Arbeiten mit der Rechenaufgabe

In diesem Dokument soll die Verwendung des Bausteins „Rechenaufgabe“ näher erläutert werden. Mithilfe dieses Bausteins können Sie Rechenaufgaben schnell verändern - und sich dabei automatisch die neuen Ergebnisse berechnen lassen.

1 Aufbau des Bausteins

In diesem ersten Abschnitt sollen zunächst die Bestandteile des Bausteins und ihre Funktionen vorgestellt werden. Dabei erfahren Sie auch, welche Einstellungsmöglichkeiten Sie haben.

1.1 Allgemeine Einstellungen

Wenn Sie eine Rechenaufgabe aus dem Menü auf das Arbeitsblatt ziehen, erhalten Sie die folgende Beispiel-Aufgabe:

① Berechne!

a) 5 + ☐ = 12	d) 5 + ☐ = 15	g) 8 + ☐ = 17
b) 3 + ☐ = 9	e) 9 + ☐ = 18	h) 7 + ☐ = 15
c) 6 + ☐ = 9	f) 7 + ☐ = 12	i) 8 + ☐ = 14

Öffnen Sie den Baustein mit einem Doppelklick. Im Textfeld „Aufgabentext“ können Sie die Beschreibung der Aufgabe ändern (s. Abb. 1). Die Schaltflächen darüber erlauben es, die Nummerierung an- bzw. auszuschalten (standardmäßig ist sie immer aktiviert) bzw. eine neue Nummerierung zu beginnen (bei Aktivierung wird die Nummerierung zurückgesetzt, sodass sie wieder bei 1 beginnt). Zusätzlich können auch Punkte und ein Schwierigkeitsgrad für die Aufgabe vergeben werden (diese Features werden erst aktiviert, wenn die Zähler auf mindestens 1 erhöht werden). Unterhalb des Aufgabentextes lässt sich die Art der Aufzählung der Teilaufgaben anpassen.

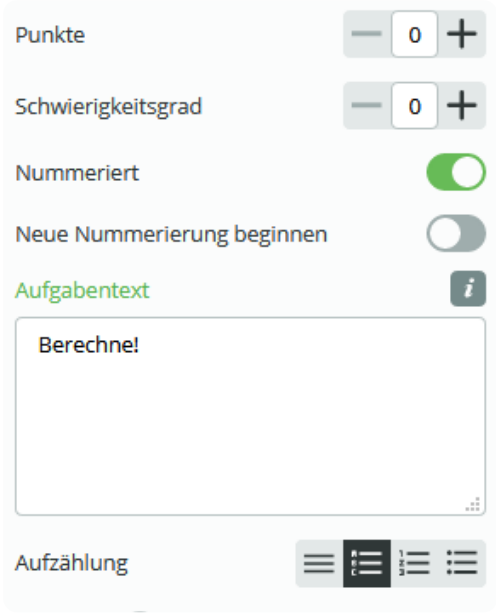


Abb. 1

1.1 Variablen

Abb. 2

Direkt darunter befinden sich die Variablen (s. Abb. 2). Sie sind die wichtigsten Elemente im Rechenaufgabe-Baustein, denn sie ermöglichen es, nach festgelegten Regeln, Zahlen zufällig zu generieren.
Der Name einer Variable beginnt immer mit #, gefolgt von einer Zeichenkette aus Buchstaben und Zahlen. Sonderzeichen sind nicht erlaubt bzw. werden nicht als Bestandteil eines Variablennamens erfasst.

In der Spalte „Regel“ werden die Variablen mit mathematischen Ausdrücken definiert. Zusätzlich zu den gewöhnlichen Rechenoperatoren (in Zeichen +, -, *, /) kann eine vielfältige Auswahl an Konstanten und Funktionen verwendet werden (eine Übersicht hierzu finden Sie im Anhang). Auf diese Weise können aus bereits bestehenden Variablen neue Variablen gebildet werden (wie hier in der Variable #c). Außerdem haben Sie die Möglichkeit, durch die Eingabe des Buchstaben „x“ eine Variable als Zufallsvariable zu definieren. Diese wird dann in dem durch „min“ und „max“ festgelegten Bereich „gewürfelt“.
In der letzten Spalte können Sie festlegen, wie viele Ziffern nach dem Komma ausgegeben werden sollen.

