Python Crashkurs – Part I

Mentoring – WiSe 25/26

Patricia Gerbig
Freie Universität Berlin
Fachbereich Mathematik und Informatik

25. September 2025

Webseite

Warum Python?

- Leicht erlernbar.
- Kürzere Entwicklungsdauer im Vergleich zu anderen Programmiersprachen.
- Es gibt viele mächtige Bibliotheken für Python:
  - NumPy
  - Matplotlib
  - SciPy
  - TensorFlow
  - Biopython
  - ...
- Es ist kostenlos im Vergleich zu Matlab!
- (Indizierung beginnt ab 0!)

Was benötigt ihr?

- python3
- Terminal
- numpy
- matplotlib
- Texteditor (Empfehlung: VSCode)

Inhalt

Inhalt

Ausführen einer Python-Datei

1. Speichern des Skripts mit der Endung .py
2. Ins Verzeichnis der Python-Datei gehen.
3. Ausführen der Python-Datei mit

```
$ python3 dateiname.py
```

Interaktive Modus

Starten des interaktiven Modus mit

```
$ python3
Python 3.9.12 (main, Mar 26 2022, 15:44:31)
Type "help", "copyright", "credits" or "license" for more
    information.
>>>
```

Ausführen von Funktionen und Operationen z.B.

```
>>> 5 + 9
14
```

Importieren von Dateien (im selben Verzeichnis) und Bibliotheken

```
>>> import dateiname
```

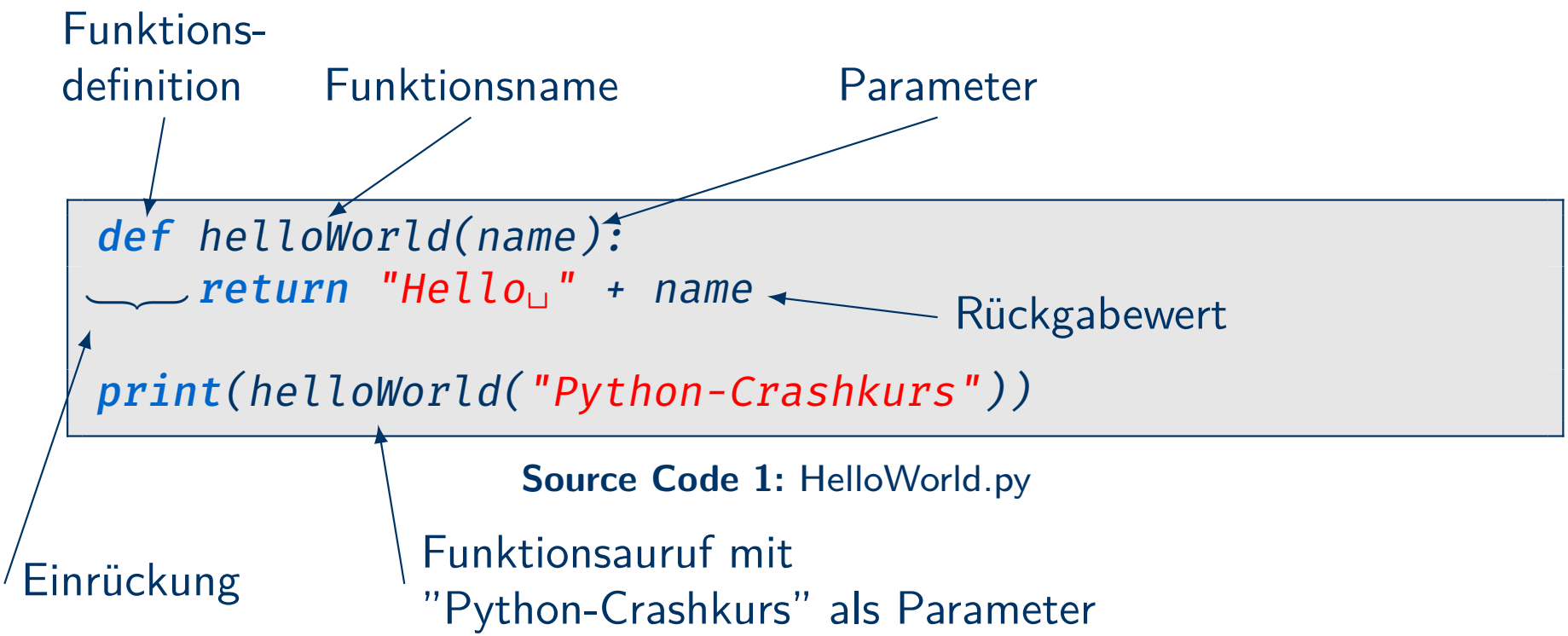
Verlassen des interaktiven Modus mit

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

Inhalt

Funktionen

Code in Funktion auslagern:



Source Code 1: HelloWorld.py

Was ist eine Funktion?

