CORRELAID

GOOD CAUSES. BETTER EFFECTS.

Python Workshop

Termin 1: Python Basics

Torben Abts & Jonas Hummel

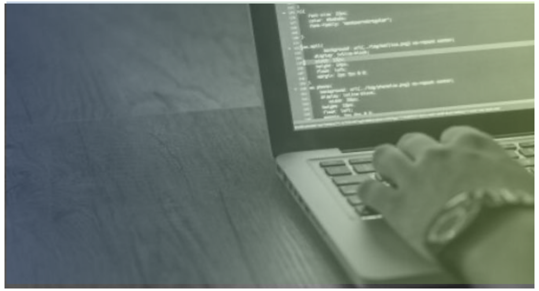
10.05.2022

Wer wir sind

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

#data4good

Unsere Mission

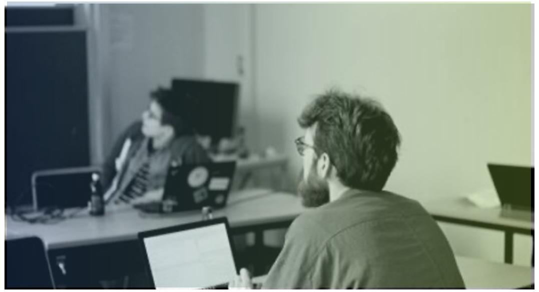


DATA4GOOD PROJEKTE

Wir führen pro-bono Datenanalyseprojekte für gemeinnützige Organisationen durch.

EDUCATION

Wir vernetzen engagierte sozial denkende Datenanalyst:innen und bieten ihnen Möglichkeiten ihr Wissen anzuwenden und zu erweitern. Außerdem vermitteln wir engagierten Menschen von gemeinnützigen Organisationen grundlegende Data Literacy Skills.



DIALOG

Wir treten in den Dialog über den Wert und Nutzen von Daten und Datenanalysen für das Gemeinwohl.

CorrelAid Projekte - Beispiele

Stolpersteine Konstanz

Erstellen eines strukturierten Datensatzes und einer Website für Stolpersteine Konstanz.

Hackathon Südkurier

Organisation und Durchführung eines Hackathons in Kooperation mit dem Südkurier am 11. Juni 2022.

SWR (Datenjournalismus)

Unterstützung bei der Datenanalyse für das Datenjournalismus-Team des SWR (vermutlich Evaluation des 9€-Tickets).

CitiesROpen

Erstellung eines R-Packages zum direkten Importieren von Daten der Stadt Konstanz (und möglicherweise für weitere Städte)

Mitmachen

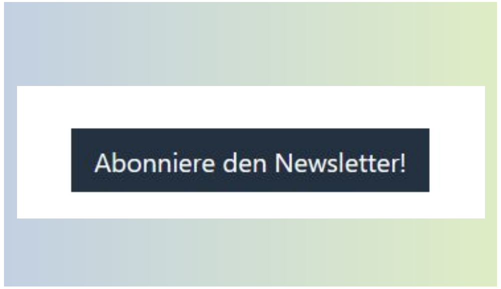
- CorrelAid ist ein offenes Netzwerk für alle Menschen
- Jede:r der oder die unseren Code of Conduct respektiert, ist willkommen
- Neben der Projektarbeit kannst du auf unterschiedliche Art und Weise bei uns aktiv werden

Do you want to change the world with me?
1 [ ]   0 [ ]
#DataForGood
CORRELAID
DO YOU WANT TO CHANGE THE WORLD WITH ME?

Mitmachen

SLACK

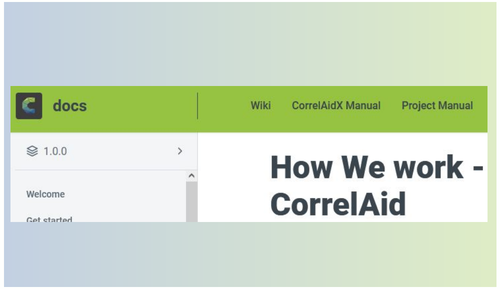
konstanz@correlaid.org



Abonniere den Newsletter!

NEWSLETTER

https://correlaid.org/volunteer/



How We work - CorrelAid

HITCHHIKER'S GUIDE

https://docs.correlaid.org/wiki/hitchhikers-guide

Agenda

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

* Die Unterscheidung zwischen Datentypen und Datencontainern ist konzeptionell nicht ganz richtig, aber zum Verständnis hilfreich.

https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Python-logo-notext.svg/1200px-Python-logo-notext.svg.png

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)

TensorFlow

https://camo.githubusercontent.com/aeb4f612bd9b40d81c62fcbebd6db44a5d4344b8b962be0138817e18c9c06963/68747470733a2f2f777777
2e74656e736f72666c6f772e6f72672f696d616765732f74665f6c6f676f5f686f72697a6f6e74616c2e706e67

1. Intro - Warum Python?

Flexibel
&
Einfach

Große Community

Windows
MacOS
Linux

Viele
Libraries
(Erweiterungen)