1.1 Einstellung der Teilaufgaben

Unterhalb der Variablen finden Sie das Textfeld „Textausgabe“ (s. Abb. 3). Hier wird der Text bzw. die Formel eingegeben, die in jeder Teilaufgabe erscheinen soll. In diesem Feld haben Sie unter anderem die Möglichkeit, Lücken zu verwenden, deren Inhalt nur auf dem Lösungsblatt angezeigt wird (geschrieben in eckigen Klammern). Des Weiteren können Sie LaTeX-Ausdrücke für mathematische Symbole benutzen (umfassende Informationen dazu finden Sie in der LaTeX-Sammlung). Darüber hinaus gibt es noch weitere Formatierungsmöglichkeiten, wie bspw. Farben. Wenn Sie den Mauszeiger über dem Eingabefeld kurz stehen lassen, erscheint eine Info-Box mit den entsprechenden Markierungszeichen für die jeweiligen Formatierungen.
Mit der Schaltfläche „Regel hinzufügen“ können Sie eine alternative Textausgabe erstellen (mehr dazu in Abschnitt 3). Direkt darunter können Sie die Variablen neu auswürfeln und die Anzahl der Spalten und Zeilen für die Teilaufgaben festlegen.

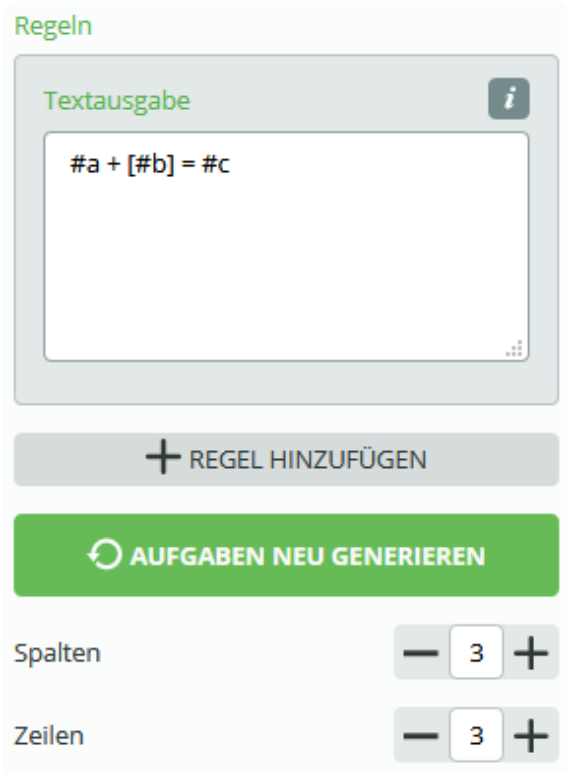
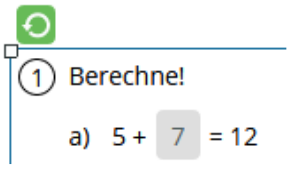


Abb. 3

Den grünen Schalter **Aufgaben neu generieren** finden Sie auch in der linken oberen Ecke direkt am Baustein selbst.



1 Variablen anlegen und verwenden

Kommen wir nun zum wichtigsten Teil: den Variablen. Für das Anlegen einer Variable gibt es keinen eigenen Schalter. Stattdessen wird eine Variable erzeugt, sobald sie innerhalb der Rechenaufgabe genutzt wird. Das heißt, sobald Sie eine Variable in eine Textausgabe schreiben, wird sie automatisch (wenn sie nicht bereits existiert) oben in der Liste der Variablen als Zufallszahl zwischen 1 und 10 angelegt. Dasselbe geschieht, wenn Sie eine neue Variable (immer beginnend mit #) in die Definition einer bereits bestehenden Variable einfügen.

