Mathehelfer Bruchrechnen

Ein Hackathon für Kinder und Jugendliche

Norbert Seulberger

September 2022

Inhaltsverzeichnis

1 Thema und Planung **3**
- 1.1 Die Aufgabenstellung . . . 4
- 1.2 Technische Rahmenbedingungen . . . 5
- 1.3 Zeitliche Planung . . . 6
- 1.4 Vorkenntnisse . . . 7

2 Zum Warmwerden: MathML **8**
- 2.1 MathML . . . 9
- 2.2 Aufgabenblatt MathML . . . 11

3 Algorithmen und Funktionen **13**
- 3.1 Algorithmen . . . 14
- 3.2 Variablen . . . 15
- 3.3 Funktionen . . . 17
- 3.4 Aufgabe: Das Kürzen eines Bruchs . . . 19
- 3.5 Aufgabe: Das Multiplizieren von Brüchen . . . 20

4 Zeichenketten **21**
- 4.1 Zeichenketten . . . 22
- 4.2 Beispiel: Die Darstellung eines Bruchs in MathML . . . 24

5 Das Erstellen der Formeln **25**
- 5.1 Beispiel: Die Formel zum Erweitern eines Bruchs in MathML . . . 26
- 5.2 Aufgabe: Die Formel zum Kürzen eines Bruchs in MathML . . . 28
- 5.3 Aufgabe: Die Formel zum Multiplizieren von Brüchen in MathML . . . 30

6 Weitere Aufgaben **31**
- 6.1 Aufgabe: Das Addieren von Brüchen . . . 32
- 6.2 Aufgabe: Objektorientierte Programmierung . . . 33

7 Lösungen **34**
- 7.1 Lösung: Das Erweitern eines Bruchs . . . 35
- 7.2 Lösung: Das Kürzen eines Bruchs . . . 37
- 7.3 Lösung: Das Multiplizieren von Brüchen . . . 40

1 Thema 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.

1.2 Technische Rahmenbedingungen

1.3 Zeitliche Planung

1.4 Vorkenntnisse

2 Zum Warmwerden: MathML

2.1 MathML

MathML ist eine einfache 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.

MathML: Anfang und Ende einer Formel

Eine Formeln beginnt mit `<math>` und endet mit `</math>`.

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:

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

Tipp

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

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

Brüche in MathML

Ein Bruch beginnt mit `<mfrac>` und endet mit `</mfrac>`. Den Zähler und den Nenner schreiben wir zwischen `<mn>` und `</mn>`.

Der Bruch $\frac{2}{3}$ sieht also so aus:

```
<math>
    <mfrac>
        <mn>2</mn>
        <mn>3</mn>
    </mfrac>
</math>
```

Eine Formelzeile

Damit jede Formel in eine eigene Zeile geschrieben wird, verwenden wir das Absatz-Tag von HTML: ...

Also zusammen:

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

Eine Anleitung zu MathML

Eine einfache, aber umfassende Anleitung zu MathML findest du auf folgender Internetseite: MathML-Tutorial

2.2 Aufgabenblatt MathML

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.“

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>
                    <mn>6</mn>
                    <mn>8</mn>
                </mfrac>
                <mo>=</mo>
                <mfrac>
                    <mn>3</mn>
                    <mn>4</mn>
                </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?

Aufgabe: Die Umrechung eines Bruchs in eine Dezimalzahl

Füge in die Datei `manuell.html` den Text in der MathML-Sprache ein, so dass die Umrechnung eines Bruchs in eine Dezimalzahl dargestellt wird. Der Text „Hier kannst du . . . “ darf überschrieben werden.

Konkret soll die letzte Zeile so aussehen.

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

Überprüfe das Ergebnis im Firefox-Browser. Die Seite sollte jetzt ungefähr so aussehen:

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

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

Aufgabe: Die Umrechnung einer Dezimalzahl in einen Bruch

Füge in die Datei `manuell.html` den Text in der MathML-Sprache ein, so dass diese Zeile als letzte Zeile angezeigt wird:

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

Überprüfe das Ergebnis im Firefox-Browser. Die Seite sollte jetzt ungefähr so aussehen:

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

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

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

3 Algorithmen und Funktionen

3.1 Algorithmen

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

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:

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

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.

