# Variablen 24.10.2024

Variablen sind «Speicherorte für Daten». Jede Variable hat einen Namen und einen Wert.

- Der Name einer Variablen ist wie in der Mathematik oft ein einzelner Buchstabe (etwa `x`, `y`, `z`, `a`), aber auch Wörter sind als Variablennamen erlaubt (etwa `alter`, `flaeche hoehe_des_dreiecks`, `alterInJahren`). Variablennamen sollten kurz, aber aussagekräftig sein.
- Der Wert einer Variablen kann eine Zahl oder eine Zeichenkette sein. Auch komplexere Objekte wie Punkte, Vektoren, Listen von Zahlen oder Bilder sind als Werte möglich.

Eine Variable nimmt oft während des Programmablaufs verschiedene Werte an.

Beispielprogramm: Das Gleichheitszeichen in Python ist ein **Zuweisungszeichen**, das einer Variablen einen Wert zuweist.

```
a = 2

b = -3*a

c = -20

x = (-b + (b**2 - 4*a*c)**0.5)/(2*a)

a = 77

b = b + 11

quotient = a / b

ganzzahlquotient = a // b

rest = a % b
```

## Wiederholungen

Oft wird ein gewisser Programmcode so lange wiederholt, wie eine gewisse Bedingung erfüllt ist. Dies geschieht in Python durch sogenannte «while-Schleifen».

Beispielprogramm: Die Einrückungen haben entscheidende Bedeutung. Nur die um 4 Leerschläge eingerückten Zeilen werden wiederholt.

```
i = 1
s = 0
while i <= 4:
    print(i*2)
    s = s + i**2
    i=i+1
print(s)
```

Im obigen Programm kommt das Vergleichszeichen `<=` vor für «kleiner-gleich». Allgemein werden die Vergleichszeichen wie folgt notiert:

- Gleichheit wird mit **doppeltem Gleichheitszeichen** `==` getestet.
- «kleiner», «kleiner-gleich», «grösser-gleich» und «grösser» werden als `<`, `<=`, `>=` und `>` notiert.
- «ungleich» ($\neq$) wird als `!=` notiert.

### ⚒ Aufgabe A1

(a) Welche Ausgabe produziert das folgende Programm?

```
a = 512
x = a % 10
b = a // 10
y = b % 10
z = b // 10
print(x)
print(y)
print(z)
print(100 * y + 10 * x + z)
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet, wenn man in der ersten Zeile die Zahl 512 durch eine beliebige dreistellige Zahl ersetzt!

### ⚒ Aufgabe A2

Welche Ausgabe produziert das folgende Programm?

```
n = -1
s = 0
while n <= 10:
    print(n, s)
    n = n + 2
    s = s + n
```

### ⚒ Aufgabe A3

(a) Spielen Sie Computer! Welche Ausgabe produziert das unten gegebene Programm für die Eingaben

(i) $a = 31$ und $b = 10$; (ii) $a = 23$ und $b = 5$.

```
a = int(input("Gib eine natürliche Zahl a ein: "))
b = int(input("Gib eine natürliche Zahl b ein: "))
q = 0
while a>=b:
   a = a-b
   q = q+1
print(a)
print(q)
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet!

### ⚒ Aufgabe A4

(a) Spielen Sie Computer! Welche Ausgabe produziert das unten gegebene Programm für die Eingaben

(i) $n = 24$; (ii) $n = 64$.

```
n = int(input("Gib eine natürliche Zahl a ein: "))
z = 1
while z*z<n:
    z = z+1
print(z*z)
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet!

## Verzweigungen

**(ausführlicher das nächste Mal)**

Oft werden manche Teile eines Programms nur unter bestimmten Bedingungen ausgeführt. Man beachte die Einrückungen.

```
if b==42:
    print("Die Antwort!")
else:
    print("Nicht korrekt")
```

### ⚒ Aufgabe A5

(a) Spielen Sie Computer! Welche Ausgabe produziert das unten gegebene Programm für die Eingaben

(i) $z = 16$; (ii) $z = 15$; (iii) $z = 42$.

```
z = int(input("Gib eine natürliche Zahl ein: "))
while z>0:
    if z % 2 == 0:
        print(0)
        z = z/2
    else:
        print(1)
        z = (z-1)/2
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet!