- Zusammenfassung von Anweisungen
- beliebig oft aufrufbar
- Parameter und Rückgabewert können auch leer sein, oder man kann mehrere haben

Beispiel 1: $f : \mathbb{R} \to \mathbb{R}$; $f(x) = x^2$ als Python-Funktion:

```
1       def f(x):
2           return x**2 # = x
        ^2
3
```

Beispiel 2: Hello World

```
1  def helloWorld():
2      print("Hello World")
3      return
4  helloWorld() #so wird die
       Funktion aufgerufen
5
```

Beispiel 3: $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$; $g(x, y) = x \cdot y$

```
1  def g(x,y):
2      return x*y
```

Aufgabe 1

1. Schreibe selber eine Funktion die "Hello World" ausgibt.
2. Rufe diese in deinem Skript auf und führe die Datei aus.
3. Rufe die Funktion im interaktiven Modus auf, indem du
   - 3.1 Den interaktiven Modus im richtigen Verzeichnis öffnest.
   - 3.2 Die Datei importierst.
   - 3.3 Deine Funktion aufrufst.
   - 3.4 Vergesse nicht den interaktiven Modus wieder zu verlassen.

Inhalt

Variablen & Datentypen

- Eine Variable besteht aus einen **Variablennamen**, einem **Datentyp** und ggf. einen **Startwert**
- Es gibt viele verschiedenen Datentypen. Hier ein paar Beispiele
  - Boolean
  - Integer
  - Float
  - complex
  - String
  - List
  - Mengen
  - Arrays (Numpy-Arrays)
  - ...

Boolean

- Kann zwei Zustände annehmen: **True** und **False**
- Beispiel:

```
1 bool1 = True
2 bool2 = False
```

Integer

- ganzzahliger Wert
- kann bis zu einer bestimmten Größe gespreichert werden: $[-2147483648, 2147483647]$ (Wieso erfahrt ihr in TI2)

```
1  int1 = 1
2  int2 = -50
3  int3 = 1397
```

floats

- Variablen vom Datentyp Float sind Kommazahlen, oder eher Gleitkommazahlen
- Gleitkommazahlen sind (grob gesagt) eine annährende Darstellung einer reellen Zahl (s.h. CoMa I)
- größte Float: 1.7976931348623157e+308
- kleinste Float: 2.2250738585072014e-308

```
1  float1 = 67.2
2  float2 = 3.14159265359
3  float2 = 2.0
4  float3 = 1.72e+3 #=1720.0
5  float4 = 7e-4 #=0.0007
```

String

- String ist eine Zeichenkette
- können in einzelne oder doppelte Anführungszeichen gesetzt werden

```
1 string1 = 's'
2 string2 = "Hello World"
3 string3 = 'Hello World'
4 string4 = '42'
5 string5 = '1.7692'
6 string6 = 'True'
7 string7 = '[1, 2, 3, 4]'
```

List

- Eine Liste (engl. List) ist eine **geordnete** Sammlung von Elementen
- Elemente können verschiedene Datentypen sein
- Elemente dürfen sich **doppeln**
- Listen sind **dynamisch** d.h. Elemente können verändert, hinzugefügt und gelöscht werden.
- Jedes Element hat einen **Index**. Dieser beginnt bei 0

```
1 list1 = [1,2,3,4]
2 list2 = [3.7, 2.9, 5.3]
3 list3 = [True, False]
4 list4 = [True, 1, 3.4, '
      Hallo']
5
```

Aufgabe 2

Bestimme die Datentypen folgender Variablen:

```
variable1 = 57
variable2 = 3.7
variable3 = 10.0
variable4 = 3 / 2
variable5 = 57 // 5
variable6 = '42'
variable7 = 'Python'
variable8= [1, 2, 3]
variable9 = ['Hallo', 'Welt']
variable10 = variable9[0]
variable11 = not False
```

Tipp: Zum überprüfen: Recherchiert mal was die **type()-Funktion** tut.