Nehmen Sie z.B. den Baustein von der ersten Seite und fügen Sie in die Definition der Variablen #c am Ende noch „+ #d“ an (vgl. Abb. 4). Die Variable #d wird nun automatisch als Zufallsvariable zwischen 1 und 10 neu angelegt. Die Textausgabe muss dann natürlich noch entsprechend angepasst werden (zum Aktualisieren auf das grüne Feld links oben klicken).

Abb. 4

Variablen umbenennen

Auf der anderen Seite wird eine Variable sofort gelöscht, sobald sie **nicht** mehr verwendet wird (also sie weder in einer der Textausgaben noch in einer Variablendefinition vorkommt). Mit ihr werden dann auch die evtl. vorgenommenen Einstellungen gelöscht. Bevor Sie eine Variable aus der Textausgabe oder einer „Regel“ löschen, sollten Sie sich daher kurz überlegen, ob Sie die Variable noch brauchen.

Das gilt auch, wenn Sie Variablen **umbenennen** oder die Reihenfolge in der Ausgabe verändern wollen. Nehmen wir wieder das Eingangsbeispiel mit der obigen Anpassung (wie in Abb. 4):

② Berechne!

a) 1 + ☐ + 4 = 15	d) 3 + ☐ + 3 = 11	g) 4 + ☐ + 2 = 13
b) 3 + ☐ + 6 = 14	e) 5 + ☐ + 6 = 19	h) 4 + ☐ + 5 = 13
c) 2 + ☐ + 3 = 7	f) 4 + ☐ + 7 = 14	i) 8 + ☐ + 5 = 20

Angenommen, wir wollen #d und #c in der Ausgabe vertauschen. Wenn wir nun #c aus der Textausgabe entfernen, verschwindet die Variable einfach, weil sie dann nirgendwo mehr vorkommt. Die Variable müsste dann neu angelegt und definiert werden (drücken Sie STRG+Z um die Löschung rückgängig zu machen). Um dies zu vermeiden, können Sie stattdessen zuerst #d in der Ausgabe in #c umbenennen (Abb. 5). Die Variable #d verschwindet dadurch nicht, weil sie noch in der Definition von #c verwendet wird. Nun können Sie wiederum nach dem Gleichheitszeichen #c in #d umbenennen (Abb. 6). Dies verursacht nun keine Probleme mehr, da #c jetzt auf der linken Seite noch vorkommt. Damit die Rechnung auch stimmt, müssten nun natürlich noch die Definitionen von #c und #d entsprechend geändert werden (Abb. 7).

Abb. 5

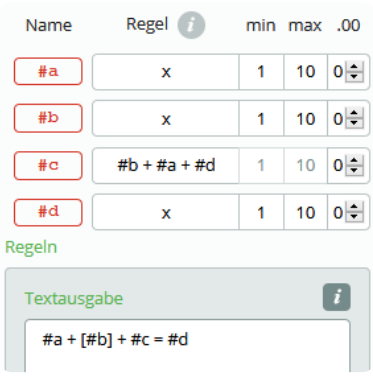


Abb. 6

Abb. 7

1 Verschiedene Textausgaben verwenden

Sie haben in der Rechenaufgabe auch die Möglichkeit, verschiedene Textausgaben miteinander zu kombinieren. Das eröffnet eine Vielzahl an Möglichkeiten, eine Aufgabe interessanter zu gestalten: So können z.B. die Position der Lücken oder die Anzahl der Nachkommastellen variiert, Gleichungen in verschiedenen Stellungen ausgegeben, oder auch unterschiedliche Zahlenbereiche miteinander kombiniert werden. Im Nachfolgenden finden Sie einige Beispiele hierzu.