## Aufgabe A6

(a) Spielen Sie Computer! Welche Ausgabe produziert das unten gegebene Programm für die Eingabe $x = 2004$, $y = 900$?

```
x = int(input("Gib eine natürliche Zahl x ein: "))
y = int(input("Gib eine natürliche Zahl y ein: "))

while y > 0:
    print("x =", x, "und y =", y)
    r = x % y
    x = y
    y = r
print("Das Ergebnis ist:", x)
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet!

## Aufgabe A7

(a) Spielen Sie Computer! Welche Ausgabe produziert das unten angegebene Programm für die Eingabe $x = 2$, $b = 8$.

```
a = int(input("Gib eine natürliche Zahl a ein (mindestens 2): "))
b = int(input("Gib eine grössere natürliche Zahl b ein: "))
x = a
while x < b:
    t = 2
    gefunden = False
    while t < x:
        if x % t == 0:
            gefunden = True
        t = t + 1
    if gefunden == False:
        print(x)
    x = x + 1
```

(b) Finden Sie heraus, was dieses Programm allgemein berechnet!

# Verzweigungen = if-Statements 31.10.2024

Motivation an Tafel: Entscheidungsbaum für Lösungen von $ax^2 + bx + c = 0$ anzeichnen.

Oft werden manche Teile eines Programms nur unter bestimmten Bedingungen ausgeführt. Hier einige Beispiele. Man beachte die Einrückungen.

```
if b==42:
    print("Die Antwort!")
```

```
if b==42:
    print("Die Antwort!")
else:
    print("Nicht korrekt")
```

Man kann if-Statements auch verschachteln.

```
if b % 4 == 0:
    print("Die Zahl b ist durch 4 teilbar.")
else:
    if b % 2 == 0:
        print("Die Zahl b ist durch 2 teilbar, aber nicht durch 4.")
    else:
        print("Die Zahl b ist nicht durch 2 teilbar.")
```

Das folgenden Programme macht genau dasselbe wie das gerade betrachtete Programm: Das *Schlüsselwort* `elif` ist eine Abkürzung für «else if».

```
if b % 4 == 0:
    print("Die Zahl b ist durch 4 teilbar.")
elif b % 2 == 0:
    print("Die Zahl b ist durch 2 teilbar, aber nicht durch 4.")
else:
    print("Die Zahl b ist nicht durch 2 teilbar.")
```

### ⚒ Aufgabe A8

```
minuten = int(input("Wie viele Minunten Tageslicht hattest du heute bereits? "))
#
# Hier ist Code
# zu ergänzen.
#
```

Erweitern Sie das obige Programm wie folgt:

Falls der Benutzer 120 oder mehr antwortet, soll er gelobt werden. Sonst wird ihm gesagt, wie viele Minuten er noch draussen verbringen sollte, damit er mindestens 2 Stunden draussen ist.

Beispiel: Der Benutzer gibt ein, dass er 35 Minuten Tageslich hatte. Dann soll der Computer Folgendes ausgeben:

```
Bitte gehe noch 85 Minuten nach draussen.
```

Testen Sie Ihr Programm mit Eingaben wie 30, 170, $-10$, 2400.

Bonus: Wenn der Benutzer etwas Unsinniges eingibt, etwa eine negative Zahl oder eine Zahl, die grösser als $24 \cdot 60$ ist: Melden Sie dies.

Hintergrundinformation: Zwei Stunden Tageslicht pro Tag minimieren laut wissenschaftlicher Studien das Risiko, kurzsichtig zu werden, siehe Deutschlandfunk Kultur: https://www.deutschlandfunkkultur.de/sehstoerung-des-auges-immer-mehr-kinder-werden-kurzsichtig-100.html

**⚒ Aufgabe A9** Ergänzen Sie das folgende Programm so, dass es alle Lösungen der folgenden linearen Gleichung ausgibt.

$$ax + b = 0$$

Das Programm soll auch im Fall $a = 0$ funktionieren und dann ausgeben, ob jede reelle Zahl eine Lösung ist oder ob die Gleichung keine Lösung hat.

