

# Einführung in die Programmierung mit Python

**HSLU** Hochschule  
Luzern

# Syllabus

1. Python installieren; Jupyter Notebooks
2. **Grundlagen: Arithmetik, Variablen und Datentypen**
3. Methoden; Listen; Bedingte Anweisungen
4. Schleifen
5. Benutzerdefinierte Funktionen
6. Datenaufbereitung und Grafische Darstellungen

# Python-Grundlagen

## Zahlen

Mathematik mit Ganzzahlen funktioniert wie erwartet:

```
In [1]: 5 + 2
```

```
Out[1]: 7
```

```
In [2]: 2 - 4
```

```
Out[2]: -2
```

```
In [3]: 7 * (6 + 1) # Klammern funktionieren wie üblich
```

```
Out[3]: 49
```

```
In [4]: 2 ** 3 # zwei hoch drei
```

```
Out[4]: 8
```

Alles in einer Zelle:

```
In [5]: 5 + 2  
2 - 4  
7 * (6 + 1)  
2 ** 3
```

```
Out[5]: 8
```

Was ist passiert? Jupyter gibt nur das Ergebnis des *letzten* Ausdrucks in einer Zelle aus. Wir können das mit der `print`-Funktion korrigieren:

```
In [6]: print(5 + 2)  
print(2 - 4)  
print(7 * 7)  
print(2 ** 3)
```

```
7  
-2  
49  
8
```

## Was ist mit Division?

```
In [7]: 2 / 3
```

```
Out[7]: 0.6666666666666666
```

Das funktioniert auch, gibt aber eine andere Art von Zahl zurück: eine Gleitkommazahl oder `float`. Dies gilt auch, wenn die Division prinzipiell exakt durchgeführt werden könnte:

```
In [8]: 6 / 2
```

```
Out[8]: 3.0
```

Für einen Menschen repräsentieren `3.0` (ein `float`) und `3` (eine Ganzzahl oder `int`) dieselbe Zahl, sie werden jedoch im Speicher unterschiedlich dargestellt; wir sagen, dass diese beiden Objekte einen unterschiedlichen **Typ** haben. Wir können den Typ eines Objekts wie folgt herausfinden:

```
In [9]: type(3)
```

```
Out[9]: int
```

```
In [10]: type(3.0)
```

```
Out[10]: float
```

Mathematik mit Gleitkommazahlen kann knifflig sein, da sie mit begrenzter Genauigkeit dargestellt werden, was bedeutet, dass nicht alle Zahlen darstellbar sind:

In [11]: `1 - 0.9`

Out[11]: `0.0999999999999998`

# Variablen

Eine Variable ist ein benannter Speicherort. Sie wird mit " = " zugewiesen.

In [12]:

```
a = 2  
b = 4  
c = a + b  
print(c)
```

6

Können Sie erraten, was der folgende Code tut?

In [13]:

```
a = 2  
a = a + 1  
print(a)
```

3

Der folgende Code ist äquivalent:

In [14]:

```
a = 2  
a += 1 # Kurzform für a = a + 1  
print(a)
```

3



Variablennamen können aus Buchstaben, Zahlen und dem Unterstrich bestehen. Sie dürfen nicht mit einer Zahl beginnen. Python ist case-sensitive: A ist nicht dasselbe wie a .

# Zuweisung versus Gleichheit

Wir haben gerade gesehen, dass Variablen mit = zugewiesen werden.

In [15]:

```
a = 3  
print(a)
```

3

Was, wenn wir prüfen wollen, ob zwei Zahlen gleich sind? Erster Versuch:

In [16]:

```
# Entkommentieren Sie die nächste Zeile und führen Sie die Zelle aus  
# 3 = 3
```

Das hat offensichtlich nicht funktioniert. Der richtige Weg ist die Verwendung von == :

```
In [17]: 3 == 3
```

```
Out[17]: True
```

```
In [18]: 1 == 2
```

```
Out[18]: False
```

Das zurückgegebene Objekt ist vom Typ `bool` (ein "Boolean"):

```
In [19]: type(True)
```

```
Out[19]: bool
```

Ein `bool` kann einen von zwei Werten annehmen: `True` oder `False`.

