



CORRELAID

GOOD CAUSES. BETTER EFFECTS.

Python Workshop

Termin 1: Python Basics

Wer wir sind



Wir sind ein deutschlandweites Netzwerk von über 2,000 Data Scientists, die die Welt durch Data Science verbessern wollen.

#MetaWeltretter



Data4Good Projekte



Der Kern unserer Arbeit ist die Durchführung von pro-bono Projekten mit Non-Profit-Organisationen.

Mehr als 55 Data4Good Projekte wurden seit 2015 umgesetzt

Logos auf Konstanzer Projekte upo

Einige unserer Kooperationspartner:innen:

project
together



CORRELAID
GOOD CAUSES. BETTER EFFECTS.

Mitmachen



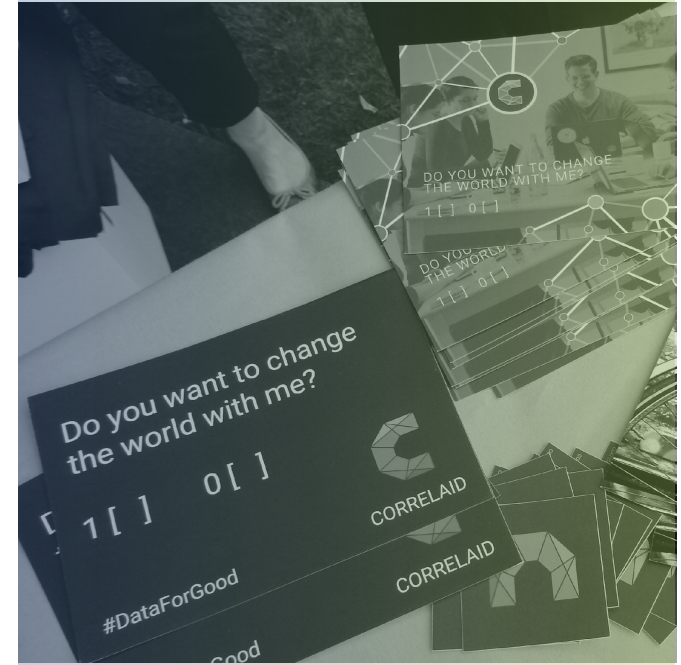
CorrelAid ist ein offenes Netzwerk für alle Menschen



Jede:r der oder die unseren Code of Conduct respektiert, ist willkommen



Neben der Projektarbeit kannst du auf unterschiedliche Art und Weise bei uns aktiv werden



Agenda

1. Intro
2. Absolute Basics
3. Datentypen
4. Datencontainer
5. Loops und Conditional Statements
6. Funktionen



1. Intro - Was ist Python?

- Allzweck-Programmiersprache
- Entwickelt von Guido van Rossum in den 1980er Jahren
- Name leitet sich von "Monty Python's Flying Circus" ab
- Aufstieg unter anderem dank Google (Machine Learning Libraries)



1. Intro - Warum Python?

Flexibel
&
Einfach



Große Community



Windows
MacOS
Linux



Viele
Libraries
(Erweiterungen)



Flexible in
Programmier-
paradigmen



Kostenlos!

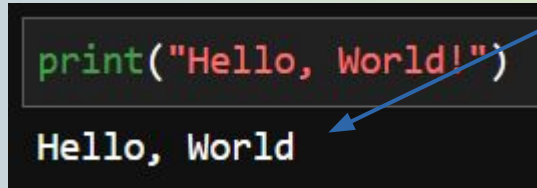


2. Absolute Basics

- Hello, World!
- Variablen
- Mathematische Operatoren
- Logische Operatoren
- Kommentare



2. Absolute Basics - Hello, World!



```
print("Hello, World!")  
Hello, World
```

- Input
- Output
- () → Funktion wird auf Inhalt der Klammer ausgeführt
- print() → Output in die Konsole
- "" oder " → zum Schreiben von Ausdrücken (später mehr)



2. Absolute Basics - Variablen

```
tutor1 = "Torben"
tutor2 = "Jonas"
age_tutor1 = 22
age_tutor2 = 23

print(tutor1, age_tutor1)
print(f"{tutor2} ist {age_tutor2} alt.")

Torben 22
Jonas ist 23 alt.
```

- Man kann Code in Variablen speichern
- Jegliche Datentypen können zugewiesen werden
- Assignment Operator =
- Konvention: klein schreiben und _ als Leerzeichen



2. Absolute Basics - Mathematische Operatoren

```
# Addition
sum_numbers = 7 + 8 + 15
print("sum_numbers:", sum_numbers)

# Subtraktion
number1, number2, = 25, 17
diff_numbers = number1 - number2
print("diff_numbers:", diff_numbers)

sum_numbers: 30
diff_numbers: 8
```

- Alle mathematischen Operatoren können auch in Python genutzt werden
- Bei numerischen Ausdrücken funktionieren diese immer
- Bei anderen Datentypen kommt es darauf an, ob diese "sinnvoll" sind (Achtung, kann zu anderem Ergebnis führen!)



