Wir überprüfen das Ergebnis analog zum Vorgehen im Beispiel für die Darstellung eines Bruchs. Dazu ergänzen wir in der Funktion `schreibeMathML()` die Zeilen

```
bruchUngekuerzt: Bruch = Bruch(6,8)
inhalt += "\n\t\t<p>"
inhalt += schreibeKuerzen(bruchUngekuerzt)
inhalt += "</p>"
```

2.7 Beispiel: Die Darstellung eines Bruches in MathML

Lies dieses Beispiel sorgfältig durch. Es dient als Vorlage für die Aufgaben, die du danach selbstständig lösen sollst.

2.7.1 Der Algorithmus

Einen Bruch in MathML zu schreiben, ist nicht schwierig, aber mühsam. Daher möchten wir eine Funktion ein Python programmieren, die das automatisch erledigt. Der Algorithmus lautet so:

- Nimm einen Bruch entgegen.
- Wandle den Zähler des Bruchs in eine Zeichenkette.
- Wandle den Nenner des Bruchs in eine Zeichenkette.
- Gib der Zeichenkette, die den Bruch in MathML beschreibt, einen Namen.
- Baue die Zeichenketteaus Einzelteilen zusammen.
- Beginne die Zeichenkette mit `<mfrac>`.
- Rahme die Zeichenkette, die den Zähler beschreibt, mit den Tags `<mi>` und `</mi>` ein.
- Verfahre ebenso mit der Zeichenkette für den Nenner.
- Beende die Zeichenkette mit `</mfrac>`.
- Gib die Zeichenkette zurück.

2.7.2 Die Funktion

Die Funktion sieht dann so aus:

```
def schreibeBruch(bruch: Bruch) -> str:
    zaehlerAlsString = str(bruch.zaehler)
    nennerAlsString = str(bruch.nenner)
    ergebnis = "<mfrac>"
    ergebnis += "<mi>" + zaehlerAlsString + "</mi>"
    ergebnis += "<mi>" + nennerAlsString + "</mi>"
    ergebnis += "</mfrac>"
    return ergebnis
```

2.7.3 Das Überprüfen des Ergebnisses

In der Funktion `schreibeMathML()` ergänzen wir nun die Zeilen für die Darstellung des Bruchs.

```
def schreibeMathML() -> str:
    inhalt = "<p><math><mi>2</mi><mo>+</mo><mi>3</mi>"
                + "<mo>=</mo><mi>5</mi></math></p>"
    bruch: Bruch = Bruch(3,4)
    inhalt += "\n\t\t<p>"
    inhalt += "<math>" + schreibeBruch(bruch) + "</math>"
    inhalt += "</p>"
    return inhalt
```

Danach führen wir die Python-Datei `htmlErzeugung.py` aus und betrachten im Firefox-Browser die Datei `index.html`

2.8 Beispiel: Das Erweitern eines Bruchs

Lies dieses Beispiel sorgfältig durch. Es dient als Vorlage für die Aufgaben, die du danach selbstständig lösen sollst.

2.8.1 Das Erweitern eines Bruches

Ein Bruch wird erweitert, indem der Zähler und der Nenner mit demselben Faktor multipliziert werden. Wird beispielsweise der Bruchs $\frac{2}{5}$ mit dem Faktor 3 erweitert, so ergibt sich diese Gleichung:

$$\frac{2}{5} = \frac{2 \cdot 3}{5 \cdot 3} = \frac{6}{15}$$

Das Erweitern wird beim Addieren von Brüchen benötigt.

2.8.2 Der Algorithmus

- Nimm einen Bruch und den Faktor, mit dem du den Bruch erweitern willst.
- Merke dir den Zähler und den Nenner des Bruchs jeweils in einer Variablen.
- Multipliziere diese Variablen jeweils mit dem Erweiterungsfaktor.
- Erstelle einen neuen Bruch aus dem neuen Zähler und dem neuen Nenner.

2.8.3 Die Funktion zum Erweitern

2.8.4 Die Funktion zur Darstellung des erweiterten Bruchs

Der Algorithmus lautet:

- Nimm einen Bruch.
- Erzeuge aus diesem Bruch einen erweiterten Bruch. Verwende dazu die Funktion, die du eben gesehen hast.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion, die du in 2.7 gesehen hast.
- Erzeuge aus dem erweiterten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion aus 2.7

- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen.
- Die Zeichenkette beginnt mit <math>.
- Es folgt der MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folge der MathML-Text für den gekürzten Bruch.
- Die Zeichenkette beginnt mit </math>

Die Funktion lautet dann:

2.8.5 Überprüfung des Ergebnisses

Wir überprüfen das Ergebnis analog zum Vorgehen im Beispiel für die Darstellung eines Bruchs. Dazu ergänzen wir in der Funktion `schreibeMathML()` die Zeilen

```
bruchUngekuerzt: Bruch = Bruch(6,8)
inhalt += "\n\t\t<p>"
inhalt += schreibeKuerzen(bruchUngekuerzt)
inhalt += "</p>"
```

2.9 Aufgabe: Das Kürzen von Brüchen

Lies dieses Beispiel sorgfältig durch. Es dient als Vorlage für die Aufgaben, die du danach selbstständig lösen sollst.

2.9.1 Der Algorithmus

Das Kürzen eines Bruches funktioniert so:

- Nimm einen Bruch.
- Merke dir den Zähler und den Nenner des Bruchs.
- Berechne den größten gemeinsamen Teiler von Zähler und Bruch. (Es gibt dafür eine fertige Funktion.)
- Berechne den gekürzten Zähler.
- Berechne den gekürzten Nenner.
- Erstelle einen neuen Bruch aus dem gekürzten Zähler und dem gekürzten Nenner.

