Mathehelfer Bruchrechnen

Ein Hackathon für Kinder und Jugendliche

Norbert Seulberger

Mai 2023

Inhaltsverzeichnis

1 Fahrplan für den Kurs

Der Fahrplan zur Orientierung

MathML

MathML: Einführung

Lesen und verstehen!

Algorithmen

Variablen

Funktionen

Beispiel Kürzen

Aufgabe Erweitern

Aufgabe Multiplizieren

Zeichenketten

Formel Kürzen

Formel Erweitern

Formel Multiplizieren

2 MathML

2.1 MathML: Einführung

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.
- `<mo>&middot;</mo>` schreibt das Multiplikationszeichen.

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>
```

2.2 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

2.3 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: Das Erweitern eines Bruchs

Füge in die Datei

`manuell.html`

den Text in der MathML-Sprache ein, so dass das Erweitern eines Bruchs dargestellt wird. Der Text „Hier kannst du . . . “ darf überschrieben werden.

Die letzte Zeile soll so aussehen:

$$\frac{3}{4} = \frac{9}{12}$$

Überprüfe das Ergebnis im Firefox-Browser. Die Datei `manuell.html` sollte jetzt so dargestellt werden:

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

$$\frac{3}{4} = \frac{9}{12}$$

Aufgabe: Das Multiplizieren von zwei Brüchen

Füge in die Datei

`manuell.html`

den Text in der MathML-Sprache ein, so dass diese Zeile als letzte Zeile angezeigt wird:

$$\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$$

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

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

$$\frac{3}{4} = \frac{9}{12}$$

$$\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$$

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 $\frac{3}{4}$ mit dem Faktor 3 erweitert, so ergibt sich diese Gleichung:

$$\frac{3}{4} = \frac{3 \cdot 3}{4 \cdot 3} = \frac{9}{12}$$

Der Algorithmus für das Erweitern

- Nimm einen Bruch und den Faktor, mit dem du den Bruch erweitern willst.
- Merke dir den Zähler des Bruchs in einer Variablen.
- Merke dir den Nenner des Bruchs in einer zweiten 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.
- Gib den neuen Bruch zurück.

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`).

Wir merken uns den Zähler und Nenner zusammen in einem Bruch. Diesen Bruch können wir dann als einen Wert in einer Variablen aufbewahren.

```
bruch = 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 = bruch.zaehler
nenner = bruch.nenner
```

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

```
neuer_zaehler = 7
neuer_nenner = 8
bruch = Bruch(neuer_zaehler, neuer_nenner)
```

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 die gleichen Eingabewerte erhält man immer dasselbe Ergebnis. Die Eingabewerte heißen Parameter. Es kann einen oder mehrere Parameter geben, manchmal sogar gar keinen.

Der Aufbau einer Funktion in Python

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

- die Signaturzeile
- der Rumpf der Funktion
- der Rückgabewert

Die Signaturzeile

Die Signaturzeile nennt den Namen und beschreibt die Parameter.

```
def kuerzeBruch(bruch):
```

- Zuerst steht immer das Wort `def`.
- Dann folgt der Name der Funktion. Der Name sollte ausdrücken, welche Berechnung die Funktion durchführt.
- In den runden Klammer stehen die Eingabewerte für die Funktion. Hier ist es ein Parameter mit dem Namen `bruch`. Die Funktion kann den Wert des Parameters für die Berechnung nutzen.

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 Kürzen lautet dann:

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

Die Funktion nimmt also einen Parameter entgegen, dessen Name `alterBruch` lautet und dessen Datentyp ein `Bruch` ist (groß geschrieben!).

Die Funktion führt eine Berechnung durch, bei der ein Rückgabewert berechnet wird, der vom Datentyp `Bruch` ist. Das steht hier: `-> Bruch`.

Der Rumpf einer Funktion

Der Rumpf der Funktion ist eingerückt. Wir benutzen dafür vier Leerzeichen.

Hier wird die Berechnung durchgeführt. Das erklären wir ausführlich im nächsten Beispiel.

Der Rückgabewert

Die letzte Zeile im Rumpf gibt das Ergebnis zurück. Sie beginnt mit dem Wort

```
return
```

3.4 Beispiel: Das Kürzen eines Bruchs

Im folgenden Beispiel lernst du das Programmieren einer Funktion in Python.

Aufgabe: Der Algorithmus zum Kürzen eines Bruchs

Lies den Algorithmus zum Kürzen eines Bruchs durch:

- Nimm einen Bruch.
- Merke dir den Zähler des Bruchs in einer Variablen.
- Merke dir den Nenner des Bruchs in einer anderen Variablen.
- 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.

Verstehst du das Vorgehen? Andernfalls stelle deine Fragen dem Betreuer.

Der größte gemeinsame Teiler zweier Zahlen (ggT)

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)
```

Die Funktion zum Kürzen von Brüchen

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

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

Aufgabe: Programmieren der Funktion

Wir bearbeiten jetzt die Datei

`mathehelfer.py`

hinter der Zeile

`# Ab hier dürft ihr Änderungen vornehmen.`

Gib die Funktion in den Editor der Entwicklungsumgebung (IDE) ein. Zeigt die IDE noch Fehler an? Vielleicht hast du dich vertippt? Falls du den Fehler nicht finden kannst, sprich den Betreuer an.

Überprüfung des Ergebnisses

Wir überprüfen das Ergebnis mit Hilfe folgender Datei:

`uebungsplatz.py`

Wir tragen dort die folgenden Zeilen ein:

```
bruch = Bruch(6,8)
gekuerzter_bruch = mathehelfer.kuerzeBruch(bruch)
print(gekuerzter_bruch)
```

Ausführen eines Python-Programms

Jetzt führen wir `uebungsplatz.py` als Python-Programm aus. Das geht in VS Code auf zwei verschiedene Weisen:

- Im Terminal: `python3 uebungsplatz.py`
- Klicke mit der Maus rechts oben in der Ecke auf ▷ und wähle *Führen Sie die Python-Datei aus*.