2. Absolute Basics - Mathematische Operatoren

Übersicht mathematischer Operatoren

Syntax	Math	Operation Name
<code>a+b</code>	$a + b$	addition
<code>a-b</code>	$a - b$	subtraction
<code>a*b</code>	$a \times b$	multiplication
<code>a/b</code>	$a \div b$	division (see note below)
<code>a//b</code>	$\lfloor a \div b \rfloor$	floor division (e.g. $5//2=2$) - Available in Python 2.2 and later
<code>a%b</code>	$a \bmod b$	modulo
<code>-a</code>	$-a$	negation
<code>abs(a)</code>	$ a $	absolute value
<code>a**b</code>	a^b	exponent
<code>math.sqrt(a)</code>	\sqrt{a}	square root



2. Absolute Basics - Mathematische Operatoren

Diese mathematischen Operatoren können direkt in die Assignment Operators eingebaut werden.

`+=` (increment assignment)

Adds a value and the variable and assigns the result to that variable.

`-=` (decrement assignment)

Subtracts a value from the variable and assigns the result to that variable.

`*=` (multiplication assignment)

Multiplies the variable by a value and assigns the result to that variable.

`/=` (division assignment)

Divides the variable by a value and assigns the result to that variable.

`=` (power assignment)**

Raises the variable to a specified power and assigns the result to the variable.

`%=` (modulus assignment)

Computes the modulus of the variable and a value and assigns the result to that variable.

`//=` (floor division assignment)

Floor divides the variable by a value and assigns the result to that variable.



2. Absolute Basics - Logische Operatoren

```
print(True and False)
print(True or False)
print(not False and True)
```

```
num0 = 0
print(num0 or True)
print(5 >= 7)
```

```
False
True
True
True
False
```

- Ergeben stets Wahrheitsausdrücke (True/False)
- Wichtig für Vergleiche und Bedingungen



2. Absolute Basics - Logische Operatoren

Übersicht Logische Operatoren

Operator	Description	Example
and	Returns True if both statements are true	<code>x < 5 and x < 10</code>
or	Returns True if one of the statements is true	<code>x < 5 or x < 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x < 5 and x < 10)</code>

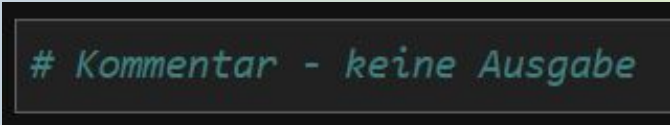


2. Absolute Basics - Logische Operatoren

Übersicht Vergleichsoperatoren

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

2. Absolute Basics - Kommentare



```
# Kommentar - keine Ausgabe
```

- # Kommentar
- Sollte unbedingt genutzt werden, um den Code verständlicher zu machen



3. Datentypen

- Strings
- Ints
- Floats
- Boolean



3. Datentypen - Strings

```
print("That is a string.")  
print('That is a string as well.')  
print("even numbers can be strings 1, 2, 3")
```

✓ 0.1s

```
That is a string.  
That is a string as well.  
even numbers can be strings 1, 2, 3
```

- Buchstaben- und Zeichenfolgen (können auch Zahlen enthalten)
- stehen in Anführungszeichen ("" oder ")
- Allerhand Methoden (siehe nächste Seiten)



4. Datentypen - Strings

Übersicht Methoden I

Method	Description
<u>capitalize()</u>	Converts the first character to upper case
<u>casefold()</u>	Converts string into lower case
<u>center()</u>	Returns a centered string
<u>count()</u>	Returns the number of times a specified value occurs in a string
<u>encode()</u>	Returns an encoded version of the string

4. Datentypen - Strings

Übersicht Methoden II

<u>endswith()</u>	Returns true if the string ends with the specified value
<u>expandtabs()</u>	Sets the tab size of the string
<u>find()</u>	Searches the string for a specified value and returns the position of where it was found
<u>format()</u>	Formats specified values in a string
<u>format_map()</u>	Formats specified values in a string
<u>index()</u>	Searches the string for a specified value and returns the position of where it was found

4. Datentypen - Strings

Übersicht Methoden III

<u>isalnum()</u>	Returns True if all characters in the string are alphanumeric
<u>isalpha()</u>	Returns True if all characters in the string are in the alphabet
<u>isdecimal()</u>	Returns True if all characters in the string are decimals
<u>isdigit()</u>	Returns True if all characters in the string are digits
<u>isidentifier()</u>	Returns True if the string is an identifier
<u>islower()</u>	Returns True if all characters in the string are lower case
<u>isnumeric()</u>	Returns True if all characters in the string are numeric