Flexible in
Programmier-
paradigmen

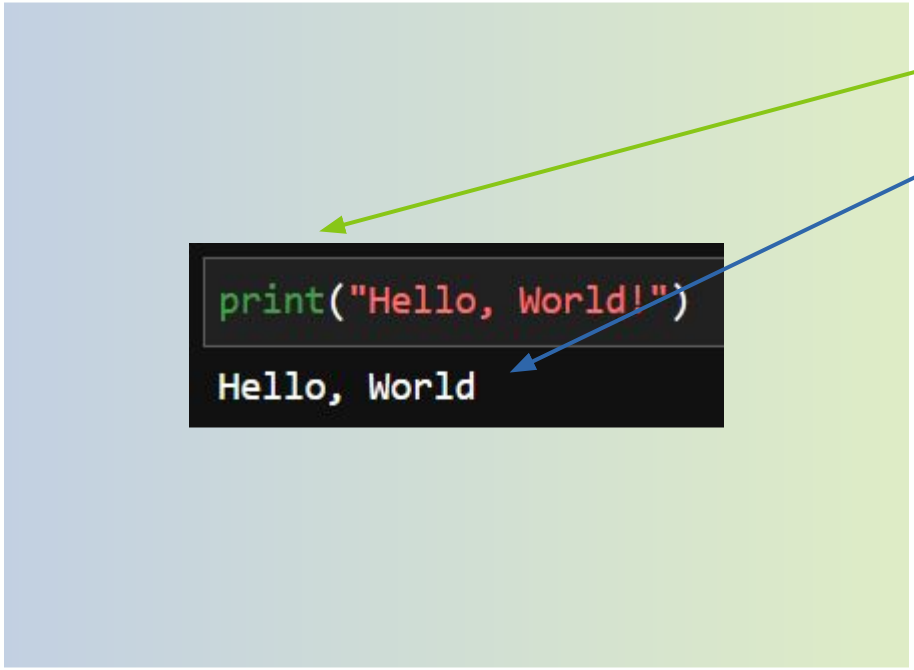
Kostenlos!

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

2. Absolute Basics

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

2. Absolute Basics - Hello, World!

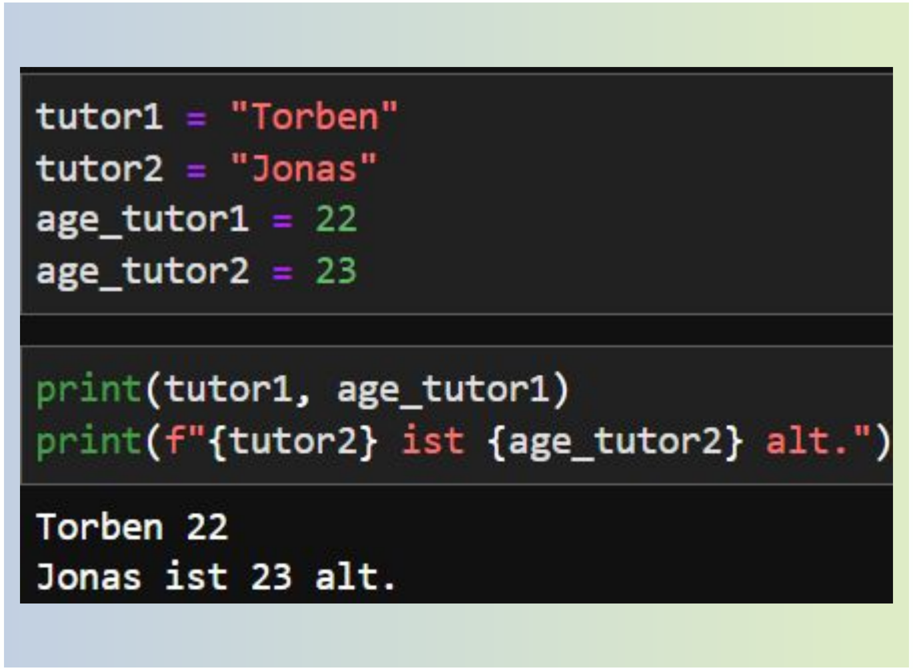


```
print("Hello, World!")
Hello, World
```

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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

2. Absolute Basics - Variablen



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

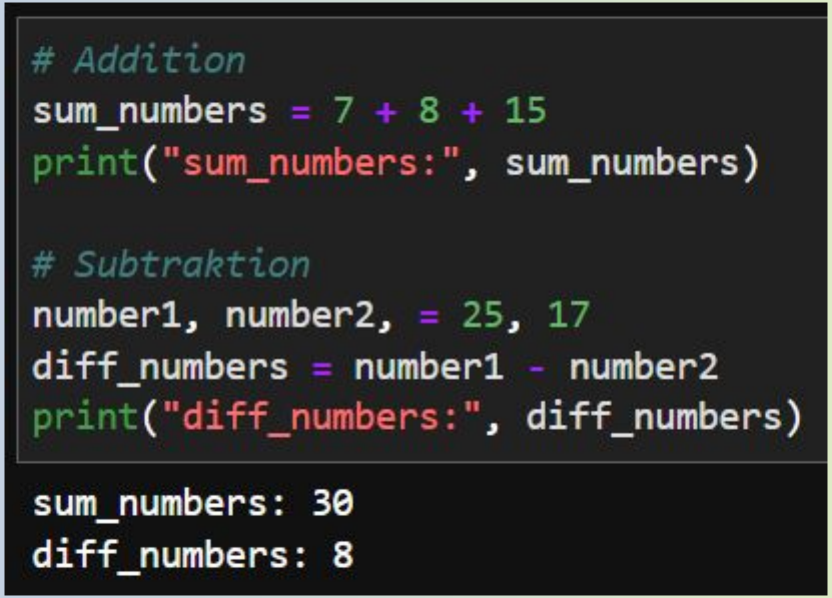
```
print(tutor1, age_tutor1)
print(f"{tutor2} ist {age_tutor2} alt.")
```

```
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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

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)
```