1.1 Position der Lücken variieren

Öffnen Sie wieder das Eingangsbeispiel (oder ziehen Sie eine neue Rechenaufgabe auf das Arbeitsblatt; dafür können Sie auch z.B. oben rechts eine neue leere Seite einfügen). Gehen Sie zur Textausgabe und klicken Sie auf "Neue Regel hinzufügen". Es erscheint ein zweites Eingabefeld, das auch schon eine alternative Ausgabe beinhaltet. Anstatt dem zweiten Summanden ist jetzt das Ergebnis auf der rechten Seite als Lücke gesetzt (vgl. Abb. 8). Wenn Sie nun die Aufgabe neu generieren, werden die erste und die zweite Textausgabe gemischt.

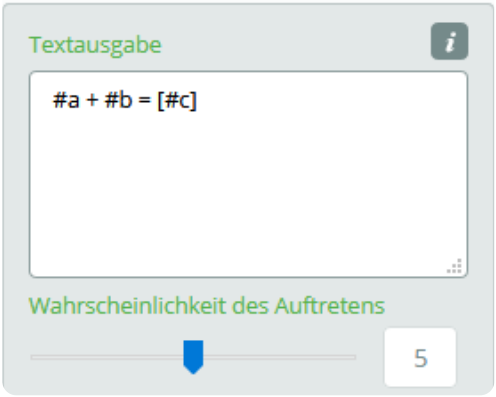


Abb. 8 — Neue automatisch generierte Textausgabe

Wahrscheinlichkeit des Auftretens regulieren

Aller Voraussicht nach wird dabei die neue Variante häufiger vorkommen als die alte. Das liegt daran, dass die „Wahrscheinlichkeit des Auftretens“ bei einer neuen Textausgabe immer automatisch auf 5 steht, während sie bei der ersten noch den Wert 1 hat. Die Skala reicht von 0 bis 10 und dient dazu, die Häufigkeiten der einzelnen Ausgabevarianten zu regulieren. Der Begriff „Wahrscheinlichkeit“ sollte dabei nicht zu genau genommen werden - der Wert 10 steht nicht für die Wahrscheinlichkeit 1. Sicher ist jedoch, dass bei einem Wert von 0 eine Ausgabe gar nicht vorkommt - so kann man temporär eine Variante „ausschalten“, ohne sie löschen zu müssen. Wenn Sie die beiden Regler auf den gleichen Wert stellen (z.B. beide auf 5) und auf „aktualisieren“ klicken, sollten die beiden Varianten einigermaßen gleich verteilt sein (es kann natürlich Schwankungen in die eine oder andere Richtung geben - besonders bei einer niedrigen Anzahl von Teilaufgaben).
Zur Übung können Sie noch eine dritte Alternative hinzufügen, bei der der erste Summand in der Lücke steht. Dann könnte die Aufgabe z.B. so aussehen:

③ Berechne!

a) ☐ + 6 = 8	d) ☐ + 4 = 12	g) 3 + ☐ = 13
b) 7 + ☐ = 16	e) 9 + 4 = ☐	h) ☐ + 2 = 5
c) 6 + 5 = ☐	f) ☐ + 6 = 16	i) 4 + ☐ = 6

1.1 Verschiedene Zahlenbereiche kombinieren

Wir wollen uns an einem einfachen Beispiel anschauen, wie mit der Rechenaufgabe verschiedene Zahlenbereiche genutzt werden können. Das kann z.B. für Aufgaben in der Grundschule hilfreich sein, wo man oft innerhalb von bestimmten Zahlenbereichen bleiben möchte. Aber auch positive und negative Zahlen können so kombiniert werden.
Wir gehen wieder von der „Standard-Rechenaufgabe“ aus (mit der Formel #a + #b = #c) und wollen sie so verändern, dass die Zahlen alle zwischen 1 und 100 liegen. Der Einfachheit halber nehmen wir zunächst immer #b für die Lücke.
Legen Sie eine neue Rechenaufgabe an und setzen Sie die obere Grenze für die Variablen #a und #b jeweils auf 50 (vgl. Abb. 9). Die Summe #c liegt dann natürlich immer im Bereich 1 bis 100. Die Aufgabe erfüllt also schon unsere Bedingung, sich im „Zahlenraum 100“ zu bewegen - allerdings kommt auf diese Weise kein Summand vor, der größer ist als 50. Wenn man es variantenreicher gestalten möchte, braucht man also noch zusätzliche Variablen.

