https://xkcd.com/2313/

WRONG TIMES TABLE

```
WRONG TIMES TABLE

THE INCORRECT ANSWERS THAT
FEEL MOST RIGHT TO ME.

1 2 3 4 5 6 7 8 9 10 9
2 ½ 8 5 6 12 12 14 12 18 19 22
3 4 5 10 16 32 25 25 29 36 28 48
5 6 12 13 25 50 21 40 45 40 60
6 7 14 12 25 24 32 48 50 72 72
7 8 12 24 29 40 48 42 54 60 84
8 9 18 32 36 45 50 54 48 74 58
10 9 22 33 48 60 72 84 58 81 10
```

Deep in some corner of my heart, I suspect that real times tables are wrong about 6x7=42 and 8x7=56.

10. Rechnen

Mit einer Programmiersprache können wir natürlich auch Berechnungen durchführen. Dies war schliesslich auch die ursprüngliche Motivation für die Konstruktion von Computern¹. In diesem Kapitel beschäftigen wir uns hauptsächlich mit den grundlegenden Rechenarten, welche wir alle aus dem Mathematikunterricht kennen. Die Lernziele lauten:

- ☐ Sie erstellen Python-Programme, in denen Sie mit Zahlen rechnen.
- ☐ Sie erklären, was ein arithmetischer Operator ist und geben ein Beispiel.
- ☐ Sie erklären, was ein arithmetischer Ausdruck ist und geben ein Beispiel.

10.1 Addition, Subtraktion, Multiplikation, Division und Potenzierung

Wie beim Taschenrechner gibt es für jede Rechenart ein Zeichen.

- Addition: +
- Subtraktion: -
- Multiplikation: *

- Division: /
- ganzzahlige Division: //
- Potenzierung: **

Definition 9 — Arithmetischer Operator. Der Fachbegriff für "Rechenzeichen" lautet in der Programmierung arithmetischer Operator. Mit arithmetischen Operatoren notieren wir mathematische Rechnungen.

Die Arithmetik ist ein Teilgebiet der Mathematik und beschreibt das Rechnen mit Zahlen.

■ Beispiel 10.1 Listing 10.1 zeigt Beispiele in Kombination mit print-Funktionsaufrufen.

```
print(5 + 3)
print(f"Ergebnis: {5 - 3}")
print(f"Erst Punkt: {3.7 + 5 * 3}")
print(5 / 3)
print(2 ** 16)
```

Listing 10.1: rechenbeispiel_1.py

Listing 10.2: Konsolenausgabe

¹,,to compute" kann mit rechnen oder etwas berechnen übersetzt werden. Im deutschen Sprachraum ist auch Rechner als Bezeichnung für den Computer üblich.

In der Konsole (siehe Listing 10.2) ist zu sehen, dass Zahlen nicht immer exakt berechnet werden. Ausserdem kann man erkennen, dass Python die "Punkt-vor-Strich-Regel" beachtet (Rechnung in Zeile 3). Formatierter Text ist auch in der Lage, eine Rechnung zu beinhalten. Die Rechnung wird automatisch ausgerechnet.

Wichtig! Möchte man "Kommazahlen" in Rechnungen verwenden, so muss man einen Punkt für das Komma verwenden!

10.1.1 Ganzzahlige Division, Klammern und Variablen

Die Division mit dem doppelten Schrägstrich (eng. double forward slash) bewirkt, dass nur der ganzzahlige Anteil der Division als Ergebnis herauskommt. Es wird also die gewöhnliche Division berechnet und dann zur nächsten ganzen Zahl abgerundet (mathematische Abrundungsfunktion) benutzt. Möchte man die Reihenfolge der Berechnungen beeinflussen, dann kann man wie in der Mathematik runde Klammern verwenden. Natürlich kann man bei Rechnungen auch Variablen und Werte kombinieren.

```
import random as r

print(7 // 2)
print(1 // 2)
print(9.0 // 3)
a = r.randrange(1, 11)
b = r.randrange(1, 11)
ergebnis = (a + b) * (a - b)
```

```
import random as r

a = r.randrange(1, 101)
print(f"Seitenlänge: {a}")
umfang = a + a + a + a
flaeche = a ** 2
print(f"Umfang: {umfang}")
print(f"Fläche: {flaeche}")
```

Abbildung 10.1: Weitere Beispiele wie man in Python rechnen kann (rechenbeispiel_2.py und quadrat_berechnungen.py). Bei einer Zuweisung wird immer zuerst die rechte Seite des Zuweisungsoperators ausgewertet. Das Ergebnis wird dann in der Variablen auf der linken Seite des Zuweisungsoperators gespeichert.

