Wirtschaftsinformatik 2

Einführung in Python

Prof. Dr. Christian Bockermann, Prof. Dr. Volker Klingspor

Hochschule Bochum

Sommersemester 2026

Dieser Foliensatz ist eine Einführung in die Programmiersprache Python für Studierende der Studiengänge BWL und IBM an der Hochschule Bochum. Python ist eine leicht erlernbare Programmiersprache, die sich gut für die Verarbeitung und Analyse von Daten eignet und in diesem Bereich einer großen Beliebtheit erfreut.
Python lässt sich in verschiedenen Varianten mit Hilfe von Entwicklungsumgebungen nutzen, aber auch im Browser über sogenannte Jupyter Notebooks (vgl. 1. Vorlesung). Sämtliche Code-Beispiele in diesem Foliensatz können (und sollen) direkt in Jupyter Notebook ausprobiert werden.

Python ist eine interpretierte Skript-Sprache

- Programme in Text-Dateien (Skripte) mit Endung `.py`
- Skripte werden zeilenweise abgearbeitet

Beispiel: Datei `HelloWord.py`

```
print("Hallo, Welt!")
```

Python ist eine interpretierte Skript-Sprache

- Programme in Text-Dateien (Skripte) mit Endung `.py`
- Skripte werden zeilenweise abgearbeitet

Beispiel: Datei `HelloWord.py`

```
print("Hallo, Welt!")
```

Starten durch Aufruf des Interpreters mit dem Skript:

```
# python3 HelloWorld.py
```

Python in Jupyter Notebooks

Eine einfache Alternative zum Programmieren mit Text-Dateien bieten die sogenannten Jupyter Notebooks. Dazu können Sie auf der Seite

`https://jupyter.hs-bochum.de`

nutzen, über die Sie eigene Notebooks erstellen können.

Für den Zugriff auf den Jupyter Server benötigen Sie Ihren HSBO Account (Benutzername und Passwort).

Hinweis: Das Begleitbuch Einfach Python gibt eine Einführung in Python mit der Programmierung über Programmierumgebungen und die Ausführung z.B. über die Kommandozeile. Dies ist für diese Vorlesung nicht erforderlich: **In der Vorlesung Wirtschaftsinformatik 2 werden wir nur die Programmierung über Jupyter Notebooks behandeln.**

Python Skripte sind Folgen von Ausdrücken

- Variablenzuweisung, Funktionsaufrufe
- Funktions- oder Klassendefinition
- Kontrollstrukturen (**`if`**, **`for`**,...)

Beispiel:

```
a = 42
b = 21
c = a + b
xs = [0, 1, 2, 3]  # eine Liste
summe = sum(xs)    # Funktionsaufruf
```

Python erlaubt Kommentare im Code

```
# Pythagoras aus der Schule
a = 3.0
b = 4.0
c2 = a*a + b*b     # c2 = c zum Quadrat
# wie berechnen wir die Wurzel aus c2?
```

Python - Kommentare

Python erlaubt Kommentare im Code

```
# Pythagoras aus der Schule
a = 3.0
b = 4.0
c2 = a*a + b*b     # c2 = c zum Quadrat
# wie berechnen wir die Wurzel aus c2?
```

- Kommentare starten mit Raute (#)
- Kann an beliebiger Stelle starten
- Kommentare sind wichtig um Code zu verstehen

Python ist eine dynamisch typisierte Sprache, d.h.

- Variablen haben einen Typ
- der Typ einer Variablen wird von Python zur Laufzeit ermittelt
- Typ wird nicht explizit vom Benutzer festgelegt

Beispiel:

```
a = 42          # Variable a ist ein  int
b = 42.0        # b ist vom Typ  float
c = a * b       # Welchen Typ hat c?
xs = [a, b]     # xs ist vom Type  list
```

Die Funktion `type(x)` gibt den Typ von `x` zurück.

Probieren Sie es im Notebook aus!

Text	`str`
Numerische Werte	`int`, `float`, `complex`
Sequenzen	`list`, `tuple`, `range`
Maps	`dict`
Mengen	`set`, `frozenset`
Bool'sche Werte	`bool`
Binäre Daten	`bytes`, `bytearray`, `memoryview`

Abbildung: Datentypen in Python (Auszug)

Python bietet Operatoren für Grundrechenarten:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division (Gleitkomma)
%	Modulo Operator (Rest)
//	Division (ganzzahlig)
**	Potenzieren

Spaß mit Variablen:

```
a = 3.0
b = 4.0
c2 = a**2 + b**2

c = c2 ** 0.5    # Wurzelziehen => hoch 0.5
```

Weitere Beispiele:

```
m = 8 % 3
x = 8 / 3
y = 8 // 3
```

Welche Werte ergeben sich für `m`, `x` und `y`?

Probieren Sie es im Notebook aus!

Relationale Operatoren für den Vergleich von Werten:

>	größer als
<	kleiner als
==	gleich
!=	ungleich
>=	größer oder gleich
<=	kleiner oder gleich

Vergleichsoperatoren liefern einen ein Wahrheitswert (`bool`)

```
erg = 1 < 3
print(erg)
```

Vergleich von Variablen:

```
a = 3.0
b = 4.0
x = a >= b
print(x)
```

Probieren Sie es im Notebook aus!

Vergleichsoperatoren können mit **and**, **or** und **not** benutzt werden:

```
a = 3.0
print( a > 2.0 and a < 5.0 )
```

Ablauf mit IF

Python stellt Kontrollstrukturen für Programmablauf bereit

- Bedingte Anweisungen mit **`if`**, **`else`**
- Schleifen mit **`for`** oder **`while`** (später)
- Bedingte Anweisungen in Block (Einrückung!)

Beispiel:

```
x = 30
if x > 10:
  print("x ist mehr als 10!")
else:
  print("x ist weniger als 10!")
```

Bedingte Anweisungen ermöglichen es, das Teile eines Programms nur ausgeführt werden, wenn bestimmte Vorbedingungen erfüllt sind.

Python verwendet Einrückungen um Code zu strukturieren

- Aufeinanderfolgende Zeilen mit Tiefe bilden Block
- Im Pythonumfeld werden die Blöcke Suites genannt
- Dadurch weniger Klammern

```
if x > 4:
  print("Mehr als 4")
  x = x + 4
else:
  print("Weniger als 5")
```

Python verwendet Einrückungen um Code zu strukturieren

- Aufeinanderfolgende Zeilen mit Tiefe bilden Block
- Im Pythonumfeld werden die Blöcke Suites genannt
- Dadurch weniger Klammern

```
if x > 4:
    print("Mehr als 4")
    x = x + 4
else:
    print("Weniger als 5")
```

Zusammenhängender Block

Beispiel: Bedingte Ausführung

```
wert = 42
ergebnis = 0

if wert > 38:
  ergebnis = wert * 1.20
else:
  ergebnis = wert * 0.8
ergebnis = ergebnis + 10
```

Welchen Wert hat die Variable **ergebnis**?

◀ Probieren Sie es im Notebook aus!