Inhalt

Operatoren

- Operatoren sind Funktionen, mit einer anderen Notation
- Zwei Kategorien:
  - **unäre Operatoren** (ein Operand)
  - **binäre Operatoren** (zwei Operanden)
- drei Schreibweisen von Operatoren:
  - **Prefix** - Operator wird am Anfang geschrieben
  - **Infix** - Operator wird in der Mitte geschrieben
  - **Postfix** - Operator wird am Ende geschrieben
- In Python sind unäre Operatoren meistens Prefix und binäre Operatoren Infix

Vergleichsoperatoren

- Vergleichsoperatoren sind grundsätzlich binär
- Rückgabe ist vom Datentyp Boolean

```
4 == 4.0 #True
3.14 != 3.14159 #True
4 < 3 #False
'Hello' => 'Hi' #True
```

Vergleichsoperatoren (binär)		
==	Gleichheit	Bool, Int, Float, List, Str
!=	Ungleichheit	Bool, Int, Float, List, Str
<	kleiner als	Int, Float, List, Str
>	größer als	Int, Float, List, Str
<=	kleiner gleich	Int, Float, List, Str
>=	größer gleich	Int, Float, List, Str

logische Operatoren

- können unär und binär sein
- Ruckgabe ist von Datentyp Boolean
- können praktisch auch auf Integer, Floats, Strings und Lists angewendet werden. (d.h. es gibt keine Fehlermeldung, Sinn ergeben tut es z.B. bei Integer aber nur im Binärsystem)

logische Operatoren (unär)		
not	log. Negation	Boolean
logische Operatoren (binär)		
and	logisches und	Boolean
or	logisches oder	Boolean

```
1  not True #False
2  True and False #False
3  True or False #True
4  0 and 1 #0
5  0 or 1 #1
6  1 and not 0 #True
```

arithmetische Operatoren

arithmetische Operatoren (unär)		
+	ändert nichts	Int, Float
-	dreht das VZ um	Int, Float
arithmetische Operatoren (binär)		
+	Addition	Int, Float
-	Subtraktion	Int, Float
*	Multiplikation	Int, Float
**	Power	Int, Float
//	ganzzahl Division	Int, Float
%	Modulo	Int
/	Division	Float

- Rückgabewert ist vom Datentyp Integer oder Float
- praktisch können alle Operanden Integer und Float sein

```
1  67 + 5 #72
2  3.14 - 3 #0.14
3  2.0 * 3 #6.0
4  4**2 #16
5  37 // 5 #7
6  14 % 3 #2
7  24 / 5 #4.8
```

Modulo

- gibt den Rest der ganzzahligen Division zurück

$$10 \text{ mod } 4 = 2, \text{ da } 10//4 = 2 \quad (\text{Rest } 2)$$
$$53 \text{ mod } 10 = 3, \text{ da } 53//10 = 5 \quad (\text{Rest } 3)$$
$$24 \text{ mod } 2 = 0, \text{ da } 24//2 = 12 \quad (\text{Rest } 0)$$

- kann auch mit Floats verwendet werden
  Vorsicht: Rundungsfehler können zu falschen Ergebnissen führen!

Listenoperatoren

- Bei Anwendung auf Strings: Strings als Liste von Zeichen vorstellen
- **Achtung:** erste Index ist 0

Listenoperatoren (unär)		
`[i]`	i-te Stelle	List, String
Listenoperatoren (binär)		
`+`	Verkettung	List, String
`*`	Wiederholung	List, String
`[i:j]`	Teilliste	List, String
`in`	enthalten	List, String

```
1  x = [1,2,3,4,5]
2  x[2] #3
3  y = x[1:3] #[2,3]
4  2*y #[2,3,2,3]
5  6 in x #False
6  z = 'Hello'
7  'o' in z #True
8  z[0] #'H'
```

Funktionen & Methoden für Listen	
`len(list)`	gibt die Länge aus
`max(list)`	findet das Maximum
`min(list)`	findet das Minimum
`sorted(list)`	gibt die sortierten Elemente aus
`print(list)`	gibt die Liste aus
`list.append(x)`	fügt hinten an
`list.pop()`	löscht letztes Element
`list.insert(i,x)`	Fügt x an Pos. i ein.
`list.remove(x)`	löscht das Elem. x
`list.index(x)`	Gibt den Index des 1. Matches aus
`list.copy()`	Kopiert die Liste

Quelle: (o.A., 2019)

Aufgabe 3

- Welche verschiedenen Ideen fallen dir ein, wie du ein Programm umsetzen kannst, dass 4mal Hello World ausgibt.
- Wie könntest du eine Funktionen gestalten, die 2 Strings übergeben bekommt und diese aneinander kettet?