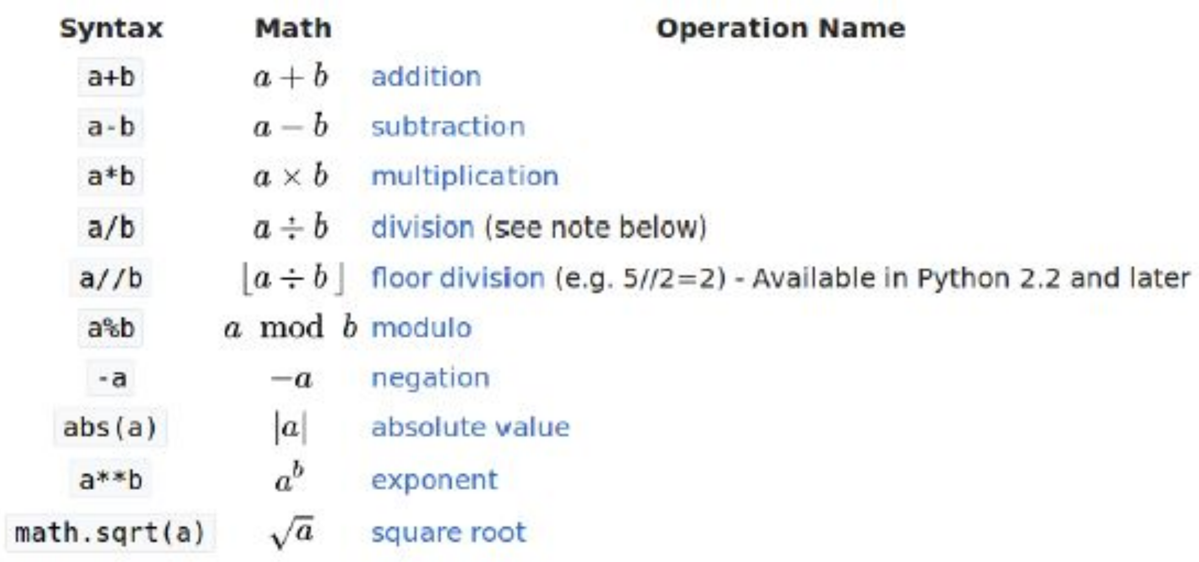
```
sum_numbers: 30
diff_numbers: 8
```

- 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!)

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

2. Absolute Basics - Mathematische Operatoren

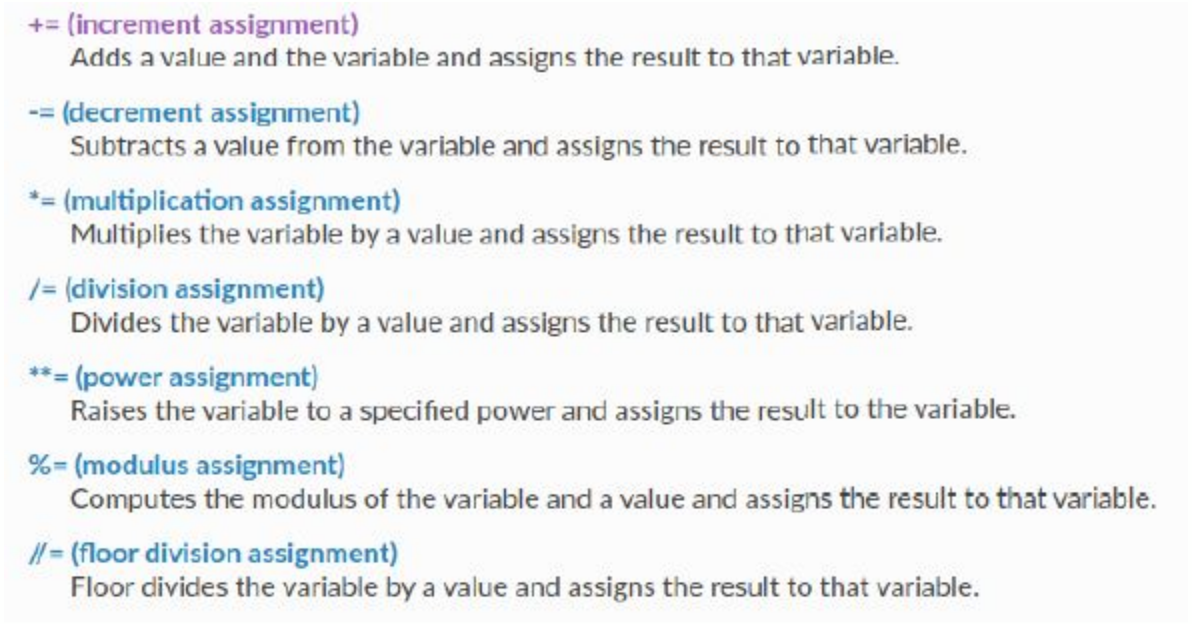
Übersicht mathematischer Operatoren



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

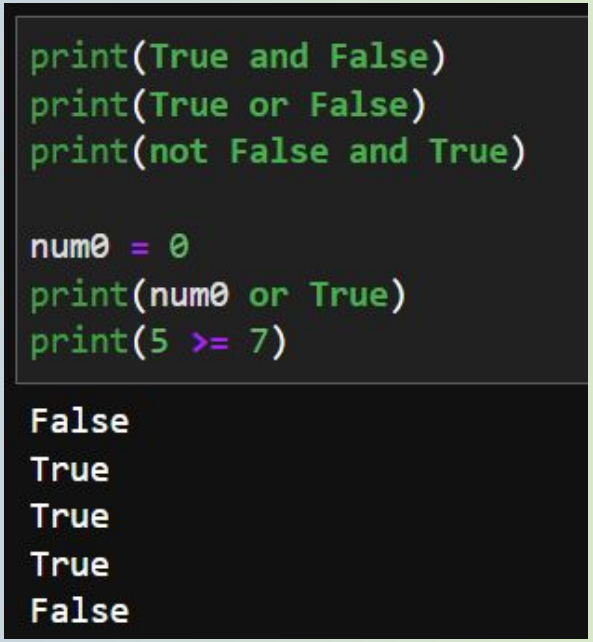
%= (modulus assignment)
Computes the modulus of the variable and a value and assigns the result to that variable.

