# Aufgabenblatt Strings

## Aufgabe 1

Bestimmen Sie jeweils den Datentyp von folgenden Werten:

```
True, 1, 'c', 1.0, [1, 2], {1:2}, (1, 2)
```

## Aufgabe 2

Schreiben Sie eine Funktion, die einen String als Argument erwartet und die Zeichen rückwärts anzeigt – eines pro Zeile.

## Aufgabe 3

Sie haben eine Liste von sogenannten Präfixen

```
praefixe='DHLMRS'
```

Und einen sogenannten Suffix

```
suffix='aus'
```

Schreiben Sie ein Programm mit diesen Zeichenketten, dass nacheinander einen Buchstaben aus der Präfix-Liste nimmt und ihn mit dem Suffix konkateniert. Die Ergebnisse sollen mit einem `print`-Statement ausgegeben werden. Die Ausgabe fängt also wie folgt an:

```
Daus
Haus
…
```

## Aufgabe 4

Informieren Sie sich mittels `help(str)` über die Methoden der Klasse String.

Betrachten Sie insbesondere folgende Methoden: __ge__,__le__, count, endswith, find, index, islower, isupper, join, lower, partition, replace, rindex, split, startswith, upper.

Nutzen Sie diese Methoden um zu testen:

a) ob ‘banane’ alphabetisch vor ‘Birne’ einzuordnen ist.
b) ob der Buchstabe ‘x’ in ‘Idefix’ auftritt.
c) wie oft der Buchstabe ‘s’ in ‘Fischers Fritz fischt frische Fische’ auftritt.
d) an welcher Stelle der Substring ‘ind’ das erste mal in ‘Blumenkinder’ auftritt.
e) ob ‘banane’ mit ‘B’ anfängt.

Nutzen Sie die Methoden zudem um:

f) eine Liste von Worten jeweils mit einem Komma zu verbinden. Aus [‘a’,’b’,’c’] soll der String ‘a,b,c’ gebildet werden.
g) das erste Wort innerhalb eines Strings zu separieren (Hinweis: Es gibt verschiedene Ansätze: Sie können beispielsweise den String mittels der Methode *split* in eine Liste der einzelnen Worte umformen und dann das erste Element aus dieser Liste auslesen oder Sie können den String so aufteilen, dass Sie drei Teile haben: den Teilstring vor dem ersten Leerschlag, den String mit dem Leerschlag und den String nach dem Leerschlag. Dazu nutzen Sie die Methode *partition*. Probieren Sie beide Wege aus.)
h) alle Vorkommen von ‘a’ in ‘banane’ durch ‘oi’ zu ersetzen.

i) die ersten 3 Vorkommen von ‘sch’ in ‘Fischers Fritz fischt frische Fische’ durch ‘s’ zu ersetzen.
j) den Index des letzten Auftretens des Buchstabens ‘n’ in ‘banane’ zu ermitteln
k) den String ‘Fischers Fritz fischt frische Fische’ in eine Liste der auftretenden Worte zu zerlegen

## Aufgabe 5

a) Sie haben in den Folien die Funktion `suche(wort, zeichen)` kennengelernt. Erweitern Sie diese so, dass die Funktion zwei weitere Parameter erwartet – die Indices des Slices, das durchsucht werden soll. Also einen Wert, der den Anfangsindex angibt und einen Wert, der den Endindex angibt.
b) Sie haben die Funktion `zeichen_in_wort(wort, zeichen)` kennengelernt. Verändern Sie diese so, dass sie die ursprüngliche Funktion `suche(wort, zeichen)` aus den Folien nutzt, statt selbst den String zu traversieren und True zurückgibt, falls zeichen in wort vorkommt und False sonst.

## Aufgabe 6

Palindrome sind Worte, die vorwärts und rückwärts gelesen dasselbe ergeben. Schreiben Sie eine Funktion, die einen String als Argument erwartet und testet, ob dieser ein Palindrom ist oder nicht. Als Rückgabewert soll ein Boolean gegeben werden. Testen Sie ihre Funktion in geeigneter Weise.

