Aufgabe 1.1.1

Schreibe ein Programm, das die Summe von 17 und 4 ausrechnet und ausgibt.

Aufgabe 1.1.2

Schreibe ein Programm, das das Produkt von 1234 und 4321 ausrechnet und ausgibt.

Aufgabe 1.1.3

Eine Klasse mit 25 Schülern möchte eine Ausstellung besuchen. Der Eintritt kostet 4,80 €. Sie haben 130 € in der Klassenkasse. Schreibe ein Programm, das ausrechnet, wieviel in der Klassenkasse hinterher noch übrig ist.

Aufgabe 1.2.1

Eine Milchkuh liefert in einem Jahr etwa 4500 Liter Milch. Ein Bauer hat 19 Milchkühe. Wieviel Milch liefert der Bauer in der Molkerei ab?
Schreibe ein Programm, das diese Frage beantwortet, und verwende dabei Variablen Milch und Kuehe

Aufgabe 1.2.3

Mit dem Zeichen % kann man in Python den Rest ermitteln, der bei Division einer Zahl durch eine andere entsteht. Zum Beispiel ergibt 17%4 den Wert 1, weil 17 geteilt durch 4 das Ergebnis 4 mit dem Rest 1 liefert.
Schreibe ein Programm, das den Rest ermittelt, den A*B bei Division durch C liefert, wobei A den Wert 1234, B den Wert 4321 und C den Wert 111 hat. Verwende dabei Variablen A, B, C und Rest.

Aufgabe 1.2.2

Eine Autofahrt von Berlin nach Hamburg dauert 2 Stunden und 48 Minuten. Wie viele Sekunden sind das?
Schreibe ein Programm, das diese Frage beantwortet, und schreibe dazu die Werte aus der Aufgabe in die Variablen Stunden und Minuten und das Ergebnis in die Variable Sekunden.
Hinweis: Eine Stunde hat 60 Minuten, eine Minute 60 Sekunden. Du kannst das Ergebnis in mehreren Schritten ausrechnen: Lege eine weitere Variable an, die die Gesamtminutenzahl ermittelt, und rechne diese dann in Sekunden um.

Aufgabe 1.3.1

Schreibe ein Programm, das den Text “Hallo, Welt!” ausgibt

Aufgabe 1.3.2

Schreibe ein Programm, das die Ausgabe von Aufgabe 1.2.1 in der Form
Der Bauer liefert XXX Liter Milch in der Molkerei ab.
ausgibt, wobei für XXX die richtige Zahl eingetragen wird. Verwende dazu die Operation format().

```
Kuehe = 19
Milch = 4500
Liter = Kuehe * Milch
print('Der Bauer liefert {0} Liter Milch in der Molkerei ab'.format(Liter))
```

Schreibe ein Programm, das den Text
-*-
ausgibt. (zwanzigmal minus-stern, und dann zusätzliches minus). Verwende dazu die Variablen

```
minus = "-"
minus_stern = "-*"
```

Aufgabe 1.4.1

Schreibe ein Programm, das mit der Turtle-Grafik ein Rechteck mit der Kantenlänge 100 zeichnet.

Aufgabe 1.4.2

Schreibe ein Programm, dass mit der TurtleGrafik zwei nebeneinanderliegende Rechtecke (siehe Bild) mit der Kantenlänge 100 ausgibt. Ob die gemeinsame Kante zweimal gezeichnet wird oder nicht, ist Euch überlassen.

Aufgabe 1.4.3

Schreibe ein Programm, das das Haus vom Nikolaus zeichnet.

Hinweise:

- Anstelle der Funktionen forward, right und left kann man auch die Funktion goto verwenden.
- Als Koordinaten kann man die Punkte aus der folgenden Liste verwenden:
  - 0, 0 (Start)
  - 100, 0
  - 100, 100
  - 0, 100
  - 50, 150
  - 100, 100
  - 0, 0
  - 0, 100
  - 100, 0

Aufgabe 2.1.1

Schreibe ein Programm, das die größere der Zahlen A und B ausgibt. Die Festlegung, welchen Wert A und B haben, wird vom Computer getroffen - Dein Programm soll die jeweils richtige Ausgabe erzeugen. Beginne dazu Dein Programm mit

```
from daten import A,B
```

Wenn beispielsweise A den Wert 3 und B den Wert 5 hat, soll Dein Programm die Ausgabe

`5 ist die größere Zahl.`

Falls die beiden Zahlen gleich sind, soll die Ausgabe

`Beide Zahlen sind gleich.`

erscheinen.