//= (floor division assignment)
Floor divides the variable by a value and assigns the result to that variable.

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

https://python-reference.readthedocs.io/en/latest/docs/operators/

2. Absolute Basics - Logische Operatoren



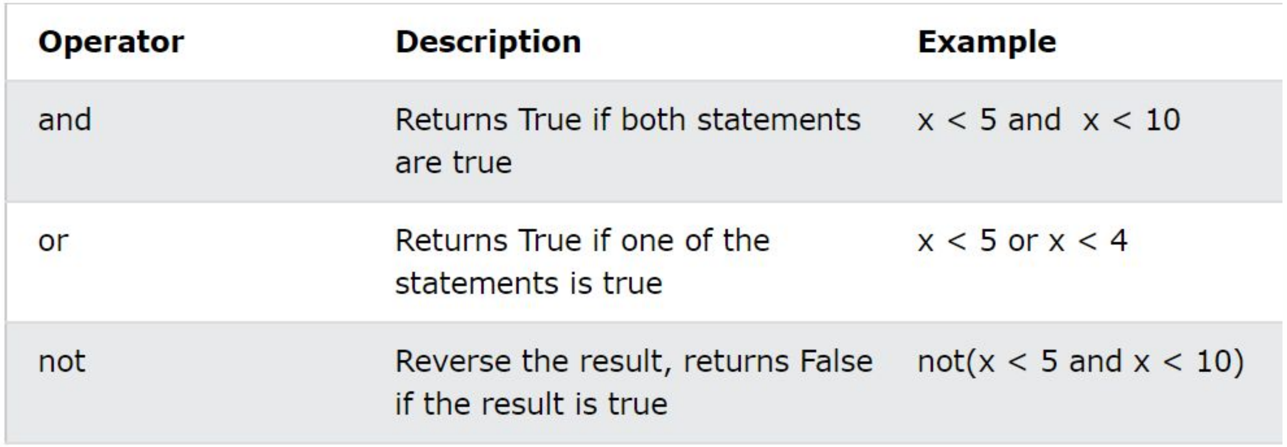
```
print(True and False)
print(True or False)
print(not False and True)

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

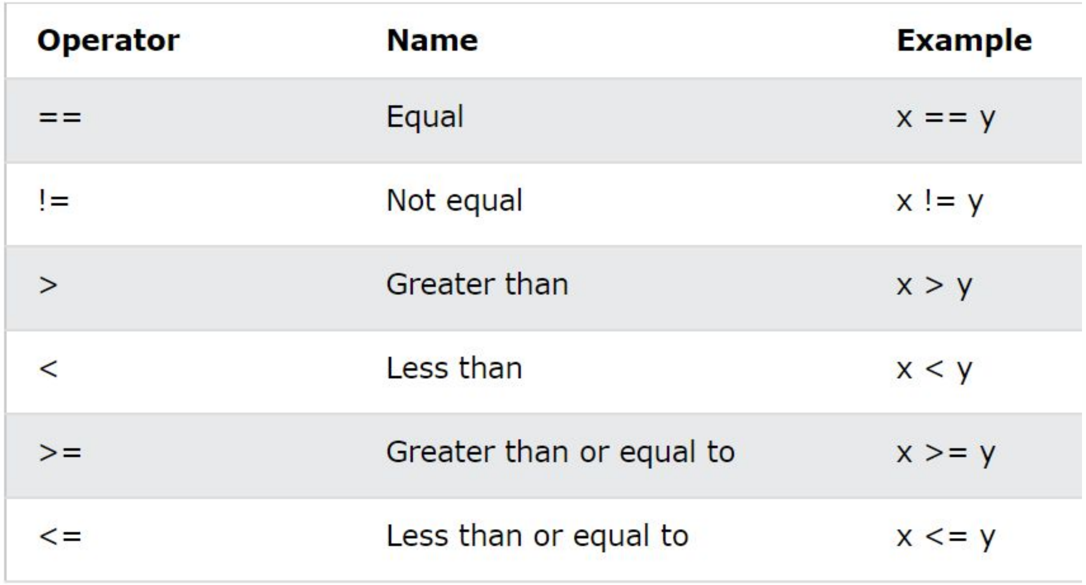
Übersicht Logische Operatoren



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

