

Algorithmen und Datenstrukturen

A2. Eine sehr kurze Einführung in Python

Marcel Lüthi and Gabriele Röger

Universität Basel

3. März 2021

A2.1 Python

A2.2 Kurzer Sprachüberblick

A2.3 Selectionsort in Python

A2.1 Python



- ▶ interpretierte High-Level-Programmiersprache
- ▶ unterstützt imperative, objekt-orientierte und funktionale Programmierung
- ▶ gut lesbarer Code
- ▶ hohe Produktivität: für gleiche Funktionalität deutlich weniger Code erforderlich als z.B. mit Java
- ▶ umfangreiche Bibliotheken
- ▶ Ausführung oftmals langsamer als mit kompilierten Sprachen
- ▶ benannt nach Monty Python
(englische Komikergruppe aus den 1970ern)

Python-Interpreter

- wir verwenden Python 3.x
- Das Programm `python3` kann Programme ausführen oder als interaktiver Interpreter verwendet werden:

Python-Interpreter

```
Python 2.7.18 (default, Aug  4 2020, 11:16:42)
[GCC 9.3.0] on linux2
Type "help", "copyright", "credits" or "license"
for more information.

>>> 5 * 4
20
>>> exit()  (Linux: Strg+d)
```

Ressourcen

- Python: <https://www.python.org/downloads/> oder aus Paketrepository (Ubuntu: `apt install python3`)
- Referenz und Tutorial: <https://docs.python.org/3/>
- IDE: z.B. PyCharm (<https://www.jetbrains.com/pycharm/>)
- oder Editor: z.B. `emacs` oder `vim` (falls bereits damit vertraut), sonst z.B. `Geany` (<https://www.geany.org/>)
- Style-Checker: z.B. Flake 8 (<http://flake8.pycqa.org/>) (Ubuntu: `apt install python3-flake8`)

A2.2 Kurzer Sprachüberblick

Dynamische Typisierung

- Variablen haben keinen Typ, sondern nur die Objekte, auf die sie referenzieren.
- Typprüfung erst zur Laufzeit

```
>>> a = 3
>>> a/2
1.5
>>> a = "jetzt referenziert a einen String"
>>> a/2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

Einrückung statt Klammern

Einrückung zur Abgrenzung von Anweisungsblöcken
(wie Funktionen, Schleifenrümpfen, etc.)

```
def count(to):
    for val in range(to): # val = 0, ..., to-1
        print(val + 1)
    print("fertig")
```

Java: geschweifte Klammern

Tab ≠ Leerzeichen
Empfehlung: 4 Leerzeichen pro Ebene

range

- ▶ `range(stop)`: Generiert Zahlen von 0 bis stop - 1
→ `range(3)` liefert 0, 1, 2
- ▶ `range(start, stop[, step])`:
Generiert Zahlen von start bis (exklusive) stop mit Schrittweite step
→ `range(3, 11, 2)` liefert 3, 5, 7, 9
→ `range(2, -3, -1)` liefert 2, 1, 0, -1, -2
→ `range(2, 5)` liefert 2, 3, 4

Listen und Tupel

- ▶ Listen und Tupel enthalten Sequenzen von Objekten
- ▶ Listen werden in **eckigen Klammern** notiert:
`[13, "Hund", "Katze"]`
- ▶ Tupel werden in **runden Klammern** notiert:
`("Superpapagei", 31, ["???", "Bob"])`
- ▶ Unterschied
 - ▶ Listen sind **veränderlich** (mutable), man kann Elemente einfügen und entfernen.
 - ▶ Tupel sind **unveränderlich** (immutable), sie enthalten stets die gleichen Objekte in der gleichen Reihenfolge (diese können aber verändert werden).

Indizierung und Manipulation

- ▶ Sequenzen können von vorne (nicht-negative Zahlen) oder hinten (negative Zahlen) indiziert werden.
- ▶ Das vorderste Element hat Index 0.
`(4, 5, 2, 9)[1]` referenziert 5.
- ▶ Das letzte Element hat Index -1.
`(4, 5, 2, 9)[-2]` referenziert 2.
- ▶ Bei veränderlichen Sequenzen ist neue Zuweisung möglich.
`a[2] = 4` für Liste a
- ▶ Listen können mit `append` um ein Element erweitert werden.
`a.append(8)` fügt 8 hinten an Liste a an.

Beispiel Indizierung und Manipulation