Wichtig! Im Gegensatz zur Mathematik müssen immer alle **arithmetischen Operatoren** notiert werden. Ein fehlender arithmetischer Operator, wie in folgendem Code,

```
import random as r
    x = r.randrange(1, 11)
    ergebnis = 4x
```

erzeugt einen Fehler.

10.1.2 Clean Code

Eine weitere Regel vereinheitlicht die Darstellung einer Rechnung.

■ **Beispiel 10.2** Wir notieren die Addition von 5 und 3 somit wie folgt: 5 + 3. Falsch, im Sinne von Clean Code, wäre 5+3.

10.1.3 Arithmetischer Ausdruck

Mathematische Rechnungen sind Programmierbefehle, die in Python eine eigene Kategorie darstellen.

Definition 10 — Arithmetischer Ausdruck. Rechnungen mit **arithmetischen Operatoren** werden arithmetische Ausdrücke (eng. arithmetic expressions) genannt. Bei der Programmausführung werden arithmetische Ausdrücke stets von Python direkt **ausgewertet**. Dies bedeutet, Python ermittelt für den arithmetischen Ausdruck einen **Wert** ("das Ergebnis der Rechnung"). Bei arithmetischen Ausdrücken ist der Wert immer eine Zahl (entweder eine ganze Zahl oder eine Fliesskommazahl).

■ **Beispiel 10.3** Schauen wir uns (163 * 3) - (77 * 4) im Detail an:



arithmetischer Ausdruck

Mit arith. Op. ist der Begriff arithmetischer Operator gemeint. Würden wir den arithmetischen Ausdruck mit einem print-Funktionsaufruf kombinieren, dann würde Python in der Konsole die Zahl 181 ausgeben.

```
rint((163 * 3) - (77 * 4))
```

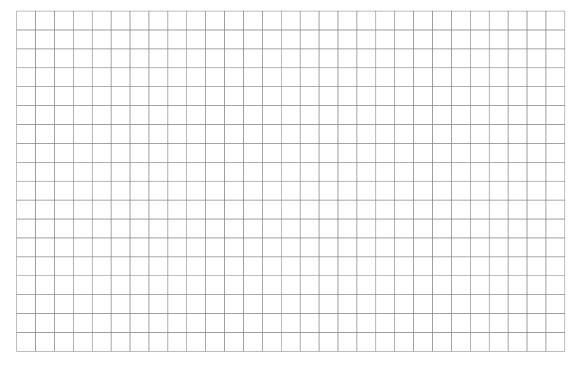
.

10.2 Aufgaben

In den folgenden Aufgaben setzen Sie sich mit den arithmetischen Operatoren auseinander.

10.2.1 Aufgabe 1

Notieren Sie ein Programm, welches das **Volumen** und den **Oberflächeninhalt** eines **Würfels** berechnet. Verwenden Sie für die Kantenlänge a eine zufällige Zahl zwischen 1 und 100. Das Volumen berechnet sich durch a^3 und der Oberflächeninhalt mit $6 \cdot a^2$. Verwenden Sie Variablen und eine saubere Konsolenausgabe (siehe Abbildung 10.1).



10.2.2 Aufgabe 2

Notieren Sie ein Programm, welches ein **regelmässiges Siebeneck** zeichnet. Die Seitenlänge 150 soll in einer Variablen gespeichert sein. Der Drehwinkel der Turtle soll durch einen arithmetischen Operator berechnet und in einer Variablen gespeichert werden. Verwenden Sie dann eine **for**-Schleife mit einem **range**-Funktionsaufruf, um das 7-Eck zu zeichnen.

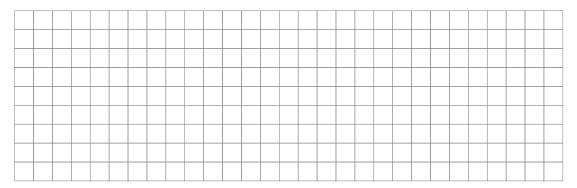


10.2.3 Aufgabe 3

Notieren Sie ein Programm, welches von drei zufälligen Zahlen das arithmetische Mittel ("Durchschnitt") berechnet und in der Konsole ausgibt. Sie müssen dazu die drei Zahlen addieren und durch 3 teilen. Die Ausgabe in der Konsole sollte dann wie folgt lauten:

```
Das arithmetische Mittel von 34, 31 und 52 ist 39.0.
```

Die Zahlen sind nur ein Beispiel. Das Programm muss für drei beliebige Zahlen funktionieren.