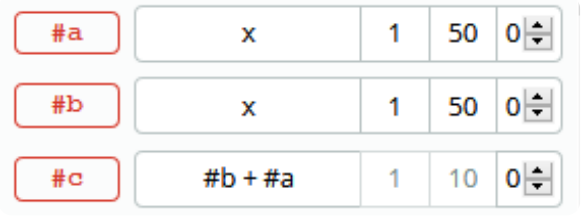


Abb. 9

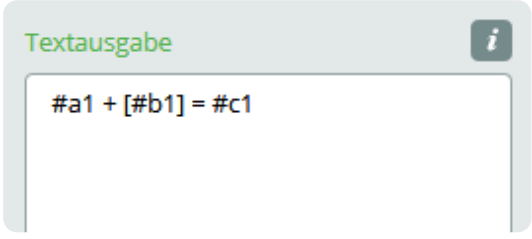


Abb. 10

Wir fügen eine neue Textausgabe hinzu, löschen den dortigen Text und geben die Formel #a1 + [#b1] = #c1 ein (Abb. 10). Bei der Variable #a1 setzen wir den Zahlenbereich auf 50 (min) bis 65 (max) und bei #b1 setzen wir 35 als obere Grenze, sodass die Summe auch hier wieder zwischen 1 und 100 liegt. Dementsprechend definieren wir #c1 durch #a1 + #b1 (vgl. Abb. 11).

#a1	x	50	65	0
#b1	x	1	35	0
#c1	#a1 + #b1	1	10	0

Abb. 11

#a2	x	65	80	0
#b2	x	1	20	0
#c2	#a2 + #b2	1	10	0

Abb. 12

Schließlich ergänzen wir noch die Ausgabe #a2 + [#b2] = #c2 und legen hier die Zahlenbereiche 65-80 für #a2 und 1-20 für #b2 fest. Außerdem definieren wir natürlich #c2 durch #a2 + #b2 (s. Abb 12).
Man könnte noch weitere solcher Varianten erstellen, auch mit feinerem Raster, etwa in 10er-Schritten, um noch mehr mögliche Zahlenpaare (#a und #b) abzudecken. Hier muss man letztendlich abwägen, ob es den Aufwand lohnt - denn alle zulässigen Zahlenpaare abzubilden, ist ohnehin kaum möglich.
Es ergibt sich dann etwa das folgende Bild:

④ Berechne!

a) 37 + ___ = 50
b) 59 + ___ = 79
c) 68 + ___ = 73
d) 46 + ___ = 66
e) 61 + ___ = 94
f) 51 + ___ = 67
g) 69 + ___ = 76
h) 44 + ___ = 52
i) 8 + ___ = 17

Da die Zahlenbereiche bei den ersten beiden Zufallsvariablen deutlich größer sind als in den anderen beiden Fällen, wurde hier die Wahrscheinlichkeit doppelt so hoch angesetzt, damit die Zahlenbereiche einigermaßen gleichmäßig verteilt vorkommen.

