

# Informatik 1

## Python I - Allgemeines

Jonas Miederer

DHBW Stuttgart - Campus Horb

3. Dezember 2020



# Outline:

- ① Über Python
- ② Interpreter  
Aufgaben
- ③ Bibliographie

# Inhaltsverzeichnis

① Über Python

② Interpreter  
Aufgaben

③ Bibliographie

# Python Historie

- Entwickelt in den 1990er Jahren vom Niederländer **Guido van Rossum** im Rahmen seiner Tätigkeit am Centrum Wiskunde & Informatica in Amsterdam
- Python Version 1.0 ist im Januar 1994 erschienen
- Python Version 2.0 ist im Oktober 2000 erschienen
- Python Version 3.0 ist im Dezember 2008 erschienen
- Nachdem lange Zeit sowohl Python 2 als auch Python 3 unterstützt wurden, wird der offizielle Support von Python 2 zum 31.12.2019 eingestellt. Daher sollte ausschließlich Python 3 verwendet werden

# Warum Python?

Python ist äußerst einsteigerfreundlich und intuitiv in der Anwendung

## Beispiel

Ausgabe von 'Hello World' im Vergleich: C (links), Java (mitte) und Python (rechts)

```
#include <stdio.h>
```

```
int main(int argc, char** argv)
{
    printf("Hello World\n");
}
```

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello World");
    }
}
```

```
print("Hello World")
```

# Python: Ziele

Ursprüngliche Ziele von Python, die 1999 von van Rossum im Essay "Computer Programming for Everybody" festgehalten wurden [1]:

- Einfache und intuitive Sprache, die dennoch sehr umfangreich und mächtig ist
- Die Sprache soll Open Source (quelloffen) sein, sodass die Sprache als Gemeinschaftsprojekt weiterentwickelt werden kann
- Einfach zu lesender Quellcode, der dem Englischen möglichst nahe sein soll
- Hohe Praktikabilität, es sollen schnelle Entwicklungszeiten möglich sein

# Python: Eigenschaften

## Python...

- ist sowohl **einsteigerfreundlich**, als auch für professionelle Anwendungen und umfassende Business-Anwendungen geeignet
- ist eine **universelle** Programmiersprache, d.h. sie ist nicht nur domänenspezifisch einsetzbar (Häufige Anwendungsgebiete: Web (serverseitig), AI, Desktop-Anwendungen, mathematische Anwendungen, ...)
- ist eine **höhere** Programmiersprache
- ist (meist) eine **interpretierte** Programmiersprache
- ist eine **Multiparadigmensprache**, z.B. Objektorientiert, aspektorientiert, funktional, imperativ, prozedural, ...
- ist **dynamisch typisiert**
- wird häufig als **Skriptsprache** genutzt

## Python: Grundkonzepte

Festgehalten in "The Zen of Python":

*Beautiful is better than ugly.*

*Explicit is better than implicit.*

*Simple is better than complex.*

*Complex is better than complicated.*

*Flat is better than nested.*

*Sparse is better than dense.*

*Readability counts.*

*Special cases aren't special enough to break the rules.*

*Although practicality beats purity.*

*Errors should never pass silently.*

*Unless explicitly silenced.*

*In the face of ambiguity, refuse the temptation to guess.*

*There should be one– and preferably only one –obvious way to do it.*

*Although that way may not be obvious at first unless you're Dutch.*

*Now is better than never.*

*Although never is often better than \*right\* now.*

*If the implementation is hard to explain, it's a bad idea.*

*If the implementation is easy to explain, it may be a good idea.*

*Namespaces are one honking great idea – let's do more of those!*



# Python: Besonderheiten

## Einrückungen

Anders als bei den meisten anderen Programmiersprachen werden bei Python einzelne Codeblöcke (Scopes) nicht durch geschweifte Klammern gekennzeichnet, sondern aus Gründen der Übersichtlichkeit durch Einrückungen. Ein neuer Codeblock wird also eingerückt, ist der Block abgeschlossen erfolgt eine Ausrückung.

## Tabulator vs. Leerzeichen

Es ist dem Entwickler überlassen, ob er für Einrückungen jeweils 1 Tabulatorzeichen oder 8 Leerzeichen verwendet. Dies muss jedoch im kompletten Codetext einheitlich sein, da ansonsten ein Fehler auftritt ( `IndentationError: expected an indented block` )

# Inhaltsverzeichnis

① Über Python

② Interpreter  
Aufgaben

③ Bibliographie

# Hinweise zur Darstellung I - Interaktiver Interpreter

Im Folgenden werden zu den jeweiligen Themen beispielhafte Codeausschnitte (Snippets) gezeigt, um eine Umsetzung in der Praxis zu erleichtern.

Folgende Darstellung wird genutzt, um eine Verwendung des Python-Codes innerhalb des interaktiven Python-Interpreters zu kennzeichnen:

```
>>> x = 1
>>> print('Hello world')
Hello world
```

Zeilen beginnend mit `>>>` stellen Eingaben / Befehle dar, während Zeilen ohne führende Größer-Zeichen die entsprechenden Ausgaben wiedergeben.

# Hinweise zur Darstellung II - Ausführbare Snippets

Python-Snippets, die nicht im interaktiven Interpreter ausgeführt werden, sondern beispielsweise in einer Python-Datei enthalten sind, werden ähnlich dargestellt:

```
import re

txt = 'abc, def, ghi'

x=re.split(', *', txt)
for y in x:
    print(y)
```

# Python Interpreter

Um Code in Python ausführen zu können, muss dieser an einen sogenannten "Interpreter" übergeben werden.

Dieser sorgt dafür, dass der Code von der Python-Umgebung verarbeitet und ausgeführt wird. Der Interpreter ist ein Programm, das der Nutzer auf seinem Rechner installieren muss, um Python-Programme ausführen zu können.

Ein aktueller Python-Interpreter kann unter <https://www.python.org/downloads/> heruntergeladen werden. Empfohlen wird mindestens Version 3.7 oder aktuellere Versionen (3.8 / 3.9)

# Python Interpreter: Windows (I)

## Installation:

- **Option 1:** Installation via **Windows-Store** (ab Windows 10)
- **Option 2:** Installation über <https://www.python.org/downloads/>

Zur Installation des Interpreters muss den Anweisungen des Installationsprogramms gefolgt werden. Es sollte darauf geachtet werden, dass die Python-Executable zur Pfad-Variable hinzugefügt wird:



# Python Interpreter: Windows (II)

## Ausführen des Python-Interpreters:

Der Python-Interpreter kann auf unterschiedliche Arten geöffnet werden:

- Über das Startmenü → Python3.x
- Über die Kommandozeile:
  - [WIN + R] drücken (oder nach 'Eingabeaufforderung' suchen)
  - `cmd` eingeben und bestätigen
  - `python` eingeben

# Python Interpreter: Unix (macOS & Linux)

## Installation:

Auf den meisten Unix-Systemen ist bereits eine Python Installation vorhanden. Jedoch handelt es sich dabei i.d.R. um den veralteten Python 2-Interpreter, der nicht mehr unterstützt wird. Daher sollte mind. Python 3.7 installiert werden.

Zur Installation des Interpreters muss den Anweisungen des Installationsprogramms gefolgt werden.

Der Python-Interpreter wird bei der Installation meist automatisch zur Pfad-Variable hinzugefügt.

## Ausführen des Python-Interpreters:

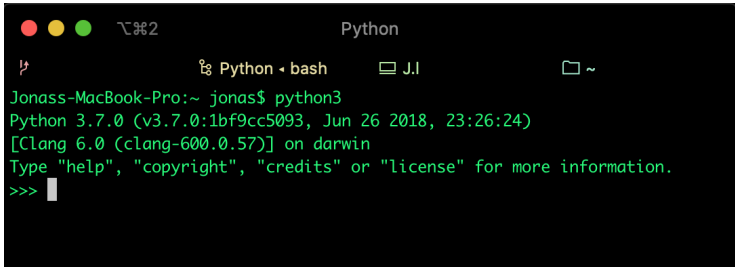
Der Python-Interpreter kann auf folgende Weise geöffnet werden:

- macOS: Programme → Dienstprogramme → Terminal → `python3` eingeben
- Linux: Terminal öffnen → `python3` eingeben



# Python Interpreter

Nachdem der Interpreter geöffnet wurde, sollte folgende (ähnliche) Ausgabe erscheinen:



```
Jonass-MacBook-Pro:~ jonas$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Abbildung 1: Der Python-Interpreter auf macOS

# Python Interpreter

Eine Möglichkeit, Python-Code auszuführen, ist, den Code direkt in die Interpreter-Umgebung einzugeben. Hier wird der Code umgehend ausgeführt und die Ausgabe dem Nutzer angezeigt.

Dass sich der Nutzer im Interpreter befindet, erkennt er daran, dass die Eingabezeile mit `>>>` beginnt.

Diese Umgebung wird auch als **REPL** (Read-eval-print loop) bezeichnet, da hier kontinuierlich Befehle vom Nutzer eingelesen, verarbeitet und die Ergebnisse wieder ausgegeben werden.

Eine weitere Möglichkeit, den Code auszuführen, besteht darin, dem Python-Interpreter den Pfad zu einer Python-Datei zu übergeben. Diese Methode wird später vorgestellt.

# Schließen des Interpreters

Um den Interpreter-Modus zu beenden und in den normalen Kommandozeilenmodus zurückzukehren (bzw. das Fenster zu schließen, falls der Interpreter nicht über die Kommandozeile geöffnet wurde), gibt es zwei Befehle, die genutzt werden können:

```
>>> exit()
```

und

```
>>> quit()
```

# Aufgaben

- 1 Öffnen Sie den Python-Interpreter. Überprüfen Sie, ob der Python-Interpreter ordnungsgemäß geöffnet wurde. Falls nicht, versuchen Sie das Problem zu lösen.
- 2 Öffnen Sie den Python-Interpreter und schließen Sie diesen wieder, indem Sie einen Befehl eingeben.

# Inhaltsverzeichnis

① Über Python

② Interpreter  
Aufgaben

③ Bibliographie

# Bibliographie I



Guido van Rossum. *Computer Programming for Everybody*. 1999. URL:  
<https://www.python.org/doc/essays/everybody/>.