2. Absolute Basics - Logische Operatoren

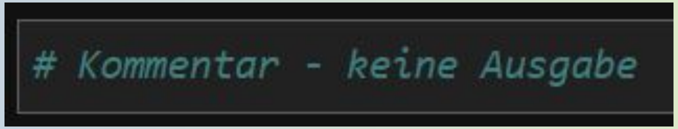
Übersicht Vergleichsoperatoren



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

https://www.w3schools.com/python/python_operators.asp

2. Absolute Basics - Kommentare



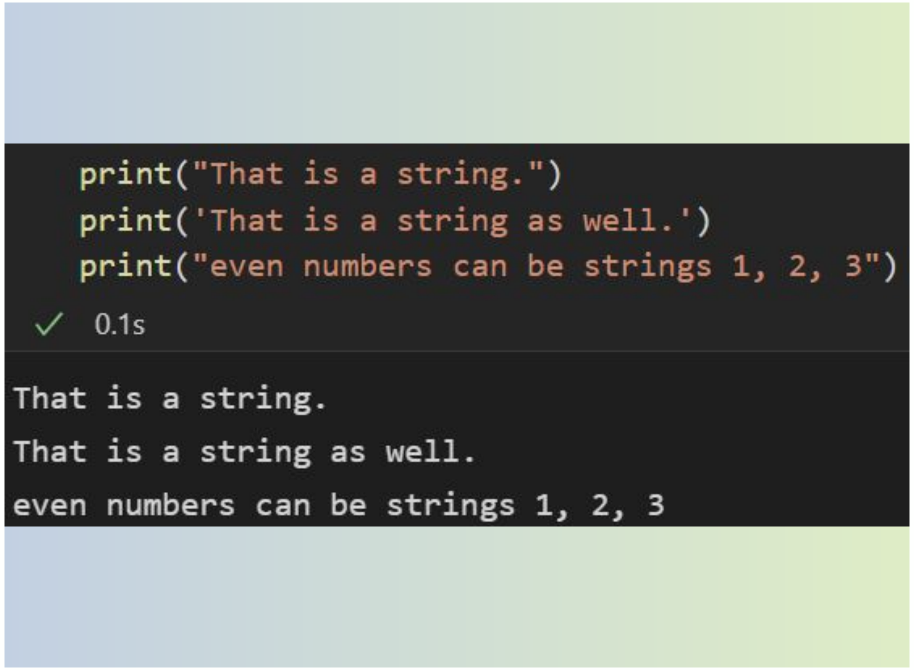
```
# Kommentar - keine Ausgabe
```

- # 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

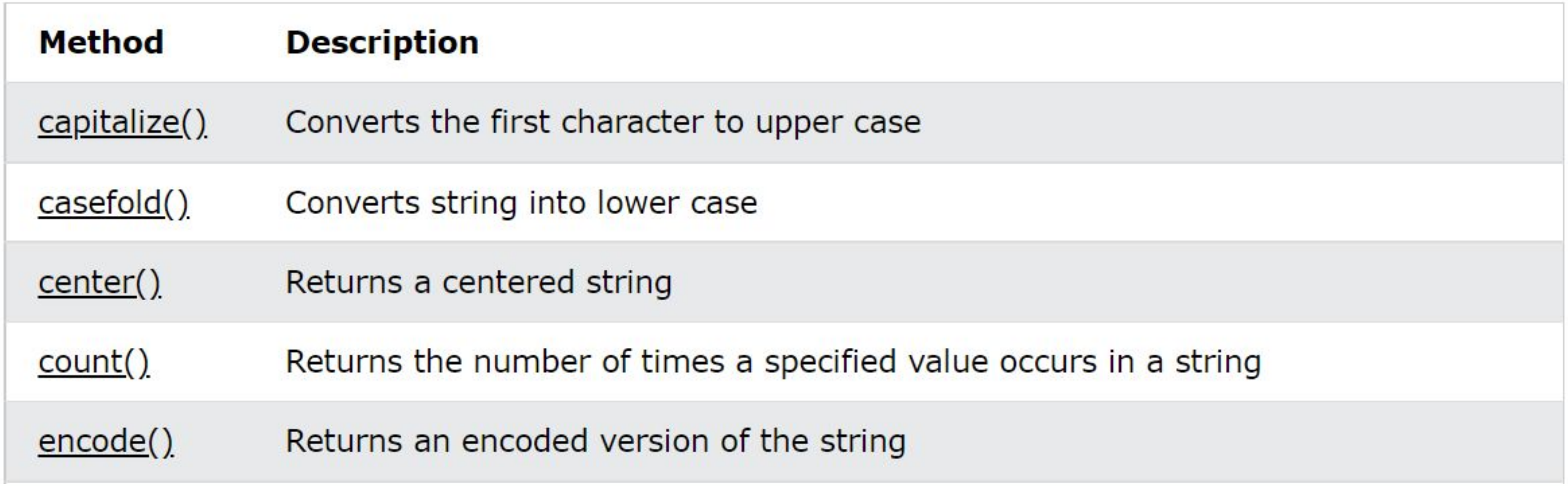
```
That is a string.
That is a string as well.
even numbers can be strings 1, 2, 3
```