Hinweis: Zum Testen ist es manchmal gar nicht so schlecht, wenn Sie sicherstellen, dass der Test auf Palindrome nicht an der Gross-Kleinschreibung scheitert.

## Aufgabe 7

a) Sie haben gesehen, wie man mittels eines Slices ein Wort umdrehen kann. Was passiert denn, wenn man in `wort[::-1]` den Wert von -1 verändert? Testen Sie das für mehrere Werte.
b) Schreiben Sie nun eine Funktion zerlege(wort), wobei der Parameter wort als Argument einen String erwartet und diesen so zerlegt ausgibt, dass zuerst beginnend bei Index 0 jeder dritte Buchstabe ausgegeben wird, dann in einer nächsten Zeile jeder zweite noch nicht ausgedruckte Buchstabe und schliesslich in einer dritten Zeile die noch verbleibenden Buchstaben.
Hinweis: diese Funktion braucht nicht mehr als 3 Zeilen im Body.

## Aufgabe 8

Die folgenden Funktionen sollen alle überprüfen, ob ein String Kleinbuchstaben enthält. Einige davon sind allerdings falsch. Beschreiben Sie für jede Funktion, was sie in Wirklichkeit tut (vorausgesetzt der Parameter ist ein String).

```
def kleine_buchstaben1(s):
    for c in s:
        return c.islower()

def kleine_buchstaben2(s):
    for c in s:
        return 'c'.islower()

def kleine_buchstaben3(s):
    for c in s:
        flag = c.islower()
    return flag
```

```
def kleine_buchstaben4(s):
    flag = False
    for c in s:
        flag = flag or c.islower()
    return flag

def kleine_buchstaben5(s):
    for c in s:
        if not c.islower():
            return False
    return True
```

## Aufgabe 9

Sie haben in den Folien gesehen, wie man Integerwerte innerhalb eines Strings formatieren kann. Das geht auch für Floats und Strings. Dabei können Sie folgendes definieren: die Breite eines Strings, die Position eines Strings, falls er kürzer ist als die definierte Breite, die Anzahl Nachkommastellen eines Floats.

Generell: man kann nach dem Doppelpunkt den Platz angeben, der für das entsprechende Argument verwendet werden soll.

Zudem kann man angeben, wo der allenfalls entstehende leere Platz hinkommen soll:

- Nichts oder >: vor dem Text, der Text wird also quasi rechtsbündig dargestellt
- <: nach dem Text, der Text wird also quasi linksbündig dargestellt
- ^: mittig

Beispiel:

```
>>> print("Sammy has {0:<4} red {1:>16}!".format(5, "balloons"))
Sammy has 5    red         balloons!
```

Zudem kann man den Leerraum mit Zeichen füllen.
Beispiel:

```
>>> print("{:*^20s}".format("Sammy"))
*******Sammy********
```

Bei Floats:
{:2.3f} wird einen Float ausgeben, bei dem zwei Stellen vor und 3 Stellen nach dem Komma dargestellt werden.
Beispiel:

```
>>> print("{1:3.4f} oder {0:5.1f}".format(123.123456,3.256))
3.2560 oder 123.1
```

**Aufgabe:**

Schreiben Sie ein Programm, in welchem die Anwenderin / der Anwender jeweils Angaben zu den Mittgliedern in einem 4er-Team eingeben soll. Zu erfassen sind: Vorname, Nachname, Alter in Jahren, Körpergrösse in Meter, Gewicht in kg. Anschliessend sollen die entsprechenden Daten schön formatiert ausgegeben werden.

Tipp: Nutzen Sie einen for-Loop und lesen Sie die entsprechenden Daten pro Team-Mitglied ein. Speichern Sie diese Daten als formatierten String in einer Liste. Beachten Sie beim Formatieren, dass sie genügend Platz reservieren. Geben Sie dann die Liste Element für Element aus.