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
A=3
B=5
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 2.1.2

Schreibe ein Programm, das für eine gegebene Zahl A den Absolutwert ausgibt. Der Absolutwert erhältst Du, wenn Du das Vorzeichen der Zahl weglässt: für -5 ist der Absolutwert also 5; für 5 (also +5) ist der Absolutwert ebenfalls 5. Die vom Computer vorgegebene Zahl erhältst Du, indem Du das Programm mit

```
from daten import A
```

beginnst.

Du musst keinen ganzen Satz ausgeben; es reicht, wenn Dein Programm bei -5 einfach

```
5
```

ausgibt.

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
A=-5
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Hinweis: Wenn Du die Fehlermeldung IndentationError bekommst, bedeutet das, dass die Zeilen nicht richtig eingerückt sind.Hinweis: Um für eine Zahl A das Vorzeichen umzukehren, kann man -A schreiben, oder mit -1 multiplizieren.

Aufgabe 2.1.3

Schreibe ein Programm, das für eine Variable Hunde ausgibt, wie viele es sind, und zwar in der Form

- Wir haben keinen Hund.
- Wir haben einen Hund.
- Wir haben zwei Hunde.
- Wir haben viele Hunde. Die Variable Hunde erhältst Du durch

```
from daten import Hunde
```

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
Hunde = 2
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Hinweis: Wenn Du die Fehlermeldung IndentationError bekommst, bedeutet das, dass die Zeilen nicht richtig eingerückt sind.

Aufgabe 2.2.1

Schreibe ein Programm, das die Quadratzahlen von 0 * 0 bis 20 * 20 ausgibt.

Aufgabe 2.2.2

Schreibe ein Programm, das untereinander alle geraden Zahlen von 0 bis 25 ausgibt (also jede Zahl in eine neue Zeile).

Aufgabe 2.2.3

Schreibe ein Programm, das gedrehte gleichseite Dreiecke zeichnet (siehe Abbildung). Die Zahl der Dreiecke, ihre Größe, und der Winkel, um die sie gedreht sind, kannst Du Dir selbst aussuchen.

Hinweis: Ein gleichseitiges Dreieck entsteht, wenn der Winkel im Dreieck 60° beträgt und alle Seiten gleich lang sind. Die Schildkröte muss sich an jeder Ecke um 120° nach links oder rechts drehen, damit der verbleibende Winkel 60° ergibt.

Aufgabe 2.3.1

Schreibe ein Programm, das die Summe der Zahlen aus einer Liste bildet. Die Liste erhältst Du durch

```
from daten import Zahlen
```

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
Zahlen = [10, 7, 9, 16, 2, 15]
```

Aufgabe 2.3.2

Schreibe ein Programm, das den Durchschnitt der Zahlen aus einer Liste bildet. Den Durchschnitt erhältst Du, wenn Du die Summe der Zahlen durch ihre Anzahl (also die Länge der Liste) teilst. Die Liste der Zahlen bekommst Du durch

```
from daten import Zahlen
```

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
Zahlen = [8, 9, 9, 16, 16, 15, 18, 5]
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 2.3.3

Schreibe ein Programm, das die kleinste und die größte Zahl in einer Liste namens Zahlen ausgibt, zum Beispiel in der Form

```
Die kleinste Zahl ist 10 und die größte Zahl ist 97.
```

Du musst aber nicht unbedingt einen ganzen Satz ausgeben.</p>

Die Liste erhältst Du mit

```
from daten import Zahlen
```

Hinweis: Neben einer Schleife musst Du hier auch if-Anweisungen einsetzen.

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
Zahlen = [41, 25, 40, 97, 76, 40, 43, 10]
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 2.4.1

Wie viele der ersten natürlichen Zahlen (1,2,3,4,...,N) muss man addieren, damit die Summe nicht mehr kleiner als (ein vorgegebenes) X ist. Schreibe ein Programm, das diese Frage beantwortet und N ausgibt. Das X wird Dir vorgegeben durch

```
from daten import X
```

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
X=1000
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 2.4.2

Schreibe ein Programm, das das erste Wort in einer Liste Namen findet, das mit D beginnt. Die Liste bekommst Du mit