Das Ergebnis sollte jetzt diese Zeile enthalten:

```
Bruch(zaehler=3, nenner=4)
```

Sieht das Ergebnis bei dir genauso aus?

3.5 Aufgabe: Das Erweitern eines Bruchs

Der Algorithmus zum Erweitern

Wie lautet der Algorithmus zur Erweitern eines Bruchs mit einem vorgegebenen Faktor?

- Nimm einen Bruch und den Faktor, mit dem du den Bruch erweitern willst.
-
-
-
-
-
-

Die Funktion zum Erweitern

Programmiere die Funktion zum Erweitern eines Bruchs. Die Funktion erhält als Parameter einen Bruch sowie den Faktor zum Erweitern. Die Signatur lautet:

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

Vergiss nicht die letzte Zeile mit dem `return`-Befehl!

Überprüfung des Ergebnisses

Überprüfe das Ergebnis. Trage dazu die notwendige Befehle in die Datei

`uebungsplatz.py`

ein. Du kannst dich daran orientieren, wie wir es beim Kürzen gemacht haben. Hier möchten wir den Bruch $\frac{3}{4}$ mit dem Faktor 3 erweitern.

Es geht los mit

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

Wie geht es weiter?

Danach führen wir `uebungsplatz.py` als Python-Skript aus. Das Ergebnis sollte jetzt diese Zeile enthalten:

```
Bruch(zaehler=9, nenner=12)
```

Sieht das Ergebnis bei dir genauso aus?

3.6 Aufgabe: Das Multiplizieren von Brüchen

Der Algorithmus zum Multiplizieren von Brüchen

Wie lautet der Algorithmus zum Multiplizieren von Brüchen? Denke daran, den neuen Bruch zu kürzen!

- Nimm zwei Brüche entgegen.
-
-
-
-
-
-
-
-

Die Funktion zum Multiplizieren von Brüchen

Wie lautet die Funktion?

Überprüfung der Funktion zum Multiplizieren

Wir möchten in der Datei

`uebungsplatz.py`

die beiden Brüche $\frac{3}{4}$ und $\frac{2}{3}$ miteinander multiplizieren. Es beginnt also mit

```
erster_bruch = Bruch(3,4)
zweiter_bruch = Bruch(2,3)
```

Wie geht es weiter?

Führe `uebungsplatz.py` als Python-Skript aus. Die Ausgabe muss folgende Zeile enthalten:

Produkt: Bruch(zaehler=1, nenner=2)

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!"`.

Tipp: Das Pluszeichen setzt Zeichenketten zusammen!

Das Pluszeichen bedeutet in diesem Fall nicht das Addieren von Zahlen, sondern das Zusammenfügen von Zeichenketten. Daran gewöhnt man sich schnell!

MathML als Zeichenkette

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 Kürzen eines Bruchs in MathML

Die Vorarbeiten

- Wir verfügen über die Funktion `kuerzeBruch`. (siehe 3.4)
- Außerdem haben wir die Funktion `schreibeBruch`, die einen Bruch in MathML darstellt. (siehe 4.2)

Der Algorithmus 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

In der Datei **`mathehelfer.py`** programmieren wir folgende Funktion:

```
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 = mathehelfer.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>
```

MathML in Python schreiben

Jetzt arbeiten wir wieder in der Datei **mathehelfer.py**.

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>"
```

Die HTML-Seite erzeugen und ansehen

Führe die Datei **htmlErzeugung.py** aus.

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

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

5.2 Aufgabe: Die Formel zum Erweitern eines Bruchs in MathML

Erledigte Vorarbeiten

Welche bereits erstellten Funktionen können wir sinnvoll nutzen?

-
-

Der Algorithmus zur Darstellung der Erweiterungsformel in MathML

Wie lautet der Algorithmus?

- Nimm einen Bruch und einen Faktor zur Erweiterung entgegen.
-
-
-
-
-
-
-

Die Funktion zur Darstellung der Formel in MathML

Wie lautet die Funktion? Die Signatur sieht so aus:

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

Wie geht es weiter?

Überprüfung des Ergebnisses

Überprüfe das Ergebnis in der Datei

`uebungsplatz.py`.

Wir erweitern beispielsweise den Bruch $\frac{3}{4}$ mit dem Faktor 3.

```
bruch = Bruch(2,3)
faktor = 3
```

Wie geht es weiter? Wie sieht das Ergebnis aus?

Darstellung der Formel im Browser

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

Ergänze in der Funktion

`schreibeMathML()`

die entsprechenden Zeilen analog zum Beispiel Kürzen.

Dann führe die Datei

`htmlErzeugung.py`

als Python-Skript aus.

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

$$\frac{3}{4} = \frac{9}{12}$$

5.3 Aufgabe: Die Formel zum Multiplizieren von Brüchen in MathML

Der Algorithmus zur Darstellung der Multiplikationsformel

Wie lautet der Algorithmus, um die Multiplikation zweier Brüche in MathML darzustellen?

- Nimm zwei Brüche entgegen.
-
-
-
-
-
-
-
-

Die Funktion zur Darstellung der Multiplikationsformel

Wie lautet die Funktion?

Überprüfung der Darstellung

Ergänze in der Funktion `mathehelfer.schreibeMathML()` die Zeilen für das Multiplizieren. Wie lauten sie?

Führe die Datei `htmlErzeugung.py` aus.

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

$$\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$$