```
print("Ich kann jede lineare Gleichung ax+b=0 lösen.")
a = int(input("Gib eine Zahl a ein: "))
b = int(input("Gib eine Zahl b ein: "))
#
# Hier ist Code
# zu ergänzen.
#
```

**Aufgabe A10** Ergänzen Sie das folgende Programm so, dass es alle Lösungen der quadratischen Gleichung $ax^2 + bx + c = 0$ ausgibt.

(a) Nehmen Sie zunächst an, dass $a \neq 0$ gilt. Das Programm soll dann einerseits ausgeben, ob die Gleichung keine, genau eine oder genau zwei Lösungen hat (Diskriminante $D = b^2 - 4ac$ verwenden); andererseits soll es alle Lösungen ausgeben.

(b) Schreiben Sie das Programm nun so um, dass es auch im Fall $a = 0$ funktioniert. In diesem Fall ist also die Lösung der linearen Gleichung $0x^2 + bx + c = bx + c = 0$ zu untersuchen (vgl. vorige Aufgabe).

```
print("Ich kann jede quadratische Gleichung ax^2+bx+c=0 lösen.")
a = int(input("Gib die Zahl a ein: "))
b = int(input("Gib die Zahl b ein: "))
c = int(input("Gib die Zahl c ein: "))
D = b**2 - 4*a*c                          # Diskriminante D
#
# Hier ist Code
# zu ergänzen.
#
```

**Aufgabe A11** Die Collatz-Folge ist wie folgt definiert:

(Schritt 1) Starte mit einer beliebigen natürlichen Zahl $n > 0$.

(Schritt 2) Wenn $n$ gerade ist: Ersetze $n$ durch $\frac{n}{2}$.
Sonst: Ersetze $n$ durch $3n + 1$.

(Schritt 3) Wiederhole den vorherigen Schritt, solange $n > 1$ gilt.

Beispiel: Die mit 13 startende Collatz-Folge sieht so aus: $13, 40, 20, 10, 5, 16, 8, 4, 2, 1$

Passen Sie das folgende Programm so an, dass es ausgehend von einer Startzahl die Collatz-Folge ausgibt.

```
n = int(input("Gib eine natürliche Zahl > 0 ein: "))
while n>1:
    print(n)
    #
    # Hier ist Code
    # zu ergänzen.
    #
print(n)
```

**Aufgabe A12** Schreiben Sie jeweils ein Programm in Python-Syntax, das

a) eine Zahl $n$ einliest und dann alle Teiler von $n$ ausgibt.

b) eine Zahl $n$ einliest und dann die Summe aller Teiler (ausser $n$ selbst) ausgibt.

c) eine Zahl $n$ einliest und dann alle Zahlen $x$ zwischen 1 und $n$ ausgibt, deren Teilersumme (wie in Aufgabe b) definiert) gleich der Zahl $x$ ist.
*Ein Beispiel für eine solche Zahl ist 6. (Teilersumme $1 + 2 + 3 = 6$). Eine Zahl mit dieser Eigenschaft heisst **perfekt**.*

**Aufgabe A13** Die Wurzel $\sqrt{x}$ einer positiven Zahl $x$ kann man näherungsweise berechnen, indem man zwei Variablen `links` und `rechts` so initialisiert, dass `links` $\leq \sqrt{x} \leq$ `rechts` gilt. Der gesuchte Wert $\sqrt{x}$ ist also «zwischen `links` und `rechts` eingesperrt». Zum Beispiel kann man `links` $= 0$ und `rechts` $= x$ setzen.

Nun berechnet man den Mittelwert `mittel` $= \frac{\texttt{links+rechts}}{2}$ und vergleicht sein Quadrat `mittel`$^2$ mit $x$. Je nachdem, wie dieser Vergleich ausfällt, ersetzt man `links` oder `rechts` durch `mittel`, so dass nach der Ersetzung wieder `links` $\leq \sqrt{x} \leq$ `rechts` gilt.

Diesen Prozess wiederholt man solange, bis die Differenz `rechts` $-$ `links` kleiner als ein gewünschter Fehler ist.

Ergänzen Sie das folgende Programm so, dass der oben beschriebene Algorithmus realisiert wird!