```
from daten import Namen
```

Hinweis: Es gibt zwei Möglichkeiten:

1. Du schreibst eine while-Schleife, die der Reihe nach die Namen anschaut, solange sie nicht mit D anfangen. Dazu verwendest Du einen Zähler, der bei 0 anfängt, und der Reihe nach alle Namen abfragt.

2. Du schreibst eine for-Schleife, die alle Namen durchgeht und abbricht (mit `break`), wenn ein Name mit D gefunden wurde

   Hinweis: Den ersten Buchstaben eines Wortes W bekommst Du mit `W[0]`. Probier's aus!

   Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

   ```
   Namen =  ['Elias', 'Julia', 'Daniel', 'Dominic']
   ```

   Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 2.4.3

Schreibe ein Programm, das alle Teiler einer Zahl N ausgibt, jeden in eine eigene Zeile. Zum Beispiel soll für N = 24 die Ausgabe

```
2
3
4
6
8
12
```

entstehen (die Zahlen 1 und 24 werden also jetzt nicht als Teiler verstanden - es sind nur die "echten" Teiler gefragt). Den Wert N erhältst mit

```
from daten import N
```

Hinweis: Um zu bestimmen, ob eine Zahl Teiler ist, kann man fragen, ob der Rest bei der Division 0 ist. Die Berechnung des Rests hast Du ja schon in Aufgabe 1.2.3 durchgeführt.

Hinweis: Um die Teiler alle zu bekommen, kann man einfach alle Zahlen von 2 bis N-1 ausprobieren. Ein Mensch kann leicht viele von diesen Zahlen ausschließen; für ein Programm ist es aber einfacher, sie alle durchzuprobieren.

Hinweis: Zum Ausprobieren kannst Du zunächst mit festen Werten arbeiten, zum Beispiel

```
N = 24
```

Zum Abgeben musst Du dann die import-Zeile oben benutzen.

Aufgabe 3.1.1

Schreibe eine Funktion Weihnachtsbaum, die 2 Paramater x und y erwartet und den Weihnachtsbaum an diesen Koordinaten zeichnet. Die genaue Form des Weihnachtsbaums ist Dir überlassen.

Rufe dann die Funktion mit den Koordinaten x=50 und y=100 auf.

Aufgabe 3.1.2

Schreibe eine Funktion Weihnachtsbaum, die 2 Paramater x und y erwartet und den Weihnachtsbaum an diesen Koordinaten zeichnet.

Schreibe eine weitere Funktion Weihnachtswald, die 9 Weihnachtsbäume an verschiedene Positionen malt. Die Positionen kannst Du Dir selber aussuchen (und zum Beispiel die Bäume in einem 3x3-Feld anordnen)

Rufe dann die Funktion Weihnachtswald auf

Aufgabe 3.1.3

Schreibe eine Funktion Summe, die die Summe der ersten N Zahlen berechnet und mit print ausgibt. Dabei soll N der Parameter der Funktion sein. Zum Beispiel soll Summe(4) die Summe 1+2+3+4 = 10 ausrechnen und die Zahl 10 ausgeben.

Zum Testen kannst Du überprüfen, ob Summe(100) den Wert 5050 ergibt.

Hinweis: Vermeide die Verwendung einer Summenformel. Du kannst die Summe auch dadurch bestimmen, indem einfach die Zahlen alle zusammenaddiert werden

Aufgabe 3.2.1

Schreibe eine Funktion Wörter_mit_a, die aus einer Liste mit Wörtern die ermittelt, die mit a anfangen.

Aufgabe 3.2.2

Schreibe eine Funktion Anzahl_Vokale, die für ein Wort bestimmt, wie viele Vokale es hat und diese Anzahl zurückgibt. Umlaute zählen hier auch als Vokale; Großbuchstaben müssen nicht (aber dürfen) berücksichtigt werden. Z.B. soll sie für das Wort "bestimmt" den Wert 2 als Ergebnis liefern

Schreibe dann eine Funktion Viele_Vokale, die zwei Parameter hat. Der erste Parameter ist eine Liste von Wörtern, und der zweite Parameter eine Anzahl. Das Ergebnis der Funktion ist die Liste der Wörte, die mindestens die angegebene Anzahl von Vokalen hat.

Z.B. soll ein Aufruf von :

Viele_Vokale(['schreibe', 'dann', 'eine', 'funktion', 'viele_vokale,', 'die', 'zwei', 'parameter', 'hat'], 4) die Liste ['viele_vokale,', 'parameter'] als Ergebnis liefern

Aufgabe 3.2.3

Schreibe eine Funktion ist_Palindrom, die ein Wort als Parameter erwartet und wahr zurückgibt, wenn das Wort ein Palindrom ist, ansonsten falsch. Ein Palindrom ist ein Wort, das vorwärts und rückwärts gelesen gleich lautet, z.B. Rentner, Reittier, oder Lagerregal.

Zur Erinnerung: Die Werte wahr und falsch werden in Python durch `True` und `False` ausgedrückt.

Du kannst für die Aufgabe auf Großschreibung verzichten (also annehmen, dass alle Wörter kleingeschrieben werden).

Zum Testen des Programms kannst Du alle Palindrome in einem Satz ausgeben lassen, z.B.

