## Datengewinnung und -analyse

# 3. Programmierung mit Python

Markus Schanta

# Agenda

1. **Theorie & Motivation:** Vorteile von Python
2. **Zuweisungen:** Werte in Variablen abspeichern
3. **Datentypen:** Einfache & Container-Datentypen
4. **Verzweigungen:** Bedingte Anweisungen
5. **Schleifen:** Wiederholte Ausführung von Code
6. **Funktionen:** Wiederverwendbare Codeblöcke

# Python: ein Überblick


- Interpretierte, universelle Programmiersprache
- 1991 erstmals von Guido von Rossum veröffentlicht
- Aktuelle (02.12.2025) Version: 3.14.1
  - Python 2.0 veröffentlicht 2000
  - Python 3.0 veröffentlicht 2008
  - Python 2.7 end of life: 2015, 2020

# Vorteile von Python

## Universell

Python ist geeignet, um für unterschiedliche Problemstellungen eingesetzt zu werden

## Erweiterbar

Python bietet ein riesiges Ökosystem an Erweiterungen (Libraries) für diverseste Zwecke

## Prägnante Syntax

Die Syntax von Python ist einfach zu verstehen und auf das Wesentliche reduziert

## Interpretiert

Interaktive Ausführungsmodi möglich

## Leicht zugänglich

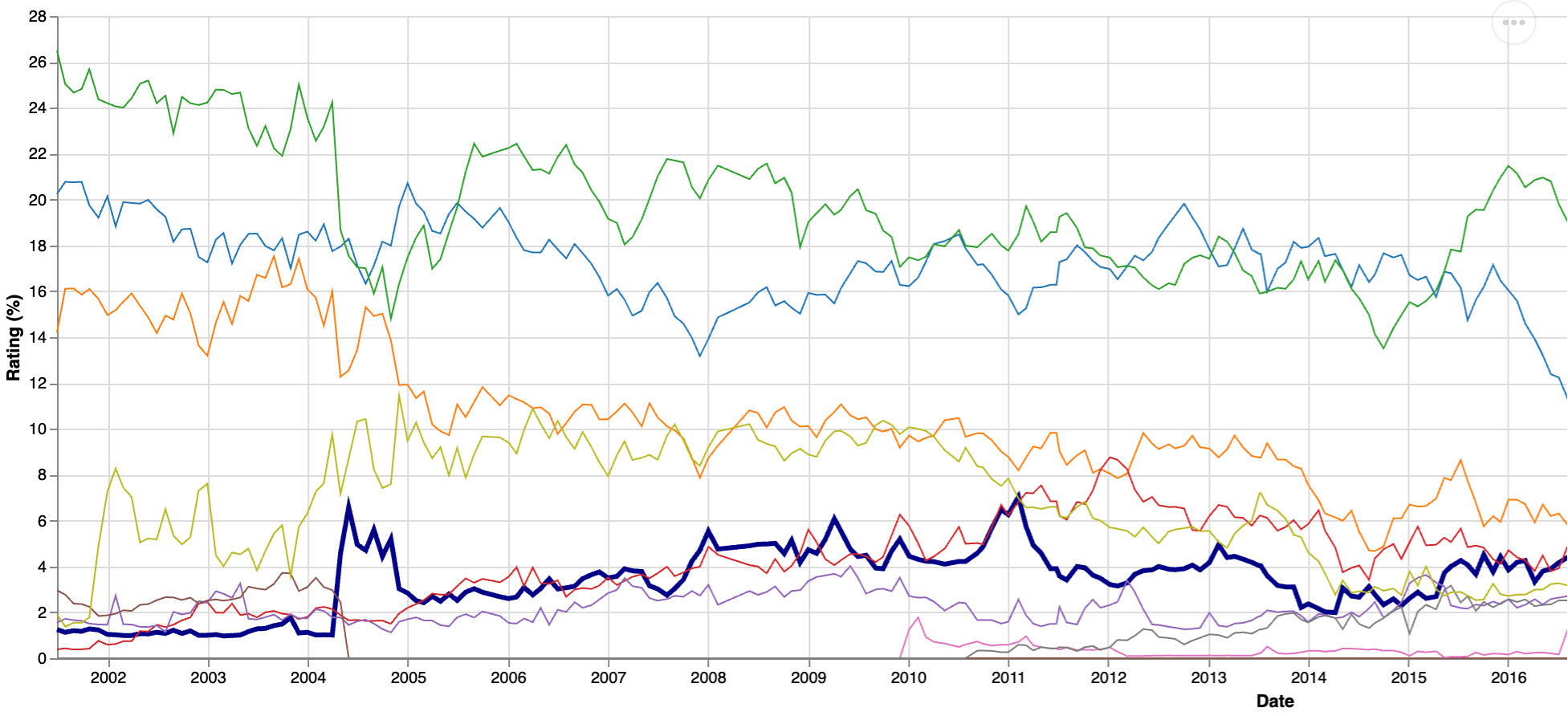
Leicht zu erlernen, sehr viele Ressourcen