4. Datentypen - Strings

Übersicht Methoden IV

<u>isprintable()</u>	Returns True if all characters in the string are printable
<u>isspace()</u>	Returns True if all characters in the string are whitespaces
<u>istitle()</u>	Returns True if the string follows the rules of a title
<u>isupper()</u>	Returns True if all characters in the string are upper case
<u>join()</u>	Joins the elements of an iterable to the end of the string
<u>ljust()</u>	Returns a left justified version of the string
<u>lower()</u>	Converts a string into lower case



4. Datentypen - Strings

Übersicht Methoden V

<u>lstrip()</u>	Returns a left trim version of the string
<u>maketrans()</u>	Returns a translation table to be used in translations
<u>partition()</u>	Returns a tuple where the string is parted into three parts
<u>replace()</u>	Returns a string where a specified value is replaced with a specified value
<u>rfind()</u>	Searches the string for a specified value and returns the last position of where it was found
<u>rindex()</u>	Searches the string for a specified value and returns the last position of where it was found

4. Datentypen - Strings

Übersicht Methoden VI

<code>rjust()</code>	Returns a right justified version of the string
<code>rpartition()</code>	Returns a tuple where the string is parted into three parts
<code>rsplit()</code>	Splits the string at the specified separator, and returns a list
<code>rstrip()</code>	Returns a right trim version of the string
<code>split()</code>	Splits the string at the specified separator, and returns a list
<code>splitlines()</code>	Splits the string at line breaks and returns a list



4. Datentypen - Strings

Übersicht Methoden VII

<u>startswith()</u>	Returns true if the string starts with the specified value
<u>strip()</u>	Returns a trimmed version of the string
<u>swapcase()</u>	Swaps cases, lower case becomes upper case and vice versa
<u>title()</u>	Converts the first character of each word to upper case
<u>translate()</u>	Returns a translated string
<u>upper()</u>	Converts a string into upper case
<u>zfill()</u>	Fills the string with a specified number of 0 values at the beginning

3. Datentypen - Ints

```
number = 100
print(type(number))
print(100 + 201)
print(int(3.1))
```

✓ 0.4s

```
<class 'int'>
```

```
301
```

```
3
```

- Ints sind ganze Zahlen
- alle mathematischen und logischen Operatoren können genutzt werden
- wird eine Kommazahl zu einem Int, wird diese immer Richtung $-\infty$ gerundet



3. Datentypen - Floats

```
first_gini = gini[0]
print(first_gini)
first_gini_minus_a_tenth = first_gini - 0.1
print(first_gini_minus_a_tenth)
```

✓ 0.3s

30.4

30.299999999999997

- Floats sind Kommazahlen
- Es gilt zu beachten, dass beim Rechnen mit Floats Rundungsfehler unterlaufen können - das liegt an deren Binärdarstellung im Rechner



3. Datentypen - Booleans

```
print(first_gini == 30.4)  
print(first_gini == 30)
```

✓ 0.2s

False

True

- Wahrheitswerte
- Grundsätzlich nur True (1) oder False (0)



4. Datencontainer

- Listen
- Tuples
- Sets
- Dictionaries



4. Datencontainer - Listen

```
gini = [48.7, 48.7, 48.6, 48.6, 48.3, 47.9, 47.7, 47.6, 35.9]

# Slicing
gini_2_4 = gini[1:4]
print(gini_2_4)
gini_2_4 = gini[1:4:2]
print(gini_2_4)

# Hinzufügen eines weiteren Datenpunktes mit .append()
gini_2_4.append("Ungleichheit") # andere Datentyp
print(gini_2_4)

✓ 0.3s

[48.7, 48.6, 48.6]
[48.7, 48.6]
[48.7, 48.6, 'Ungleichheit']
```

- Datencontainer mit `[]`-Klammern, der sehr häufig genutzt wird
- geordnet/indiziert, veränderlich, erlaubt Duplikate, erlaubt verschiedene Datentypen
- allerhand Methoden (siehe nächste Seiten)



4. Datencontainer - Listen

Übersicht Methoden I

Method	Description
<u>append()</u>	Adds an element at the end of the list
<u>clear()</u>	Removes all the elements from the list
<u>copy()</u>	Returns a copy of the list
<u>count()</u>	Returns the number of elements with the specified value
<u>extend()</u>	Add the elements of a list (or any iterable), to the end of the current list



4. Datencontainer - Listen

Übersicht Methoden II

<u>index()</u>	Returns the index of the first element with the specified value
<u>insert()</u>	Adds an element at the specified position
<u>pop()</u>	Removes the element at the specified position
<u>remove()</u>	Removes the item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

4. Datencontainer - Tuples

```
tup_gini_50_58 = tup_gini[49:58]
print(tup_gini_50_58)
tup_gini.append("Ungleichheit") # funktioniert nicht
print(tup_gini)
```