```
Worte = "anna und otto haben ein pferd als reittier".split()
for Wort in Worte:
    if ist_Palindrom(Wort):
        print(Wort)
```

Aufgabe 3.3.1

Schreibe ein Programm, das eine Variable Hauptstädte mit einem Dictionary belegt, in dem jedem Bundesland seine Hauptstadt zugeordnet ist

Aufgabe 3.3.2

Schreibe ein Funktion Stadtstaaten, die alle Bundesländer als Liste zurückgibt, bei denen der Name der Hauptstadt gleich dem Namen des Bundeslands ist. Verwende dazu das Dictionary aus Aufgabe 3.3.1.

Hinweis: die Schlüssel eines Dictionaries erhältst Du, wenn Du die Methode .keys() aufrufst (etwa `Hauptstädte.keys()`; Du kannst Schlüssel dann direkt mit einer for-Schleife verarbeiten

Aufgabe 3.3.3

Schreibe ein Funktion spiegeln, die ein Dictionary als Parameter erwartet und ein Dictionary liefert, bei dem die Schlüssel und Werte vertauscht sind.

Du kannst Deine Funktion mit dem Dictionary Hauptstädte ausprobieren und solltest ein Dictionary erhalten, mit dem für jede Hauptstadt ermittelt werden kann, zu welchem Bundesland sie gehört

Aufgabe 3.4.1

Schreibe eine Funktion zähle(wurf), die in einem Dictionary zähler einen Wurf eines Würfels (also eine Zahl zwischen 1 und 6) mitzählt.

Schreibe dann eine Funktion würfeln(anzahl), die eine anzahl würfe macht und sie zählt

Aufgabe 3.4.2

Schreibe ein Programm, das die Schildköte wild herumfahren lässt. Lasse sie eine zufällige Anzahl zwischen 10 und 40 Linien zeichnen, die eine zufällige Länge zwischen 10 und 100 haben, und sie nach jeder Linie um zufälligen Winkel zwischen 10 und 170 Grad drehen.

Wenn Du willst, kannst sich die Schildkröte auch zufällig nach links oder rechts drehen

Aufgabe 3.4.3

Schreibe ein Programm, das den Wetterbericht der nächsten Woche liefert. Das Programm soll lediglich die Temperaturen der nächsten Woche vorhersagen, in der Form

```
Am Montag werden es XXX °C.
Am Dienstag werden es XXX °C.
Am Mittwoch werden es XXX °C.
Am Donnerstag werden es XXX °C.
Am Freitag werden es XXX °C.
Am Sonnabend werden es XXX °C.
Am Sonntag werden es XXX °C.
```

Da wir die Temperaturen der nächsten Woche nicht kennen, lassen wir sie vom Zufallszahlengenerator bestimmen. Am Montag soll der Wert zwischen 10 und 20 °C liegen, an den nächsten Tagen immer nur 2 Grad höher oder tiefer als am vorigen Tag.

Verwende eine Schleife, um alle Ausgaben zu erzeugen. Um die Wochentage richtig auszugeben, kannst Du Dir eine Liste aller Wochentage anlegen.

Aufgabe 4.1.1

Schreibe ein Programm, das Deinen Namen und Dein Lieblingstier abfragt, und dann beides in einem Satz ausgibt.

Aufgabe 4.1.2

Schreibe ein Programm, das der Reihe nach

1. eine Zahl
2. eine Rechenart (+, -, *, /)
3. eine zweite Zahl abfragt, und dann die beiden Zahlen verknüpft.

Ein möglichher Dialog könnte z.B. so aussehen

```
Erste Zahl: 17
Rechenart: +
Zweite Zahl: 4
Das Ergebnis von 17+4 ist 21
```

Aufgabe 4.1.3

Schreibe ein Programm, das beim Lernen von Hauptstädten hilft.

Das Programm soll zufällig ein Bundesland nennen, dann soll der Nutzer die Hauptstadt des Landes eingeben. Das Programm soll dann sagen, ob die Eingabe richtig oder falsch war. Danach soll das Programm nach der nächsten Hauptstadt fragen, und so weiter, bis der Nutzer irgendwann "Ende" eingibt.

Ein möglicher Dialog mit dem Programm sieht so aus

