# Python

N. Kaelin

23. Februar 2019

## Inhaltsverzeichnis

**Teil I**

# Lektion 1: Variablen und Datentypen

## 1 Datentypen

- Variablen bezeichnen keinen bestimmten Typ.
- Dynamische Typdeklaration
  - **Automatische Zuweisung** des Datentyps bei Deklaration
  - Datentyp ist während dem Programmablauf **veränderbar**
  - Wert- und Typänderung erlaubt!

| **Datentyp** | **Beschreibung** | **False-Wert** |
|---|---|---|
| `NoneType` | Indikator für nichts, keinen Wert | `None` |
| **Numerische Datentypen** | | |
| `int` | Ganze Zahlen | `0` |
| `float` | Gleitkommazahlen | `0.0` |
| `bool` | Boolesche Werte | `False` |
| `complex` | Komplexe Zahlen | `0 + 0j` |
| **Sequenzielle Datentypen** | | |
| `str` | Zeichenketten oder Strings | `"` |
| `list` | Listen (veränderlich) | `[]` |
| `tuple` | Tupel (unveränderlich) | `()` |
| `bytes` | Sequenz von Bytes (unveränderlich) | `b"` |
| `bytearray` | Sequenz von Bytes (veränderlich) | `bytearray(b")` |
| **Assoziative Datentypen** | | |
| `dict` | Dictionary (Schlüssel-Wert-Paare) | |
| **Mengen** | | |
| `set` | Menge mit einmalig vorkommenden Objekten | `set()` |
| `frozenset` | Wie set jedoch unveränderlich | `frozenset()` |

- Python erkennt den Datentyp automatisch
- Python ordnet jeder Variablen den Datentyp zu
- Datentypen prüfen:
  ```
  type(object)
  isinstance(object, ct)
  ```
- Python achtet auf Typverletzungen
- Python kennt keine implizite Typumwandlung

### 1.1 Numerische Datentypen

- `bool`
- `int`
- `float`
- `complex`

### 1.1.1 Arithmetische Operationen

| Operator | Beschreibung |
|---|---|
| `x + y` | Summe von `x` und `y` |
| `x - y` | Differenz von `x` und `y` |
| `x * y` | Produkt von `x` und `y` |
| `x / y` | Quotient von `x` und `y` |
| `x // y` | Ganzzahliger Quotient[1] von `x` und `y` |
| `x % y` | Rest der Division[1] von `x` durch `y` |
| `+x` | Positives Vorzeichen |
| `-x` | Negatives Vorzeichen |
| `abs(x)` | Betrag von `x` |
| `x**y` | Potenzieren, $x^y$ |

[1]Nicht definiert für den Datentyp `complex`

**Achtung**: `x++` und `x--` gibt es **nicht**, aber `x += 1, x -0 1, x *= 2, ...`

### 1.1.2 Vergleichende Operatoren

| Operator | Beschreibung |
|---|---|
| == | wahr, wenn `x` und `y` gleich sind |
| != | wahr, wenn `x` und `y` verschieden sind |
| < | wahr, wenn `x` kleiner als `y` ist[2] |
| <= | wahr, wenn `x` kleiner oder gleich `y` ist[2] |
| > | wahr, wenn `x` grösser als `y` ist[2] |
| >= | wahr, wenn `x` grösser oder gleich `y` ist[2] |

[2]Nicht definiert für den Datentyp `complex`

### 1.1.3 Bitweise Operatoren für den Datentypen `int`

| Operator | Beschreibung |
|---|---|
| `x & y` | bitweises UND von `x` und `y` |
| `x \| y` | bitweises ODER von `x` und `y` |
| `x ^y` | bitweises EXOR von `x` und `y` |
| `~x` | bitweises Komplement von `x` |
| `x « n` | Bit-Verschiebung um `n` Stellen nach links |
| `x » n` | Bit-Verschiebung um `n` Stellen nach rechts |

### 1.1.4 Methoden nur dür den Datentyp `complex`

| Methode | Beschreibung |
|---|---|
| `x.real` | Realteil von `x` als Gleitkommazahl |
| `x.imag` | Imaginärteil von `x` als Gleitkommazahl |
| `x.conjugate()` | Liefert die zu `x` konjugiert komplexe Zahl |

## 1.2 Sequentielle Datentypen

- `str`
- `list`
- `tuple`
- `bytes`
- `bytearray`

Die folgenden Operatoren sind für **alle** sequentiellen Datentypen definiert:

| Operator | Beschreibung |
|---|---|
| `x in s` | Prüft, ob `x` in `s` enthalten ist. |
| `x not in s` | Prüft, ob `x` nicht in `s` enthalten ist. |
| `s + t` | Verkettung der beiden Sequenzen `s` und `t`. |
| `s * n` | Verkettung von `n` Kopien der Sequenz `s`. |
| `s[i]` | Liefert das `i`-te Element von `s`. |
| `s[i:j]` | Liefert den Ausschnitt aus `s` von `i` bis `j`. |
| `s[i:j:k]` | Liefert jedes `k`-te Element im Ausschnitt von `s` zwischen `i` und `j`. |
| `len(s)` | Liefert die Anzahl Elemente in der Sequenz `s`. |
| `max(s)` | Liefert das grösste Element in `s` (sofern eine Ordnung definiert ist). |
| `min(s)` | Liefert das kleinste Element in `s` (sofern eine Ordnung definiert ist). |
| `s.index(x)` | Liefert den Index des ersten Vorkommens von `x` in `s`. |
| `s.count(x)` | Zählt, wie oft `x` in `s` vorkommt. |

## 1.3 Assoziative Datentypen

- dict

| Operator | Beschreibung |
|---|---|
| `len(d)` | Liefert die Anzahl Schlüssel-Wert-Paare in `d` |
| `d[k]` | Zugriff auf den Wert mit dem Schlüssel `k` |
| `k in d` | Liefert `True`, wenn der Schlüssel `k` in `d` ist. |
| `k not in d` | Liefert `True`, wenn der Schlüssel `k` nicht in `d` ist. |

| Operator | Beschreibung |
|---|---|
| `d.clear()` | Löscht alle Elemente aus dem Dictionary. |
| `d.copy()` | Erstellt eine Kopie des Dictionaries. |
| `d.get([k,[x]])` | Gibt den Wert des Schlüssels `k` zurück, ansonsten den Wert [x]. |
| `d.items()` | Gibt eine Liste der Schlüssel-Wert-Paare als Tuple zurück. |
| `d.keys()` | Gibt eine Liste aller Schlüsselwerte zurück. |
| `d.update(d2)` | Fügt ein Dictionary `d2` zu `d` hinzu. |
| `d.pop(k)` | Entfernt das Element mit Schlüssel `k`. |
| `d.popitem()` | Entfernt das zuletzt eingefügte Schlüssel-Wert-Paar. |
| `d.setdefault(k,[x])` | Setzt den Wert [x] für den Schlüssel `k`. |

## 1.4 Mengen

- set
- frozenset