⊗ 0.7s

(48.7, 48.7, 48.6, 48.6, 48.3, 47.9, 47.7, 47.6, 35.9)

AttributeError

Traceback (most recent call last):

- Datencontainer mit `()`-Klammern
- geordnet/indiziert, unveränderlich, erlaubt Duplikate, erlaubt verschiedene Datentypen



4. Datencontainer - Tuples

Übersicht Methoden

Method	Description
<u>count()</u>	Returns the number of times a specified value occurs in a tuple
<u>index()</u>	Searches the tuple for a specified value and returns the position of where it was found

4. Datencontainer - Sets

```
list_dup = [8, 8, 7, 1, 3, 3, 1]
ohne_dup = set(list_dup)
ohne_dup
```

✓ 0.3s

```
{1, 3, 7, 8}
```

- Datencontainer mit {}-Klammern, zum Spezifizieren set() nötig!
- ungeordnet/nicht indiziert, veränderlich, keine Duplikate, erlaubt verschiedene Datentypen
- wird gerne zum Entfernen von Duplikaten verwendet



4. Datencontainer - Sets

Übersicht Methoden I

Method	Description
<u>add()</u>	Adds an element to the set
<u>clear()</u>	Removes all the elements from the set
<u>copy()</u>	Returns a copy of the set
<u>difference()</u>	Returns a set containing the difference between two or more sets

4. Datencontainer - Sets

Übersicht Methoden II

<u><code>difference_update()</code></u>	Removes the items in this set that are also included in another, specified set
<u><code>discard()</code></u>	Remove the specified item
<u><code>intersection()</code></u>	Returns a set, that is the intersection of two other sets
<u><code>intersection_update()</code></u>	Removes the items in this set that are not present in other, specified set(s)
<u><code>isdisjoint()</code></u>	Returns whether two sets have a intersection or not
<u><code>issubset()</code></u>	Returns whether another set contains this set or not



4. Datencontainer - Sets

Übersicht Methoden III

<u>issuperset()</u>	Returns whether this set contains another set or not
<u>pop()</u>	Removes an element from the set
<u>remove()</u>	Removes the specified element
<u>symmetric_difference()</u>	Returns a set with the symmetric differences of two sets
<u>symmetric_difference_update()</u>	inserts the symmetric differences from this set and another
<u>union()</u>	Return a set containing the union of sets
<u>update()</u>	Update the set with the union of this set and others



4. Datencontainer - Dictionaries

```
# nur ein Infopunkt (gini)
country_info = {list_countries[0] : gini[0]}
print(country_info)

# zwei Infopunkte (years und gini)
country_info = {list_countries[0] : [years[0], gini[0]]}
print(country_info)
```

✓ 0.4s

```
{'Afghanistan': 30.4}
{'Afghanistan': [2007, 30.4]}
```

- Datencontainer mit {}-Klammern
- key:value-Paare. Jedem key ist ein value zugeordnet
- Keys dürfen nicht mehrfach vorkommen, sonst wird der erste überschrieben
- erlauben verschiedene Datentypen



4. Datencontainer - Dictionaries

Übersicht Methoden I

Method	Description
<u>clear</u> ()	Removes all the elements from the dictionary
<u>copy</u> ()	Returns a copy of the dictionary
<u>fromkeys</u> ()	Returns a dictionary with the specified keys and value
<u>get</u> ()	Returns the value of the specified key
<u>items</u> ()	Returns a list containing a tuple for each key value pair
<u>keys</u> ()	Returns a list containing the dictionary's keys



4. Datencontainer - Dictionaries

Übersicht Methoden II

<u>pop()</u>	Removes the element with the specified key
<u>popitem()</u>	Removes the last inserted key-value pair
<u>setdefault()</u>	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
<u>update()</u>	Updates the dictionary with the specified key-value pairs
<u>values()</u>	Returns a list of all the values in the dictionary

5. Loops und Conditional Statements

- for-Loops
- while-Loops
- if-Statement
- else/elif

5. Loops und Conditional Statements - for-Loops



5. Loops und Conditional Statements - while-Loops



5. Loops und Conditional Statements - if-Statement



5. Loops und Conditional Statements - else/elif



6. Funktionen

Weiterführende Quellen

Lehrbücher

- [A Beginners Guide to Python 3 Programming](#) (John Hunt)

YouTube

- [Corey Schafer](#)

Übungen

- [w3resource](#)



CORRELAID
GOOD CAUSES. BETTER EFFECTS.

Vielen Dank für eure Aufmerksamkeit!

...und bis zum nächsten Mal

Bei Fragen, Wünschen oder Feedback zum Kurs meldet euch gerne bei
jonas.2.hummel@uni-konstanz.de oder torben.abts@uni-konstanz.de