Sie werden von *relationalen Operatoren* zurückgegeben: `<`, `<=`, `>`, `>=`, `==` (Gleichheit), `!=` (Ungleichheit) und können mit den *logischen Operatoren* `and`, `or` und `not` kombiniert werden.

```
In [20]: 1 <= 2 < 4
```

```
Out[20]: True
```

```
In [21]: 1 < 2 and 2 < 1
```

```
Out[21]: False
```

```
In [22]: not(1 < 2)
```

```
Out[22]: False
```

# Zeichenketten

Zeichenketten enthalten Text. Sie werden mit einfachen oder doppelten Anführungszeichen erstellt:

In [23]:

```
s1 = "Python"  
s2 = ' ist einfach.'  
s3 = s1 + s2 # Verkettung
```

In [24]:

```
type(s3)
```

Out[24]:

```
str
```

Das funktioniert nicht:

In [25]:

```
a = 3 # eine Ganzzahl  
b = "4" # eine Zeichenkette  
# Entkommentieren und ausführen:  
# a + b
```

Wir müssen die Zeichenkette zuerst konvertieren:

```
In [26]: a + int(b)
```

```
Out[26]: 7
```

Wir können auch in die andere Richtung konvertieren:

```
In [27]: a = 3  
b = str(a)
```

```
In [28]: type(b)
```

```
Out[28]: str
```

Das ist nützlich für die Ausgabe:

```
In [29]: height = 1.89  
print("Ich bin " + str(height) + "m gross.")
```

```
Ich bin 1.89m gross.
```

Eine Möglichkeit, eine Zeichenkette zu erhalten, besteht darin, den Benutzer nach einer Eingabe zu fragen:

```
In [30]: mystr = input("Wie heisst du? ")  
print(mystr)
```

Simon

# Übung

Schreiben Sie Code in die Zelle unten, der den Benutzer nach seinem Alter fragt und dann das Alter in Hundejahren ausgibt (d.h. geteilt durch 7).

Beispiel-Eingabe:

```
Wie alt bist du?
```

Wenn der Benutzer 28 eingibt, sollte dies die folgende Ausgabe erzeugen:

```
Dein Alter in Hundejahren ist 4.0.
```

Beachten Sie, dass `input` immer eine Zeichenkette zurückgibt, also müssen Sie sie in `int` (oder `float`) konvertieren, um damit mathematische Operationen durchzuführen.

In [ ]:

# Sequenztypen: Container mit ganzzahliger Indizierung

## Zeichenketten

Wir haben bereits Zeichenketten kennengelernt: Sie enthalten Text und werden mit einfachen oder doppelten Anführungszeichen erstellt:

```
In [31]: s1 = "Python"; s2 = ' ist einfach.'; s3 = s1 + s2 # Verkettung  
print(s3)
```

Python ist einfach.

Die `len`-Funktion gibt die Länge einer Zeichenkette zurück:

```
In [32]: len(s3)
```

```
Out[32]: 19
```

Wir können Zeichen aus einer Zeichenkette durch *Indizierung* auswählen:

In [33]: `print(s3)`

```
Python ist einfach.
```

In [34]: `s3[0] # Beachten Sie die nullbasierte Indizierung`

```
Out[34]: 'P'
```

In [35]: `s3[-1] # Negative Indizes zählen von rechts:`

```
Out[35]: '.'
```

Wir können auch mehrere Elemente auswählen ("*Slicing*"). Dies funktioniert für alle Sequenztypen (Listen, NumPy-Arrays, ...).

In [36]: `print(s3)`

```
Python ist einfach.
```

In [37]: `s3[0:2] # Elemente 0 und 1; Linker Endpunkt ist enthalten, rechter Endpunkt au.`

Out[37]: `'Py'`

In [38]: `s3[0:6:2] # Start:Stop:Schritt`

Out[38]: `'Pto'`

In [39]: `s3[::-1] # Start und Stop können weggelassen werden; Standard ist 0 und Len(string)-1`

Out[39]: `'.hcafnie tsi nohtyP'`

# Übung

Verwenden Sie Slicing, um die Teilzeichenkette `mich` aus der Zeichenkette `s` unten zu extrahieren.

```
In [40]: s = 'Warum habe ich mich nur hierfür angemeldet?'
```

# Hausaufgabe

Anfängerübungen 1-4 von <https://holypython.com/beginner-python-exercises/>