1.1 Anzahl der Nachkommastellen variieren

Ähnlich wie bei den Zahlenbereichen, können auch die Nachkommastellen variiert werden. Wir betrachten dazu das folgende Beispiel einer Rechenaufgabe, bei der Gewichte zwischen g und kg umgerechnet werden sollen:

⑤ Fülle die Lücken.

a) ___ kg = 3902 g
b) 1,396 kg = ___ g
c) ___ kg = 5200 g
d) 4,7 kg = ___ g
e) ___ kg = 7120 g
f) 3,092 kg = ___ g

Die Variablen #a1, #a2 und #a3 sind dabei als Zufallsvariablen zwischen 0 und 10 definiert - mit einer, zwei, bzw. drei Ziffern nach dem Komma (vgl. Abb. 13). Sie repräsentieren die Werte in kg. Da die Anzahl der Nachkommastellen für jede einzelne Variable festgelegt ist, braucht es für jeden Fall eine separate Variable. Dementsprechend werden für die Umrechnung in g auch nochmals drei Variablen (#b1, #b2 und #b3) benötigt - da in jeder Rechenvorschrift eine andere #a-Variable zugrunde gelegt wird.
Folglich braucht es für jeden Fall auch eine eigene Textausgabe. Hier wurde für jedes der drei „#a#b-Paare“ noch eine zweite Ausgabe angelegt, sodass die Lücke mal auf der linken und mal auf der rechten Seite auftaucht - ingesamt sind es in diesem Beispiel also 6 verschiedene Textausgaben.

Name	Regel	min	max	.00
#a1	x	0	10	1
#b1	#a1 * 1000	1	10	0
#a2	x	0	10	2
#b2	#a2 * 1000	1	10	0
#a3	x	0	10	3
#b3	#a3 * 1000	1	10	0

Abb. 13

1 Tipps & Tricks

In diesem Abschnitt geht es um einige nützliche Tipps, die Ihnen das Erstellen von Rechenaufgaben erleichtern sollen.

1.1 „Rückwärts rechnen“

Wenn man eine Rechenaufgabe erstellt, ist man geneigt, die gegebenen Zahlen zufällig auszuwürfeln und mit ihnen das Ergebnis bzw. die gesuchte Zahl zu berechnen. Oft ist das auch zielführend, z.B. in der folgenden Aufgabe:

⑥ Berechne!

a) $4 \cdot 10 =$ ____ c) $2 \cdot 7 =$ ____ e) $5 \cdot 5 =$ ____

b) $5 \cdot 7 =$ ____ d) $7 \cdot 3 =$ ____ f) $9 \cdot 2 =$ ____

Die Faktoren #a und #b werden zufällig generiert und mit ihnen das Ergebnis #c berechnet. Nun wollen wir, anstatt zu multiplizieren, dividieren. Man könnte auch wieder versuchen, die gegebenen Zahlen, also Dividend und Divisor, als Zufallsvariable zu definieren und daraus den Quotienten berechnen. Problematisch ist nur, dass durch die Zufälligkeit der beiden Zahlen es nicht gewährleistet ist, dass die Division immer ohne Rest möglich ist (was wir hier aber erreichen wollen).
Wesentlich einfacher und effizienter ist es in diesem Fall, die gesuchte Zahl (d.h. den Quotienten) und den Teiler auszuwürfeln und daraus mittels Multiplikation den Dividenden zu ermitteln. Wir gehen also so zu sagen vom Ergebnis aus und rechnen „zurück“. Vorteilhaft ist diese Methode nicht zuletzt deshalb, weil die obige Aufgabe mit den gleichen Konfigurationen komplett wieder verwendet werden kann - es muss nur die Ausgabe entsprechend angepasst werden:

⑦ Berechne!

a) $60 : 6 =$ c) $30 : 6 =$ e) $15 : 3 =$

b) $25 : 5 =$ d) $28 : 4 =$ f) $18 : 2 =$

1.1 Hilfsvariablen

Manchmal möchte man, dass Zahlen eine bestimmte Eigenschaft haben - z.B. Vielfache einer bestimmten Zahl zu sein. In solchen Fällen ist es sinnvoll, eine oder mehrere Hilfsvariablen zu verwenden.