Inhalt

Fallunterscheidung


```
1  def unterscheidung(n):
2      if n == 0:
3          return 0
4      elif n > 0:
5          return 1
6      else:
7          return -1
```

Die **if-else-Anweisung** prüft ein Prädikat und führt zu einer Entscheidung, die den Programmablauf beeinflusst.

Mithilfe von **elif** kann zwischen mehr als nur 2 Fällen unterschieden werden.

Die **else-Anweisung** ist optional. Wird sie weg gelassen, wird am Ende der Fallunterscheidung fortgefahren.

Fallunterscheidungen können auch **geschachtelt** werden.

Aufgabe 4

Das Ziel ist es, ein Konzept für eine Funktion zu erstellen, welcher du eine Jahreszahl übergibst und die Funktion sagt dir ob es ein Schaltjahr ist oder nicht. Es gilt:

- Ist die Jahreszahl durch vier teilbar, aber nicht durch 100, ist es ein Schaltjahr.
- Ist die Jahreszahl durch 100 teilbar, aber nicht durch 400, ist es kein Schaltjahr.
- Ist die Jahreszahl durch 400 teilbar, dann ist es ein Schaltjahr.

Jeder verwendete Operator gibt einen Punkt.
Überlege dir mehrere (mind. 3) verschiedene Methoden mit jeweils unterschiedlichen Punkten.

Inhalt

Schleifen

for-Schleife:

```
1  def sumAll(n):
2      sum = n
3      for i in range(n):
4          sum = sum+i
5          print(i)
6      return sum
```

for-Schleifen durchlaufen die Elemente der ihr übergebenen Liste. Der Zähler i wird in jedem Schleifendurchlauf verändert.
Besonderheit: Die Anzahl der Schleifendurchläufe steht zu Anfang fest.

while-Schleife:

```
1  def sumAll(n):
2      i, sum = 1, 0
3      while i <= n:
4          sum = sum+i
5          i = i+1
6      return sum
```

Solange die Bedingung erfüllt ist, führen **while-Schleifen** den Schleifenrumpf aus. Der Zähler *i* muss manuell verändert werden, damit die Schleife terminiert.
Besonderheit: Die Anzahl der Schleifendurchläufe braucht nicht festzustehen.

Aufgabe 5

- Recherchiere, wie man in Python über eine Liste durchlaufen kann
- Recherchiere, was die Funktionen len() und range() tun.
- Überlege dir zwei verschiedene Funktion die eine Liste übergeben bekommt und mithilfe einer **for-Schleife** jedes Listenelement einzeln ausgegeben bekommt.
- Überlege dir eine Funktion die eine Liste übergeben bekommt und mithilfe einer **while-Schleife** jedes Listenelement einzeln ausgegeben bekommt.
- Überlege dir eine Funktion, die eine natürliche Zahl übergeben bekommt. Für jede Zahl zwischen 1 und dieser Zahl die durch 3 Teilbar ist, soll Tac ausgegeben werden und wenn diese auch durch 9 Teilbar ist Tic Tac. Wenn nichts davon zutrifft, soll die Zahl augegeben werden.
  Konzipiere dir eine zweite Funktion, mit dem gleichen Ergebnis, aber unterschiedlichen If und Else abfragen.

Inhalt

Iteration & Rekursion

Iteration:

```
1  def  fakultaet(n):
2      res = 1
3      for i in  range(1,n
       +1):
4          res = res*i
5      return  res
```

Wiederholungen werden durch Schleifen realisiert.

Rekursion:

```
1  def fakultaet(n):
2      if n == 0:
3          return 1
4      else:
5          return n*
       fak_rekursiv(n-1)
```

die Funktion ruft sich immer wieder selbst auf, bis eine Abbruchbedingung eintritt.

Noch Fragen?