3.2 Variablen

Einfache Variable

Eine Variable dient dazu, sich einen Wert, einen Text oder das Ergebnis einer Rechnung zu merken. Wir geben einer Variablen einen sinnvollen Namen.

```
wichtige_zahl = 42
summe = 2 + 3
```

Mit Variablen können wir auch rechnen.

```
summe = summe + 6
```

Strukturierte Variablen

Wir können Variablen auch komplizierter aufbauen und einen Wert in einer Variablen merken, der sich aus mehreren Teilwerten zusammensetzt. Ein Beispiel ist der Bruch:

```
class Bruch():
    zaehler: int
    nenner: int
```

Ein Bruch ist ein Wert, der sich aus zwei Teilwerten zusammensetzt, nämlich dem Zähler und dem Nenner. Den Bruchstrich müssen wir uns nicht merken, weil er immer da ist. Zähler und Nenner sind ganze Zahlen (`int`).

```
bruch_1 = Bruch(3,4)
```

Jetzt können wir die Variable, die ja einen Bruch enthält, nach dem Zähler und dem Nenner fragen. Wir schreiben die beiden Werte in eigene Variablen.

```
zaehler_1 = bruch_1.zaehler
nenner_1 = bruch_1.nenner
```

Wir können auch zwei Ganzzahlen jeweils in eine Variable schreiben und damit eine Bruchvariable erzeugen.

```
zaehler_2 = 7
nenner_2 = 8
```

```
bruch_2 = Bruch(zaehler_2, nenner_2)
```

3.3 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.

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`.

Eine Beispielfunktion: Das Erweitern eines Bruchs

In 3.1 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
```

Tipp: Datentypen angeben

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

Die Signaturzeile für die Funktion zum Erweitern lautet dann:

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

Der Rumpf bleibt unverändert.

Eine Funktion verwenden

Eine Funktion kann beliebig oft verwendet werden. Oft benutzen wir anstatt „verwenden“ die Formulierung „die Funktion wird aufgerufen“. In der Regel lässt man sich für verschiedene Parameterwerte das jeweilige Ergebnis berechnen.

```
bruch_2 = Bruch(1,2)
bruch_3 = erweitereBruch(bruch_2, 5)
```

Wir können eine Funktion auch innerhalb einer anderen Funktion aufrufen. Das ist sehr hilfreich. Dann kann sich eine Funktion für die Berechnung von Zwischenschritte die Hilfe einer anderen Funktion holen.

3.4 Aufgabe: Das Kürzen eines Bruchs

Der Algorithmus

Wie lautet der Algorithmus zum Kürzen eines Bruchs?

- Nimm einen Bruch entgegen.
-
-
-
-
-

Die Funktion zum Kürzen von Brüchen

Wie lautet die Funktion?

Vermutlich benötigst du eine Funktion zur Berechnung des größten gemeinsamen Teilers von zwei Zahlen. Diese findest du bei den mathematischen Hilfsfunktionen.

```
def kuerzeBruch(bruch: Bruch) -> Bruch:
```

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `spielwiese.py`.

3.5 Aufgabe: Das Multiplizieren von Brüchen

Der Algorithmus

Wie lautet der Algorithmus zum Kürzen eines Bruchs?

- Nimm einen Bruch entgegen.
-
-
-
-
-

Die Funktion zum Kürzen von Brüchen

Wie lautet die Funktion?

Vermutlich benötigst du eine Funktion zur Berechnung des größten gemeinsamen Teilers von zwei Zahlen. Diese findest du bei den mathematischen Hilfsfunktionen.

```
    def kuerzeBruch(bruch: Bruch) -> Bruch:
```

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `spielwiese.py`.

4 Zeichenketten

4.1 Zeichenketten

Was ist eine Zeichenkette?

Eine Zeichenkette enthält ein oder mehrere beliebige Zeichen, also Buchstaben, Zahlen oder Sonderzeichen. Die Zeichen stehen hintereinander. Eine Zeichenkette wird eingeschlossen in Hochkommata (’) oder hochgestellte Anführungszeichen (").