```
x = 40
fehler = 0.000001
links = 0
rechts = x

while rechts - links > fehler:
    #
```

```
    # Hier ist Code zu schreiben.
    #
schaetzung = (links + rechts) / 2
print(f'{schaetzung} ist ungefähr die Wurzel aus {x}.')
print(f'{x**0.5} ist laut Python die Wurzel aus {x}.')
print(f'Der Fehler ist etwa {abs(mittel-x**0.5):.10f}.')
```

Bemerkung: Das beschriebene Verfahren heisst **Intervallschachtelung**, da $\sqrt{x}$ immer besser durch das kleiner werdende Intervall [`links`, `rechts`] eingeschachtelt wird.

**Aufgabe A14** Folgender Python-Code erzeugt Zufallszahlen zwischen 0 und 1:

```
from random import random
n = 6
while n>0:
    z = random()
    print(z)
    n = n-1
```

```
0.8696981969050243
0.7626787554309876
0.6196171861591572
0.17116177170210767
0.3170744431396274
0.49269143898456613
```

In dieser Aufgabe sollen Sie die Kreiszahl $\pi$ mit einem Zufallsexperiment annähern. Schreiben Sie dafür ein Python-Programm wie folgt:

- Bestimmen Sie zwei Zufallszahlen $x$ und $y$ zwischen 0 und 1.
- Dann ist $(x, y)$ ein Punkt im Einheitsquadrat. Stellen Sie fest, ob dieser Punkt auch im Einheitskreis liegt.
- Wiederholen Sie obiges Experiment 1000 oder 1'000'000 mal und zählen Sie die Anzahl der Punkte im Kreis.
- Berechnen Sie den Anteil der Punkte, die im Kreis liegen.
- Wie gross müsste dieser Anteil theoretisch sein? Wie lässt sich somit $\pi$ näherungsweise berechen? Wie genau ist das Resultat? Hinweis: Die Kreiszahl $\pi$ kann man wie folgt ausgeben:

```
from math import pi
print(pi)
```

Testen Sie Ihr Programm auf einem Computer!

**Aufgabe A15** Folgender Python-Code erzeugt **ganzzahlige** Zufallszahlen zwischen 0 (einschliesslich) und 3 (ausschliesslich).

```
from random import randrange
n = 6
while n>0:
    z = randrange(3)
    print(z)
    n = n-1
```

```
1
0
1
2
2
1
```

Schreiben Sie ein Programm, das sich eine ganzzahlige Zufallszahl zwischen 0 und 100 ausdenkt (jeweils einschliesslich). Der Benutzer soll die Zahl erraten. Das Programm gibt bei jedem Tipp an, ob die Zahl zu klein, zu gross oder korrekt ist. Ein Beispieldialog könnte so aussehen:

```
Ich habe mir eine natürliche Zahl zwischen 0 und 100 ausgedacht. Welche Zahl ist es?
Dein Tipp: 21
Zu klein.
Dein Tipp: 84
Zu gross.
Dein Tipp: 42
Korrekt. Herzlichen Glückwunsch!
Benötigte Versuche: 3
```

Testen Sie Ihr Programm auf einem Computer!

## 0.1 Lösungen

Hinweise zu den Symbolen:

⚒ Diese Aufgaben könnten (mit kleinen Anpassungen) an einer Prüfung vorkommen. Für die Prüfungsvorbereitung gilt: “If you want to nail it, you’ll need it”.

Diese Aufgaben sind wichtig, um das Verständnis des Prüfungsstoffs zu vertiefen. Die Aufgaben sind in der Form aber eher nicht geeignet für eine Prüfung (zu grosser Umfang, nötige «Tricks», zu offene Aufgabenstellung, etc.). **Teile solcher Aufgaben können aber durchaus in einer Prüfung vorkommen!**.

Diese Aufgaben sind dazu da, über den Tellerrand hinaus zu schauen und/oder die Theorie in einen grösseren Kontext zu stellen.

**⚒ Lösung zu A1** ex-ziffern-umordnen

**⚒ Lösung zu A2** ex-ungerade-und-quadratzahlen-ausgeben

**⚒ Lösung zu A3** ex-ganzzahlige-division

**⚒ Lösung zu A4** ex-groesste-quadratzahl-groesser-als-zahl

**⚒ Lösung zu A5** ex-zahl-binaer-ausgeben

**⚒ Lösung zu A6** ex-programm-verstehen-ea

(a)

- ```
  x = 12 und y = 18
  x = 18 und y = 12
  x = 12 und y = 6
  Das Ergebnis ist: 6
  ```