Beispiel 1: Runde Zahlen

Angenommen, wir möchten eine Additionsaufgabe mit Zahlen bis 1000 erstellen, wollen aber, dass die Summanden Vielfache von 10 sind. Wir verwenden die Variablen #a und #b als Summanden und definieren sie als Zehnfache der Hilfsvariablen #x1 bzw. #x2 (s. Abb 14). Diese wählen wir als Zufallsvariablen, z.B. zwischen 10 und 50 (sodass die Summanden #a und #b zwischen 100 und 500 liegen). In Aufgabe 8 können Sie sich das Beispiel anschauen.

#a	#x1 * 10	1	10	0
#b	#x2 * 10	1	10	0
#c	#b + #a	1	10	0
#x1	x	10	50	0
#x2	x	10	50	0

Abb. 14

⑧ Berechne!

a) 400 + 350 = d) 400 + 300 = g) 450 + 480 =

b) 340 + 280 = e) 280 + 150 = h) 260 + 380 =

c) 130 + 200 = f) 150 + 360 = i) 120 + 440 =

Beispiel 2: Rechnen mit runden Cent-Beträgen

Als weiteres Beispiel wollen wir eine Aufgabe erstellen, bei der zwei gebrochene €-Beträge addiert werden sollen. Damit die Rechnungen allerdings nicht zu kompliziert werden, sollen die Cent-Beträge 10er-Zahlen sein. Am einfachsten ist das zu erreichen, indem man ganze Zahlen als Hilfsvariablen würfelt (hier #x1 und #x2) und diese durch 10 teilt (vgl. Abb. 15). Dadurch bekommt man automatisch Summanden (#a und #b), die nur eine Stelle nach dem Komma haben. Wir stellen dort jedoch zwei Nachkommaziffern ein, da wir ja mit Geldbeträgen rechnen wollen. Beim Ergebnis (#c) muss das auch separat eingestellt werden. Aufgabe 9 zeigt das fertige Beispiel.

#a	#x1 / 10	1	10	2
#b	#x2 / 10	1	10	2
#c	#b + #a	1	10	2
#x1	x	1	100	0
#x2	x	1	100	0

Abb. 15

⑨ Berechne!

a) 2,70 € + 4,50 € = ____ €

b) 6,10 € + 2,20 € = ____ €

c) 2,60 € + 0,70 € = ____ €

d) 9,00 € + 8,90 € = ____ €

e) 2,00 € + 1,00 € = ____ €

f) 2,60 € + 6,70 € = ____ €

Beispiel 3: Brüche kürzen

Als drittes und letztes Beispiel betrachten wir das Kürzen von Brüchen. Wenn wir Zähler und Nenner rein zufällig wählen, kann es natürlich vorkommen, dass diese teilerfremd sind und sich der Bruch gar nicht kürzen lässt. Deshalb bietet es sich auch hier an, Hilfsvariablen zu verwenden. Diese sind so zu sagen die „vorläufigen“ (zufällig generierten) Zähler und Nenner (im Beispiel #z und #n), welche beide noch mit derselben Zufallszahl #x multipliziert werden (s. Abb. 16). Dadurch steht auf der linken Seite immer ein mit #x erweiterter Bruch, der in jedem Fall gekürzt werden kann. Dafür teilt man den Zähler #z und den Nenner #n jeweils durch den größten gemeinsamen Teiler von #z und #n. Diesen erhält man mit der math.js-Funktion gcd (*greatest common divisor*).

#zaehler	#z * #x	1	10	0
#nenner	#n * #x	1	10	0
#zkurz	#z / gcd(#z, #n)	1	10	0
#nkurz	#n / gcd(#z, #n	1	10	0
#z	x	1	10	0
#n	x	1	10	0
#x	x	2	10	0

Abb. 16

⑩ Kürze die Brüche soweit wie möglich.

a) $\frac{20}{8} =$ ____

b) $\frac{49}{28} =$ ____

c) $\frac{30}{20} =$ ____

d) $\frac{21}{42} =$ ____

e) $\frac{21}{18} =$ ____

f) $\frac{36}{27} =$ ____

Anhang: Math.js-Ausdrücke

In der Rechenaufgabe lassen sich aus bereits bestehenden Variablen neue Variablen berechnen. Am häufigsten kommen hierbei die Grundrechenarten zum Einsatz, die Sie über die Zeichen +, -, * und / direkt eingeben können. Darüber hinaus gibt es noch einige nützliche Funktionen und Konstanten, die von der Skriptsprache math.js bereitgestellt werden. Die folgenden Tabellen geben Ihnen einen Überblick über die am häufigsten verwendeten Ausdrücke.