```
eine_zeichenkette = "r2d2"
berg = ’8000er-Gipfel’
ein_ganzer_satz = "Python ist toll!"
```

Häufig benutzen wir den englischen Begriff für Zeichenkette: String. Mit Strings kann man viele interessante Dinge anstellen.

Das Zusammenfügen von Zeichenketten

In Python können zwei Strings mit dem Pluszeichen zu einem langen String zusammengefügt werden:

```
gruss = "Hallo " + "Welt!"
```

In `gruss` steht jetzt der String `"Hallo Welt!"`.

Dass das Pluszeichen in diesem Fall nicht das Addieren von Zahlen, sondern das Zusammenfügen von Strings bedeutet, daran gewöhnt man sich schnell!

Wir möchten einen String zusammenbauen, der in MathML eine Rechenformel beschreibt.

Die Formel

$$2 + 3 = 5$$

lautet in MathML:

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

Diese Formel können wir in Python schrittweise aus kleinen Strings zu einem langen String zusammensetzen.

```
formel = "<math>"
formel = formel + "<mn>2</mn>"
formel = formel + "<mo>+</mo>"
formel = formel + "<mn>3</mn>"
formel = formel + "<mo>=</mo>"
formel = formel + "<mn>5</mn>"
formel = formel + "</math>"
```

Dabei nehmen wir die bereits erstellte Formel, fügen den nächsten kurzen String an und merken uns das Ergebnis wieder in der Variablen `formel`.

4.2 Beispiel: Die Darstellung eines Bruchs in MathML

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 zur Darstellung eines Bruchs in MathML

- Nimm einen Bruch entgegen.
- Wandle den Zähler des Bruchs in eine Zeichenkette und speichere diese Zeichenkette in einer Variablen.
- Verfahre genauso mit dem Nenner.
- Baue die Zeichenkette für das Ergebnis aus Einzelteilen zusammen, wie in den nächsten Schritten beschrieben.
- Beginne die Zeichenkette mit `<mfrac>`.
- Rahme die Zeichenkette, die den Zähler beschreibt, mit den Tags `<mn>` und `</mn>` ein.
- Verfahre ebenso mit der Zeichenkette für den Nenner.
- Beende die Zeichenkette mit `</mfrac>`.
- Gib die Zeichenkette zurück.

Die Funktion zur Darstellung eines Bruchs in MathML

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

5 Das Erstellen der Formeln

5.1 Beispiel: Die Formel zum Erweitern eines Bruchs in MathML

Erledigte Vorarbeiten

- Wir verfügen über die Funktion `erweitereBruch`, die zu einem übergebenen Bruch und einem Faktor den erweiterten Bruch berechnet. (siehe 3.3)
- Außerdem haben wir die Funktion `schreibeBruch`, die einen Bruch in MathML darstellt. (siehe 4.2)

Der Algorithmus zur Darstellung der Erweiterungsformel in MathML

- Nimm einen Bruch und einen Faktor zur Erweiterung entgegen.
- Erzeuge aus diesem Bruch einen erweiterten Bruch. Verwende dazu die Funktion `erweitereBruch`. Merke das Ergebnis in einer Variablen.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion `schreibeBruch`. Merke das Ergebnis in einer Variablen.
- Erzeuge aus dem erweiterten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion `schreibeBruch`. Merke das Ergebnis in einer Variablen.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen wie in den folgenden Schritten angegeben. Verknüpfe die einzelnen Strings mit dem Pluszeichen.
- Es beginnt mit dem MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folgt der MathML-Text für den gekürzten Bruch.

Die Funktion zur Darstellung der Formel in MathML

```
def schreibeErweitern(bruch: Bruch, faktor: int) -> str:
    bruchErweitert = erweitereBruch(bruch, faktor)
    textAlterBruch = schreibeBruch(bruch)
    textNeuerBruch = schreibeBruch(bruchErweitert)
    ergebnis = textAlterBruch + "<mo>=</mo>" + textNeuerBruch
    return ergebnis
```

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `uebungsplatz.py`. Wir erweitern beispielsweise den Bruch $\frac{2}{3}$ mit dem Faktor 3.