## Verbreitet

In vielen Organisationen verbreitet, große Community

# Python TIBOE Index

Der TIOBE Programming Community Index ist ein Ranking von Programmiersprachen nach ihrer Popularität.



# Zuweisungen

# Zuweisungen

- Grundstruktur einer Zuweisung:

```
area = 3 * 5
```

# Zuweisungen

- Grundstruktur einer Zuweisung:

```
area = 3 * 5
```

- Der Variable `area` wird der Wert `3 * 5` zugewiesen.

# Zuweisungen

- Grundstruktur einer Zuweisung:

```
area = 3 * 5
```

- Der Variable `area` wird der Wert `3 * 5` zugewiesen.
- Der Wert kann eine Konstante, eine Berechnung, das Ergebnis eines Funktionsaufrufs oder das Ergebnis einer anderen Zuweisung sein.

# Zuweisungen

- Grundstruktur einer Zuweisung:

```
area = 3 * 5
```

- Der Variable `area` wird der Wert `3 * 5` zugewiesen.
- Der Wert kann eine Konstante, eine Berechnung, das Ergebnis eines Funktionsaufrufs oder das Ergebnis einer anderen Zuweisung sein.
- Der zugewiesene Wert kann später im Programm durch den Variablennamen abgerufen werden.

# Variablennamen (1/3)

# Variablennamen (1/3)

1. **Zeichen**
    - Variablennamen dürfen Buchstaben (`a–z`, `A–Z`), Zahlen (`0–9`) oder Unterstriche (`_`) enthalten.

# Variablennamen (1/3)

1. **Zeichen**
    - Variablennamen dürfen Buchstaben (`a–z`, `A–Z`), Zahlen (`0–9`) oder Unterstriche (`_`) enthalten.
2. **Länge**
    - Es gibt keine festgelegte maximale Länge für Variablennamen, aber es ist ratsam, sie kurz und prägnant zu halten, um die Lesbarkeit zu erhöhen.

# Variablennamen (1/3)

1. **Zeichen**
    - Variablennamen dürfen Buchstaben (`a–z`, `A–Z`), Zahlen (`0–9`) oder Unterstriche (`_`) enthalten.
2. **Länge**
    - Es gibt keine festgelegte maximale Länge für Variablennamen, aber es ist ratsam, sie kurz und prägnant zu halten, um die Lesbarkeit zu erhöhen.
3. **Groß- und Kleinschreibung**
    - Python unterscheidet zwischen Groß- und Kleinschreibung.
    - Beispiel: `variable`, `Variable` und `VARIABLE` sind drei verschiedene Variablen.

# Variablennamen (2/3)

# Variablennamen (2/3)

4. **Schlüsselwörter**
    - Variablennamen dürfen keine reservierten Schlüsselwörter (z.B. `if`, `for`, `while`, `def`, etc.) sein, die eine besondere Bedeutung in Python haben.

# Variablennamen (2/3)

4. **Schlüsselwörter**
    - Variablennamen dürfen keine reservierten Schlüsselwörter (z.B. `if`, `for`, `while`, `def`, etc.) sein, die eine besondere Bedeutung in Python haben.
5. **Konventionen**
    - Es ist ratsam, aussagekräftige Namen zu verwenden, die den Zweck einer Variable widerspiegeln (z.B. `total_price` statt `tp`).
    - Es ist üblich, Variablennamen in Kleinbuchstaben zu schreiben und Wörter durch Unterstriche zu trennen (z.B. `my_variable`).

# Variablennamen (3/3)

In Python müssen Variablennamen mit einem Buchstaben oder Unterstrich beginnen, dürfen nur Buchstaben, Zahlen und Unterstriche enthalten und dürfen keine reservierten Schlüsselwörter verwenden.