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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

4. Datentypen - Strings

Übersicht Methoden I

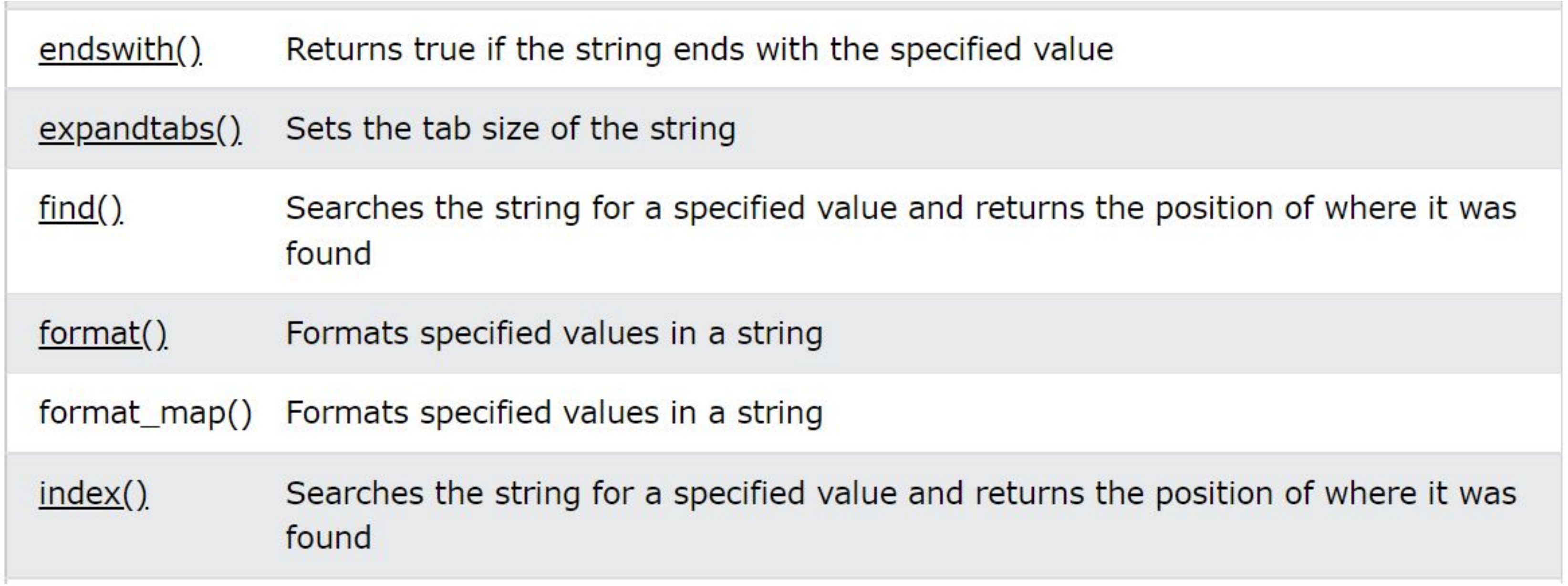


Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string

https://www.w3schools.com/python/python_strings_methods.asp

4. Datentypen - Strings

Übersicht Methoden II

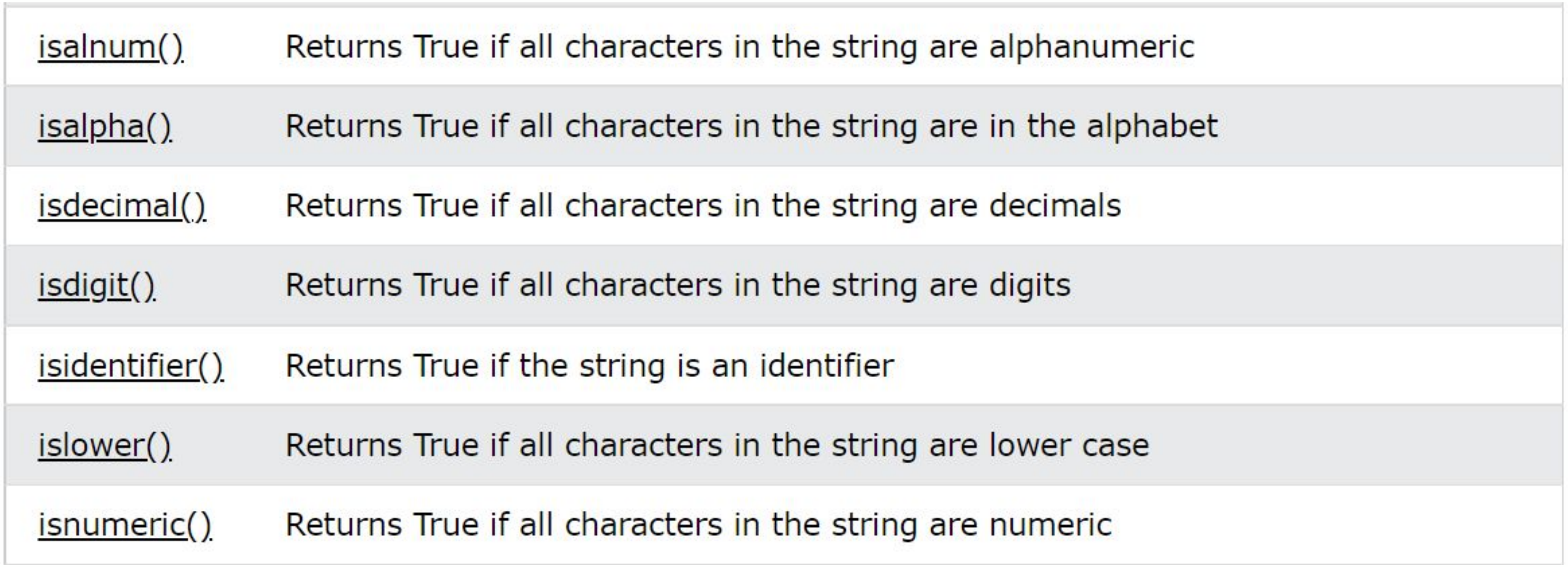


endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string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

https://www.w3schools.com/python/python_strings_methods.asp