2.9.2 Die Funktion zum Kürzen von Brüchen

Die Funktion lautet dann:

```
def kuerzeBruch(bruch: Bruch) -> Bruch:
    alterZaehler = bruch.zaehler
    alterNenner = bruch.nenner
    kuerzungsFaktor = ggT(alterZaehler, alterNenner)
    neuerZaehler = alterZaehler // kuerzungsFaktor
    neuerNenner = alterNenner // kuerzungsFaktor
    neuerBruch = Bruch(neuerZaehler, neuerNenner)
    return neuerBruch
```

2.9.3 Die Funktion zur Darstellung der Rechnung zum Kürzen

Der Algorithmus lautet:

- Nimm einen Bruch.
- Erzeuge aus diesem Bruch einen gekürzten Bruch. Verwende dazu die Funktion, die du eben gesehen hast.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion, die du im vorherigen Beispiel gesehen hast.

- Erzeuge aus dem gekürzten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion aus dem vorherigen Beispiel.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen.
- Die Zeichenkette beginnt mit <math>.
- Es folgt der MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folge der MathML-Text für den gekürzten Bruch.
- Die Zeichenkette beginnt mit </math>

Die Funktion lautet dann:

```
def schreibeKuerzen(bruch: Bruch) -> str:
    bruchGekuerzt = kuerzeBruch(bruch)
    textUngekuerzt = schreibeBruch(bruch)
    textGekuerzt = schreibeBruch(bruchGekuerzt)
    return "<math>" + textUngekuerzt + "<mo>=</mo>" + textGekuerzt + "
       </math>"
```

2.9.4 Überprüfung des Ergebnisses

Wir überprüfen das Ergebnis analog zum Vorgehen im Beispiel für die Darstellung eines Bruchs. Dazu ergänzen wir in der Funktion `schreibeMathML()` die Zeilen

```
bruchUngekuerzt: Bruch = Bruch(6,8)
inhalt += "\n\t\t<p>"
inhalt += schreibeKuerzen(bruchUngekuerzt)
inhalt += "</p>"
```

2.10 Aufgabenblatt: Umwandlung eines Bruchs in eine Dezimalzahl

2.11 Aufgabe: Umwandlung einer Dezimalzahl in einen Bruch

2.12 Aufgabe: Programmiere die Grundrechenarten für Brüche

Diese Aufgabe ist etwas schwieriger.

2.13 Aufgabe: Gemischte Zahlen

Diese Aufgabe ist schwierig.

2.14 Aufgabe: Objektorientierte Programmierung

Diese Aufgabe ist sehr schwierig. Es werden gute Kenntnisse der Programmierung in Python vorausgesetzt.

3 Lösungen

3.1 Lösung: Das Kürzen von Brüchen

Lies dieses Beispiel sorgfältig durch. Es dient als Vorlage für die Aufgaben, die du danach selbstständig lösen sollst.

3.1.1 Der Algorithmus

Das Kürzen eines Bruches funktioniert so:

- Nimm einen Bruch.
- Merke dir den Zähler und den Nenner des Bruchs.
- Berechne den größten gemeinsamen Teiler von Zähler und Bruch. (Es gibt dafür eine fertige Funktion.)
- Berechne den gekürzten Zähler.
- Berechne den gekürzten Nenner.
- Erstelle einen neuen Bruch aus dem gekürzten Zähler und dem gekürzten Nenner.

3.1.2 Die Funktion zum Kürzen von Brüchen

Die Funktion lautet dann:

```
def kuerzeBruch(bruch: Bruch) -> Bruch:
    alterZaehler = bruch.zaehler
    alterNenner = bruch.nenner
    kuerzungsFaktor = ggT(alterZaehler, alterNenner)
    neuerZaehler = alterZaehler // kuerzungsFaktor
    neuerNenner = alterNenner // kuerzungsFaktor
    neuerBruch = Bruch(neuerZaehler, neuerNenner)
    return neuerBruch
```

3.1.3 Die Funktion zur Darstellung der Rechnung zum Kürzen

Der Algorithmus lautet:

- Nimm einen Bruch.
- Erzeuge aus diesem Bruch einen gekürzten Bruch. Verwende dazu die Funktion, die du eben gesehen hast.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion, die du im vorherigen Beispiel gesehen hast.

- Erzeuge aus dem gekürzten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion aus dem vorherigen Beispiel.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen.
- Die Zeichenkette beginnt mit <math>.
- Es folgt der MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folge der MathML-Text für den gekürzten Bruch.
- Die Zeichenkette beginnt mit </math>

Die Funktion lautet dann:

```
def schreibeKuerzen(bruch: Bruch) -> str:
    bruchGekuerzt = kuerzeBruch(bruch)
    textUngekuerzt = schreibeBruch(bruch)
    textGekuerzt = schreibeBruch(bruchGekuerzt)
    return "<math>" + textUngekuerzt + "<mo>=</mo>" + textGekuerzt + "
       </math>"
```

3.1.4 Überprüfung des Ergebnisses

Wir überprüfen das Ergebnis analog zum Vorgehen im Beispiel für die Darstellung eines Bruchs. Dazu ergänzen wir in der Funktion `schreibeMathML()` die Zeilen

```
bruchUngekuerzt: Bruch = Bruch(6,8)
inhalt += "\n\t\t<p>"
inhalt += schreibeKuerzen(bruchUngekuerzt)
inhalt += "</p>"
```