Hinweis: Das Komma (als Dezimaltrennzeichen) muss in math.js als Punkt eingegeben werden.

Funktionen

Funktion	Beschreibung
abs(x)	Betrag einer Zahl x
ceil(x)	Zahl x wird aufgerundet (1)
floor(x)	Zahl x wird abgerundet (1)
fix(x)	Zahl x wird zur Null hin gerundet (2)
round(x)	Zahl x wird auf die „nächste“ ganze Zahl gerundet („klassisches“ Runden) (3)
round(x, n)	„klassisches“ Runden der Zahl x auf n Dezimalstellen (3)
gcd(a, b)	berechnet den größten gemeinsamen Teiler von a und b (4)
lcm(a, b)	berechnet das kleinste gemeinsame Vielfache von a und b (4)
mod(x, y)	berechnet den Rest, der bei ganzzahliger Division von x und y übrig bleibt
exp(x)	Wert der Funktion e^x
log(x)	natürlicher Logarithmus von x
log(x, a)	Logarithmus von x zur Basis a
pow(x, y) *oder* x^y	x „hoch“ y
sqrt(x)	Quadratwurzel von x
nthRoot(x, n)	n-te Wurzel von x

Funktionen in math.js (Auszug)

Anmerkungen

(1) Die Funktionen ceil und floor runden immer zur nächstgrößeren ganzen Zahl auf (ceil) bzw. zur nächstkleineren ganzen Zahl ab (floor) - und zwar unabhängig von den Nachkommastellen. Die Zahl 2.8 bspw. wird von der Funktion floor genauso auf 2 abgerundet wie die Zahl 2.3.

(2) Die Funktion fix rundet „zur Null hin“, d.h. auf die betragsmäßig nächstkleinere ganze Zahl; positive Zahlen werden also immer abgerundet (wie bei floor) und negative Zahlen immer aufgerundet (wie bei ceil). Zum Beispiel ist fix(3.9) = 3 und fix(-3.9) = -3.

(3) Die Funktion round entspricht der klassischen Auffassung des Rundens, d.h. es wird auf die „nächste“ ganze Zahl gerundet. Für eine positive Zahl heißt das: Wenn die erste Ziffer nach dem Komma zwischen 0 und 4 liegt, wird abgerundet - ansonsten wird aufgerundet. Bei negativen Zahlen ist es dementsprechend umgekehrt. Das zweite Argument n ist optional und kann benutzt werden, wenn man auf n Dezimalstellen runden möchte.

(4) Die Funktionen gcd und lcm können auch mit mehr als zwei Eingabewerten benutzt werden. So ist z.B. gcd(6, 8, 10) = 2

Konstanten

Konstante	Beschreibung	Wert
pi, PI	Kreiszahl Pi	3.141592653589793
e, E	Eulersche Zahl e	2.718281828459045
LN2	natürlicher Logarithmus von 2	0.6931471805599453
LN10	natürlicher Logarithmus von 10	2.302585092994046
LOG2E	Logarithmus von e zur Basis 2	1.4426950408889634
LOG10E	Logarithmus von e zur Basis 10	0.4342944819032518
phi	Goldener Schnitt $\left(\frac{1+\sqrt{5}}{2}\right)$	1.618033988749895
SQRT1_2	Quadratwurzel von $\frac{1}{2}$	0.7071067811865476
SQRT2	Quadratwurzel von 2	1.4142135623730951

Konstanten in math.js (Auszug)