4. Datentypen - Strings

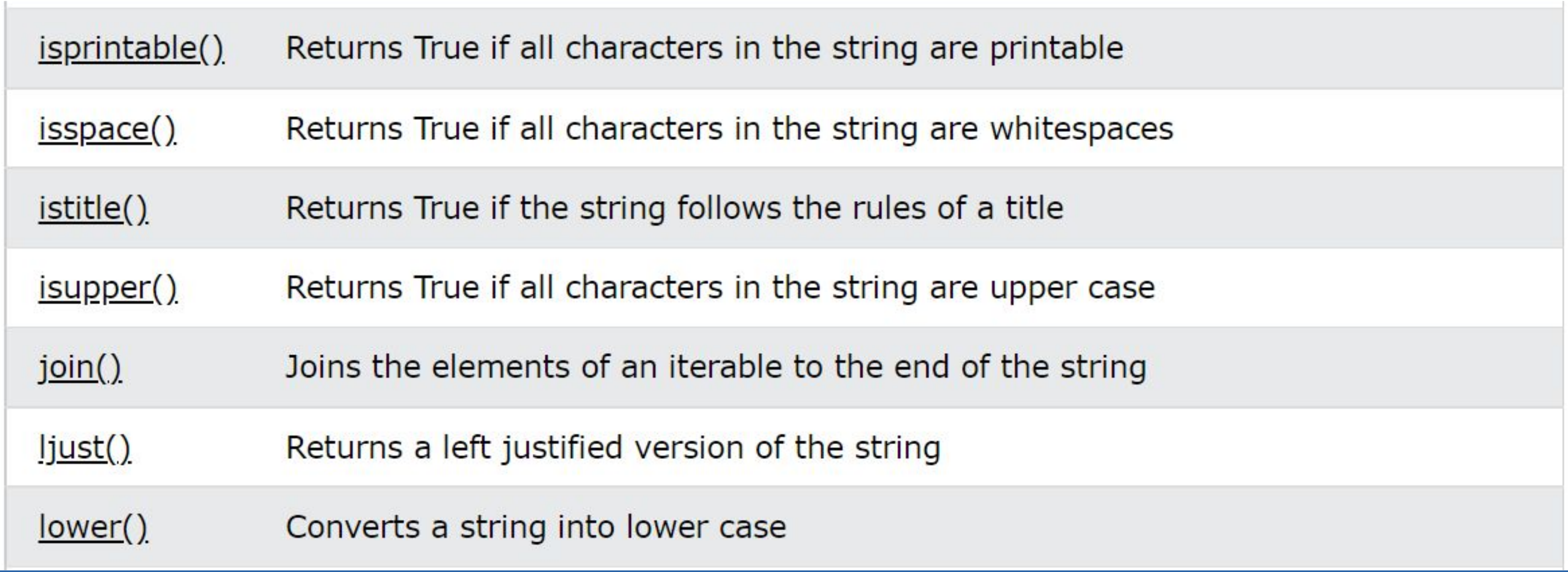
Übersicht Methoden III



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

4. Datentypen - Strings

Übersicht Methoden IV

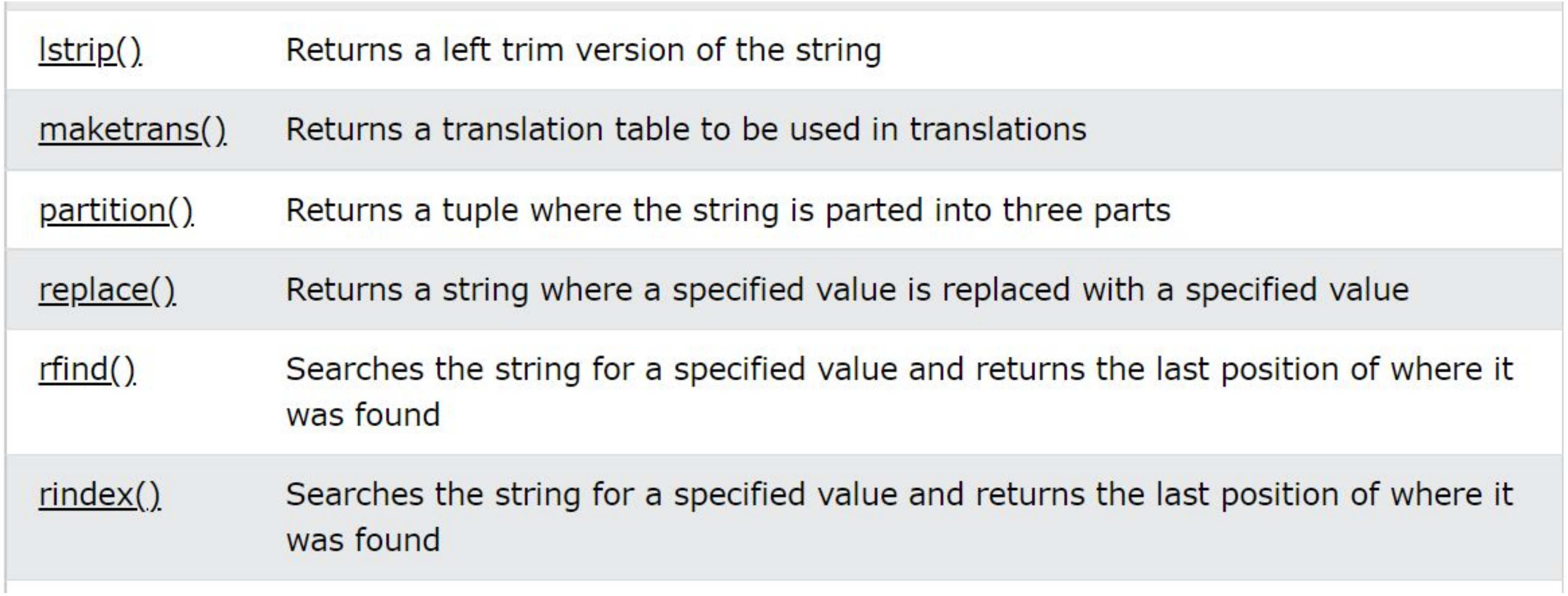


isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements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

https://www.w3schools.com/python/python_strings_methods.asp

4. Datentypen - Strings

Übersicht Methoden V



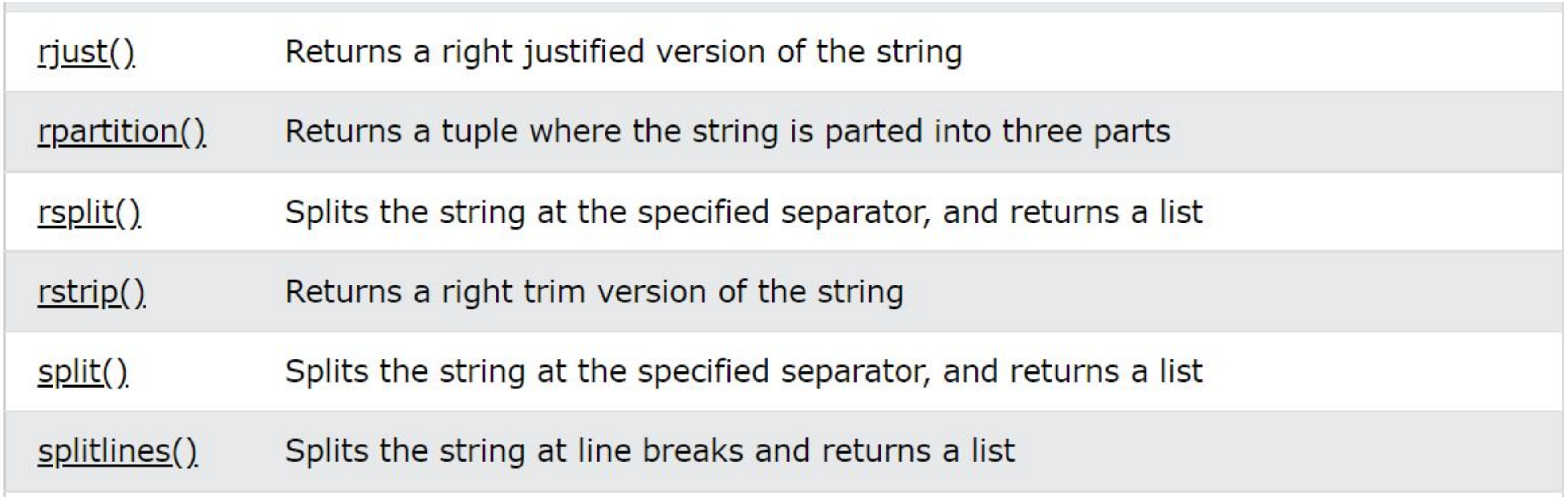
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