**Gültige Variablennamen**

- `user_age`
- `age`
- `user_name`
- `total3`
- `_value`
- `myVariable`

**Ungültige Variablennamen**

- `3total` (beginnt mit einer Zahl)
- `user-name` (Bindestrich ist nicht erlaubt)
- `if` (Schlüsselwort)
- `my variable` (Leerzeichen nicht erlaubt)

# Python Datentypen

# Python Datentypen

- `int`

  repräsentiert **ganze Zahlen** beliebiger Größe und unterstützt gängige arithmetische Operationen wie Addition, Subtraktion, Multiplikation und Division.

# Python Datentypen

- `int`

  repräsentiert **ganze Zahlen** beliebiger Größe und unterstützt gängige arithmetische Operationen wie Addition, Subtraktion, Multiplikation und Division.

- `float`

  repräsentiert **Gleitkomma-Zahlen** (=reelle Zahlen mit Dezimalstellen) und wird für präzise Berechnungen mit Brüchen, Exponentialnotation und großen oder kleinen Zahlen verwendet.

# Python Datentypen

- `int`

  repräsentiert **ganze Zahlen** beliebiger Größe und unterstützt gängige arithmetische Operationen wie Addition, Subtraktion, Multiplikation und Division.

- `float`

  repräsentiert **Gleitkomma-Zahlen** (=reelle Zahlen mit Dezimalstellen) und wird für präzise Berechnungen mit Brüchen, Exponentialnotation und großen oder kleinen Zahlen verwendet.

- `str`

  repräsentiert **Zeichenketten** (Text) und unterstützt Operationen wie Verkettung und Wiederholung.

# Python Datentypen

- **`int`**

  repräsentiert **ganze Zahlen** beliebiger Größe und unterstützt gängige arithmetische Operationen wie Addition, Subtraktion, Multiplikation und Division.

- **`float`**

  repräsentiert **Gleitkomma-Zahlen** (=reelle Zahlen mit Dezimalstellen) und wird für präzise Berechnungen mit Brüchen, Exponentialnotation und großen oder kleinen Zahlen verwendet.

- **`str`**

  repräsentiert **Zeichenketten** (Text) und unterstützt Operationen wie Verkettung und Wiederholung.

- **`bool`**

  repräsentiert **Wahrheitswerte** (`True` oder `False`) und wird für logische Operationen verwendet.

# Python Container-Datentypen

# Python Container-Datentypen

- `list`

  ist ein geordneter Container-Datentyp, der eine beliebig lange **Folge von Elementen** speichern kann und durch eckige Klammern `[]` dargestellt wird

# Python Container-Datentypen

- `list`

  ist ein geordneter Container-Datentyp, der eine beliebig lange **Folge von Elementen** speichern kann und durch eckige Klammern `[]` dargestellt wird

- `dict`

  ist ein ungeordneter Container-Datentyp, der **Schlüssel-Wert-Paare** speichert und durch geschweifte Klammern `{}` dargestellt wird, wobei jeder Schlüssel eindeutig sein muss und auf einen Wert zugreifen kann.

# Python Container-Datentypen

- `list`

  ist ein geordneter Container-Datentyp, der eine beliebig lange **Folge von Elementen** speichern kann und durch eckige Klammern `[]` dargestellt wird

- `dict`

  ist ein ungeordneter Container-Datentyp, der **Schlüssel-Wert-Paare** speichert und durch geschweifte Klammern `{}` dargestellt wird, wobei jeder Schlüssel eindeutig sein muss und auf einen Wert zugreifen kann.

- `set`

  ist ein ungeordneter Container-Datentyp, der eine **Sammlung einzigartiger Elemente** speichert und durch geschweifte Klammern `{}` dargestellt wird, wobei doppelte Werte automatisch entfernt werden.

# Python Built-in-Funktionen

Python bietet eine Vielzahl von Built-in-Funktionen, die vordefiniert sind und häufige Programmieraufgaben erleichtern, darunter Funktionen zur Typumwandlung, zur mathematischen Berechnung oder zur Datenbearbeitung.

| | | | |
|---|---|---|---|
| `print()` | `type()` | `len()` | `abs()` |
| `min()` | `max()` | `sum()` | `mean()` |
| `sorted()` | `reversed()` | `round()` | `chr()` |

# Fragen?

# Nächste Schritte

- Review Folien & Materialien der heutigen Einheit (bereits auf Moodle verfügbar)
- Angabe 1. Gruppenübung ab nächster Woche auf Moodle verfügbar
- Nächste Einheit: Mo. 15.12. 18:30

# Danke und schönen Abend!