  ```
  x = 12 und y = 21
  x = 21 und y = 12
  x = 12 und y = 9
  x = 9 und y = 3
  Das Ergebnis ist: 3
  ```

- ```
  x = 13 und y = 21
  x = 21 und y = 13
  x = 13 und y = 8
  x = 8 und y = 5
  x = 5 und y = 3
  x = 3 und y = 2
  x = 2 und y = 1
  Das Ergebnis ist: 1
  ```

(b) Das Programm berechnet den ggT$(x, y)$ (= grösster gemeinsamer Teiler) von $x$ und $y$.
Dies ist der sogenannte **Euklidische Algorithmus**.

**⚒ Lösung zu A7** ex-programm-verstehen-pl

(a) Flussdiagramm

(b) Ausgabe für $a = 20$, $b = 30$:

```
23
29
```

Ausgabe für $a = 80$, $b = 100$:

```
83
89
97
```

(c) Das Programm gibt alle Primzahlen zwischen $a$ und $b$ aus.

### Lösung zu A8 ex-tageslicht

### Lösung zu A9 ex-lineare-gleichung

### Lösung zu A10 ex-quadratische-gleichung

### Lösung zu A11 ex-collatz

```
n = int(input("Gib eine natürliche Zahl > 0 ein: "))
while n>1:
    print(n)
    if n gerade:
    n = n/2
    else:
    n = 3*n+1
print(n)
```

### Lösung zu A12 ex-teilerliste-summe-perfekte-zahlen-in-python

```
n = input()
teiler = 1
while teiler <= n:
    if n % teiler == 0:
        print(teiler)
    teiler = teiler+1
```

```
n = input()
summe = 0
teiler = 1
while teiler <= n/2:
    if n % teiler == 0:
        summe = summe + teiler
print(summe)
```

```
n=input()
x=1
while x <= n:
    summe = 0
    teiler = 1
    while teiler <= n/2:
        if n % teiler == 0:
            summe = summe + teiler
    if summe == x:
        print(x)
    x = x+1
```

### Lösung zu A13 ex-wurzel-annaehern

```
x = 40
fehler = 0.000001
links = 0
rechts = x

while rechts - links > fehler:
    mittel = (links + rechts) / 2
    if mittel**2 < x:
        links = mittel
```

```
    else:
        rechts = mittel

schaetzung = (links + rechts) / 2
print(f'{schaetzung} ist ungefähr die Wurzel aus {x}.')
print(f'{x**0.5} ist laut Python die Wurzel aus {x}.')
print(f'Der Fehler ist etwa {abs(mittel-x**0.5):.10f}.')
```

```
6.324555575847626 ist ungefähr die Wurzel aus 40.
6.324555320336759 ist laut Python die Wurzel aus 40.
Der Fehler ist etwa 0.0000000425.
```

### Lösung zu A14 ex-pi-wuerfeln

```
from random import random

n = 1000      # Anzahl Punkte = Experimente
i = 0         # Laufvariable für Wiederholung
drinnen = 0   # Zählvariable für Anzahl der Punkte im Kreis

while i<n:
    i += 1          # Kurzform für i = i + 1
    x = random()
    y = random()
    # Pythagoras lässt grüssen
    r = (x*x+y*y)**0.5
    if (r<1):
        drinnen +=1  # Kurzform für drinnen = drinnen + 1

print("Drinnen", drinnen, "von", n)
anteil = drinnen/n;
print("Anteil", anteil)
# Viertelkreisfläche ist pi/4, Quadratfläche ist 1
print("Pi ist ungefähr", anteil*4)
```

```
Drinnen 781 von 1000
Anteil 0.781
Pi ist ungefähr 3.124
```

### Lösung zu A15 ex-zahlenraten

```
from random import randrange

zahl = randrange(101)
print("Ich habe mir eine natürliche Zahl zwischen 0 und 100 ausgedacht. Welche Zahl ist es?")

zahl = 42
geratene_zahl = -100
versuche = 0
while geratene_zahl != zahl:
    geratene_zahl = int(input("Dein Tipp: "))
    versuche = versuche + 1
    if zahl == geratene_zahl:
        print("Korrekt. Herzlichen Glückwunsch!")
    else:
        if geratene_zahl < zahl:
            print("Zu klein.")
        else:
            print("Zu gross.")
print("Benötigte Versuche:", versuche)
```