```
Wie lautet die Hauptstadt von Brandenburg? Cottbus
Das ist leider falsch.
Wie lautet die Hauptstadt von Bayern? München
Das ist richtig.
Wie lautet die Hauptstadt von Sachsen? Ende
Auf Wiedersehen.
```

Hinweis: Wenn D ein Dictionary ist, bei dem die Bundesländer die Schlüssel sind, bekommst Du mit list(D.keys()) eine Liste der Bundesländer. Du kannst Dir dann einen zufälligen Index nehmen und so ein Bundesland auswürfeln.

Aufgabe 4.2.1

Schreibe ein Programm, mit dem Computer sich eine Zahl zwischen 1 und 20 ausdenkt, die Du rätst. Verwende als mögliche Antworten Deines Programms

- Zu klein
- Zu groß
- Herzlichen Glückwunsch

Aufgabe 4.2.2

Schreibe ein Programm, mit dem Computer sich eine Zahl zwischen 1 und 100 ausdenkt, die Du rätst. Verwende als mögliche Antworten Deines Programms

- Viel zu klein
- Viel zu groß
- Zu klein
- Zu groß
- Herzlichen Glückwunsch

Die ersten beiden Antworten sollen gegeben werden, wenn die Zahl mehr als

20 zu klein oder zu groß ist.

Aufgabe 4.2.3

Schreibe ein Programm, das eine Zahl zwischen 1 und 100 rät, die der Nutzer sich ausgedacht hat. Der Nutzer soll dann mit einem Buchstaben antworten, ob die Zahl, die das Programm geraten hat, zu klein, zu groß, oder richtig ist, mit den Befehlen

- r (richtig)
- k (die geratene Zahl ist zu klein)
- g (die geratene Zahl ist zu groß)

Ein möglicher Dialog könnte sein

```
50? k
75? g
62? g
56? k
59? r
```

Versuche das Programm so zu schreiben, dass es mit möglichst wenig Versuchen auskommt. Es gibt eine Lösung, die immer mit 7 oder weniger Versuchen auskommt, aber eine Lösung, die immer mit weniger als 20 Versuchen auskommt, ist ganz gut. Eine Lösung, die bis zu 100 Versuche braucht, wird ebenfalls akzeptiert.

Hinweis: Achte darauf, dass die eingegebene Zahl in der gleichen Zeile wie die Frage steht. Das kannst Du erreichen, indem Du den Text der Frage als Parameter an input() übergibst.

Aufgabe 4.3.1

Schreibe ein Programm, das mit Turtle-Grafik ein blaues Schiff über den Bildschirm fahren lässt.

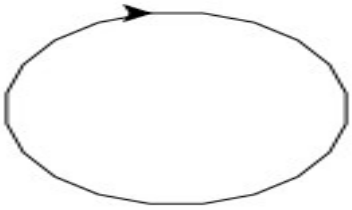
Aufgabe 4.3.2

Schreibe ein Programm, dass zwei Schiffe aufeinander zufahren lässt. Setze dazu das eine Schiff auf die Koordinaten (-100, 0), und das andere Schiff auf die Koordinaten (100,0). Lasse sie dann aufeinander zufahren, indem sie sich in jedem Schritt immer um einen x-Unterschied von 1 aufeinander zubewegen.

Aufgabe 4.3.3

Schreibe ein Programm, das ein Schiff im Kreis fahren lässt und dabei eine Kreislinie zeichnet.

Hinweis: eine wirkliche Kreislinie zu verfolgen, ist nicht möglich; eine ungefähre Kreislinie reicht. Zum Beispiel könnte das Schiff auf einem gleichseitigen 20-Eck fahren (die Abbildung zeigt so ein 20-Eck). In einem 20-Eck ist jeder Winkel 360°/20 = 18°.



Aufgabe 4.4.1

Schreibe ein Programm, bei dem die Schildkröte immer an die Position fährt, die mit der Maus angeklickt wurde.

Aufgabe 4.4.2

Schreibe ein Programm, das bei jedem Mausklick die Schildkröte an den "gespiegelten" Punkt setzt (bezüglich des Koordinatenursprungs). Wenn also an die Koordinaten x=-70, y=30 geklickt wird, soll die Schildkröte zu den Koordinaten x=70, y=-30 gehen.

Aufgabe 4.4.3

Schreibe ein Programm, bei dem die Schildkröte vor der Maus davonläuft.

Bei jedem Mausklick soll das Programm prüfen, ob der Click in der Nähe der Schildkröte war (+-10 Einheiten für x und y). Falls ja, soll die Schildkröte an eine zufällige andere Position laufen.

Hinweis: die aktuelle Position der Schildkröte erhältst Du mit

```
x, y = pos()
```