```
>>> fibonacci = (1, 1, 2, 3, 5, 8)
>>> print(fibonacci[0], fibonacci[2], fibonacci[-1])
1 2 8
>>> fibonacci_list = list(fibonacci)
>>> print(fibonacci_list)
[1, 1, 2, 3, 5, 8]
>>> fibonacci_list.append(14)
>>> print(fibonacci_list)
[1, 1, 2, 3, 5, 8, 14]
>>> fibonacci_list[-1] = 13
>>> print(fibonacci_list)
[1, 1, 2, 3, 5, 8, 13]
```

Unveränderlichkeit Tupel

```
>>> l = (3, "Hund", ["Maus"])
>>> l[2].append("Haus")
>>> l
(3, 'Hund', ['Maus', 'Haus'])
>>> l[1] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Mehr zu Tupeln

- ▶ **Tupel Unpacking** „entpackt“ Werte auf der rechten Seite, um sie Variablen in Tupel auf der linken Seite zuzuweisen.
`(number, name) = (3, "Johann Gambolputty")`
- ▶ Die Klammern um Tupel sind allgemein optional, wenn dies keine Mehrdeutigkeit verursacht.
- ▶ Tupel Unpacking daher auch ohne Klammern möglich:
`number, name = 3, "Johann Gambolputty"`
- ▶ Wird oft verwendet, um die Werte zweier Variablen zu tauschen:
`var1, var2 = var2, var1`
- ▶ Achtung: einelementige Tupel schreibt man mit abschliessendem Komma: `(2,)`

Kontrollstrukturen: if, elif, else

```
if x > 0:
    print("x ist positiv")
elif x == 0:
    print("x ist Null")
else:
    print("x ist negativ")
```

Bedingungen: Logische Verknüpfung mit `and`, `or`, `not`
z.B. `x > 5 and y < 3`

Kontrollstrukturen: while, for

Countdown von 9 bis 1 (zwei Versionen):

```
x = 9
while x > 0:
    print(x)
    x -= 1

for x in range(9, 0, -1):
    print(x)
```

- ▶ Sprung aus Schleife mit `break`
- ▶ Sprung zur nächsten Schleifeniteration mit `continue`

Funktionen und Main-Funktion

```
import sys

def power(base, exponent):
    return base ** exponent

def main():
    base, exp = int(sys.argv[1]), int(sys.argv[2])
    print(power(base, exp))

if __name__ == "__main__":
    # called if file is executed but not at import
    main()
```

A2.3 Selectionsort in Python

Beispiel: Selectionsort

```
def selection_sort(a):
    """Selectionsort sorting algorithm

    >>> selection_sort([3, 1, 6, 3, 2])
    [1, 2, 3, 3, 6]
    >>> selection_sort([])
    []
    """
    for i in range(len(a) - 1):
        min_index = i
        for j in range(i + 1, len(a)):
            if a[j] < a[min_index]:
                min_index = j
        a[i], a[min_index] = a[min_index], a[i]
    return a
```

Beispiel: Selectionsort

```
selection_sort.py
import random

def selection_sort(a):
    Siehe vorherige Folie

if __name__ == "__main__":
    a = [n for n in range(40)]  # [0, 1, ... 39]
    random.shuffle(a)  # Array zufällig mischen
    print(a)
    selection_sort(a)
    print(a)
```

Beispiel: Selectionsort

- ▶ Unitest mit `python3 -m doctest selection_sort.py`
- ▶ Stylecheck mit `python3 -m flake8 selection_sort.py`
- ▶ Ausführen mit `python3 selection_sort.py`