```
bruch = Bruch(2,3)
formel = schreibeErweitern(bruch, 3)
print(formel)
```

Die resultierende Formel lautet (allerdings in einer Zeile):

```
<mfrac><mn>2</mn><mn>3</mn></mfrac><mo>=</mo>
                                        <mfrac><mn>6</mn><mn>9</mn></mfrac>
```

Darstellung der Formel im Browser

Wir können die Formel automatisch in die HTML-Seite übertragen lassen und anschließend im Browser anschauen.

Ergänze dazu in der Funktion `schreibeMathML()` folgende Zeilen:

```
inhalt = inhalt + "\n\t\t<p><math>"
inhalt = inhalt + schreibeErweitern(Bruch(2,3), 3)
inhalt = inhalt + "</math></p>"
```

Jetzt führen wir die Datei `htmlErzeugung.py` aus.

Prüfe im Firefox-Browser, wie die Datei `index.html` aussieht. Dort sollte eine Zeile stehen, die so aussieht:

$$\frac{2}{3} = \frac{6}{9}$$

5.2 Aufgabe: Die Formel zum Kürzen eines Bruchs in MathML

Die Vorarbeiten

Hast du die Funktion zum Kürzen eines Bruchs bereit? Wie heißt sie?

Außerdem benötigst du wieder die Funktion zur Darstellung eines Bruchs in MathML.

Der Algorithmus

Wie lautet der Algorithmus für die Formeldarstellung des Kürzens?

- Nimm einen Bruch entgegen.
-
-
-
-
-

Die Funktion

Die Signatur der Funktion lautet:

```
def schreibeKuerzen(bruch: Bruch) -> str:
```

Wie geht es weiter?

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `uebungsplatz.py`. Der Bruch $\frac{6}{8}$ muss gekürzt den Bruch $\frac{3}{4}$ ergeben.

```
<mfrac><mn>6</mn><mn>8</mn></mfrac><mo>=</mo>
    <mfrac><mn>3</mn><mn>4</mn></mfrac>
```

Darstellung der Formel im Browser

Ergänze in der Funktion `schreibeMathML()` die Zeilen für Kürzen für den Bruch $\frac{6}{8}$ analog zum Erweitern.

Führe die Datei `htmlErzeugung.py` aus.

Prüfe im Firefox-Browser, wie die Datei `index.html` aussieht. Dort sollte eine Zeile stehen, die so aussieht:

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

5.3 Aufgabe: Die Formel zum Multiplizieren von Brüchen in MathML

6 Weitere Aufgaben

6.1 Aufgabe: Das Addieren von Brüchen

6.2 Aufgabe: Objektorientierte Programmierung

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

7 Lösungen

7.1 Lösung: Das Erweitern eines Bruchs

Der Algorithmus zum Erweitern

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

Die Funktion zum Erweitern

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

Der Algorithmus zur Darstellung der Erweiterungsformel in MathML

- Nimm einen Bruch und einen Faktor zur Erweiterung entgegen.
- Erzeuge aus diesem Bruch einen erweiterten Bruch. Verwende dazu die Funktion `erweitereBruch`. Merke das Ergebnis in einer Variablen.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion `schreibeBruch`. Merke das Ergebnis in einer Variablen.
- Erzeuge aus dem erweiterten Bruch die Beschreibung in MathML. Verwende dazu ebenfalls die Funktion `schreibeBruch`. Merke das Ergebnis in einer Variablen.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen wie in den folgenden Schritten angegeben. Verknüpfe die einzelnen Strings mit dem Pluszeichen.
- Es beginnt mit dem MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folgt der MathML-Text für den gekürzten Bruch.

Die Funktion zur Darstellung der Formel in MathML

```
def schreibeErweitern(bruch: Bruch, faktor: int) -> str:
    bruchErweitert = erweitereBruch(bruch, faktor)
    textAlterBruch = schreibeBruch(bruch)
    textNeuerBruch = schreibeBruch(bruchErweitert)
    ergebnis = textAlterBruch + "<mo>=</mo>" + textNeuerBruch
    return ergebnis
```

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `uebungsplatz.py`. Wir erweitern beispielsweise den Bruch $\frac{2}{3}$ mit dem Faktor 3.