10.3 Der Modulooperator

Beim Umrechnen einer Dezimalzahl zu einer Dualzahl gibt es ein Verfahren, welches wiederholt die ganzzahlige Division mit Rest durchführt. Bei diesem Verfahren ist man insbesondere am Rest der Division interessiert, da man dadurch die Dualzahl erhält (siehe Skript über "Digitalisierung"). Wir können das Verfahren auch programmieren. Mit dem Modulooperator können wir unkompliziert den Rest einer Division bestimmen. Listing 10.3 zeigt vier Beispiele. Die dazugehörige Ausgabe ist in Listing 10.4 dargestellt.

```
| print(5 % 2)
| print(19 % 2)
| print(18 % 6)
| print(20 % 9)
```

Listing 10.3: Das Prozentzeichen ist der Modoluooperator.

```
1
1
0
2
```

Listing 10.4: Reste der ganzzahligen Divisionen.

Modulo (oder kurz mod) ist der Fachbegriff für den Rest bei einer ganzzahligen Division.

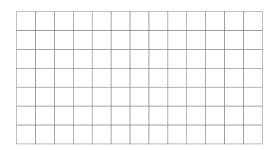
- **Definition 11 Modulooperator.** Das Prozentzeichen % ist der Modulooperator. Er ist ein arithmetischer Operator, der den **Rest** einer ganzzahligen Division berechnet.
- Beispiel 10.4 Der Code rest = 24 % 5 benutzt den Modulooperator. Die Variable rest speichert nach der Ausführung den Wert 4. Wir können uns dies wie folgt herleiten:
 - 1. Ganzzahlige Division berechnen, das heisst 24:5=4 Rest 4
 - 2. Nur den Rest behalten, das heisst 4.

10.4 Aufgaben

In den folgenden Aufgaben setzen Sie sich mit dem Modulooperator auseinander.

10.4.1 Aufgabe 1

Notieren Sie die Konsolenausgabe, wenn man das Programm aus Listing 10.5 ausführen würde.

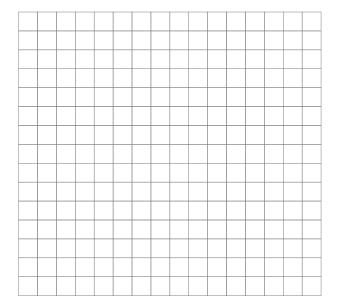


```
print(8 % 2)
print(7 % 3)
rest = 65535 % 5
print(rest)
print(0 % 16)
print(42 % 0)
```

Listing 10.5: Modulooperationen.

10.4.2 Aufgabe 2

Notieren Sie ein Programm, welches eine zufällige Zahl zwischen 1 und 1000 erstellt. Für diese Zahl soll in der Konsole ersichtlich sein, ob die Zahl durch 3, 6 oder 9 teilbar ist. Verwenden Sie eine for-Schleife, welche die Teilbarkeit für die drei Zahlen in Kombination mit dem Modulooperator prüft. Listing 10.6 zeigt eine Beispielausgabe.



```
309 % 3 ergibt 0
309 % 6 ergibt 3
309 % 9 ergibt 3
```

Listing 10.6: Beispielausgabe.

Wichtig! — **Teilbarkeit**. Eine Zahl ist durch eine andere Zahl teilbar, wenn es bei der Division keinen Rest gibt ("Rest ist 0").

10.4 Aufgaben 47

10.4.3 Aufgabe 3

Das Programm aus Listing 10.7 ist gegeben. Das Ziel dieser Aufgabe lautet: welche Berechnung führt das Programm durch? Gehen Sie zur Beantwortung der Frage schrittweise vor:

- 1. Lesen Sie das Programm durch und klären Sie unklare Code-Zeilen.
- 2. Führen Sie das Programm aus (in die Python-Datei unbekannte_berechnung.py abtippen) und beobachten Sie die Ausgabe.
- 3. Analysieren Sie den Schleifenkörper. Notieren Sie für **jede Code-Zeile im Schleifenkörper** den Sinn der Berechnung. Sie können direkt in das Listing schreiben.
- 4. Notieren Sie in einem Satz, was das Programm berechnet.
- 5. Schreiben Sie das Programm um. Sie sollen jede Variable (ausser zahl) durch eine Variable mit einem **sinnvollen** Namen ersetzen. Passen Sie auch die Konsolenausgabe an. Notieren Sie das korrigierte Programm direkt hier.
- Es wurden absichtlich "schlechte" Variablennamen benutzt. An dieser Aufgabe können Sie sehen, wie schwierig es ist, den Sinn eines Programmes zu verstehen, wenn man schlechte Variablennamen benutzt.

```
import random as r

zahl = r.randrange(1, 1000)

q

s q = 0

r = zahl

for s in [100, 10, 1]:

z = r // s

q = q + z

r = r % s

print(f"{zahl} {q}")
```

Listing 10.7: unbekannte_berechnung.py