```
bruch = Bruch(2,3)
formel = schreibeErweitern(bruch, 3)
print(formel)
```

Die resultierende Formel lautet (allerdings in einer Zeile):

```
<mfrac><mn>2</mn><mn>3</mn></mfrac><mo>=</mo>
                                    <mfrac><mn>6</mn><mn>9</mn></mfrac>
```

Darstellung der Formel im Browser

Wir können die Formel automatisch in die HTML-Seite übertragen lassen und anschließend im Browser anschauen.

Ergänze dazu in der Funktion `schreibeMathML()` folgende Zeilen:

```
inhalt = inhalt + "\n\t\t<p><math>"
inhalt = inhalt + schreibeErweitern(Bruch(2,3), 3)
inhalt = inhalt + "</math></p>"
```

Jetzt führen wir die Datei `htmlErzeugung.py` aus.

Prüfe im Firefox-Browser, wie die Datei `index.html` aussieht. Dort sollte eine Zeile stehen, die so aussieht:

$$\frac{2}{3} = \frac{6}{9}$$

7.2 Lösung: Das Kürzen eines Bruchs

Der Algorithmus zum Kürzen eine Bruchs

- Nimm einen Bruch.
- Merke dir den Zähler des Bruchs in einer Variablen.
- Verfahre ebenso mit dem Nenner.
- Berechne den größten gemeinsamen Teiler (ggT) von Zähler und Nenner. (Es gibt dafür eine fertige Funktion.)
- Berechne den gekürzten Zähler, indem du den alten Zähler durch den ggT teilst.
- Berechne ebenso den gekürzten Nenner.
- Erstelle einen neuen Bruch aus dem gekürzten Zähler und dem gekürzten Nenner.
- Gib den neuen Bruch zurück.

Die Funktion zum Kürzen von Brüchen

Die Berechnung des ggT erfolgt mittels der Funktion `mathefunktionen.berechne_ggT`. In der Funktion zum Kürzen wird also eine Zeile enthalten sein:

```
ggT = mathefunktionen.berechne_ggT(alterZaehler, alterNenner)
```

Beachte, dass beim Teilen durch den ggT die Division `//` benutzt wird, damit der neue Zähler und der neue Nenner ganze Zahlen werden!

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

Die Algorithums zur Darstellung der Formel zum Kürzen

- Nimm einen Bruch.
- Erzeuge aus diesem Bruch einen gekürzten Bruch. Verwende dazu die Funktion, die du gerade programmiert hast.
- Erzeuge aus dem ursprünglichen Bruch die Beschreibung in MathML. Verwende dazu die Funktion, die einen Bruch in MathML beschreibt.

- Erzeuge aus dem gekürzten Bruch ebenfalls die Beschreibung in MathML. Das geht genauso wie eben.
- Setze jetzt den MathML-Text aus den einzelnen Angaben zusammen.
- Die Zeichenkette beginnt mit dem MathML-Text für den ungekürzten Bruch.
- Dann kommt das Gleichheitszeichen.
- Es folge der MathML-Text für den gekürzten Bruch.

Die Funktion zur Darstellung der Formel zum Kürzen

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

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei `uebungsplatz.py`.

```
bruch = Bruch(6,8)
formel = schreibeKuerzen(bruch)
print(formel)
```

Der Bruch $\frac{6}{8}$ muss gekürzt den Bruch $\frac{3}{4}$ ergeben. In einer Zeile steht:

```
<mfrac><mn>6</mn><mn>8</mn></mfrac><mo>=</mo>
    <mfrac><mn>3</mn><mn>4</mn></mfrac>
```

Ergänze in der Funktion `schreibeMathML()` die Zeilen für Kürzen für den Bruch $\frac{6}{8}$.

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

Führe die Datei `htmlErzeugung.py` aus.

Prüfe im Firefox-Browser, wie die Datei `index.html` aussieht. Dort sollte eine Zeile stehen, die so aussieht:

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

7.3 Lösung: Das Multiplizieren von Brüchen