https://www.w3schools.com/python/python_strings_methods.asp

4. Datentypen - Strings

Übersicht Methoden VI

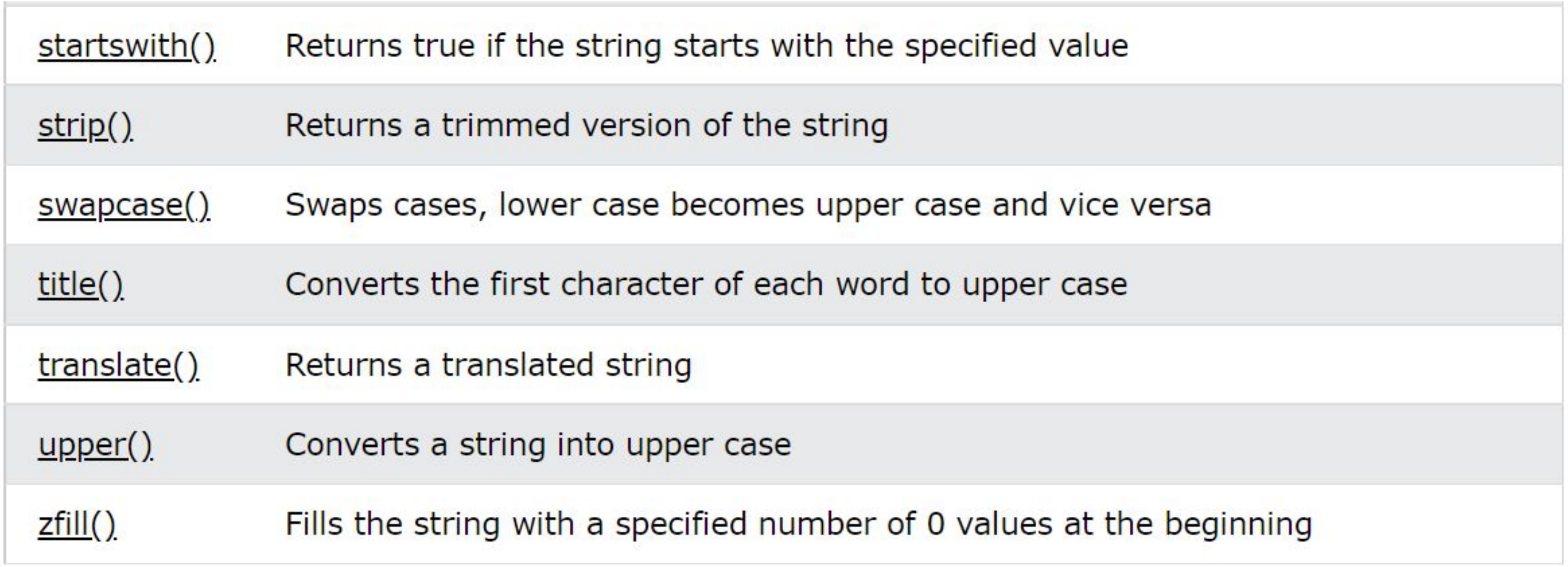


rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator,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

https://www.w3schools.com/python/python_strings_methods.asp

4. Datentypen - Strings

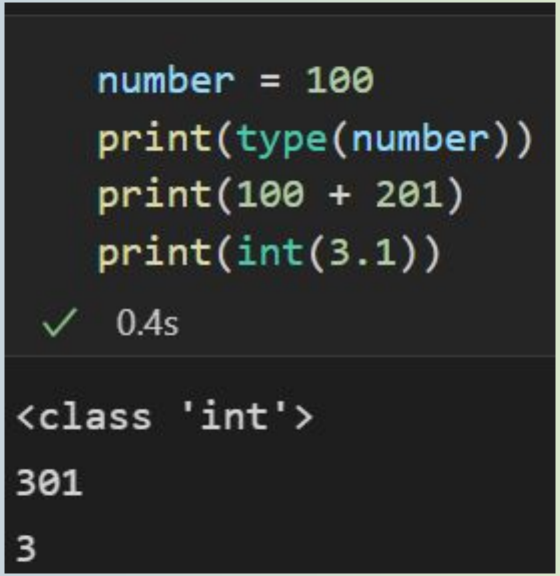
Übersicht Methoden VII



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

https://www.w3schools.com/python/python_strings_methods.asp

3. Datentypen - Ints



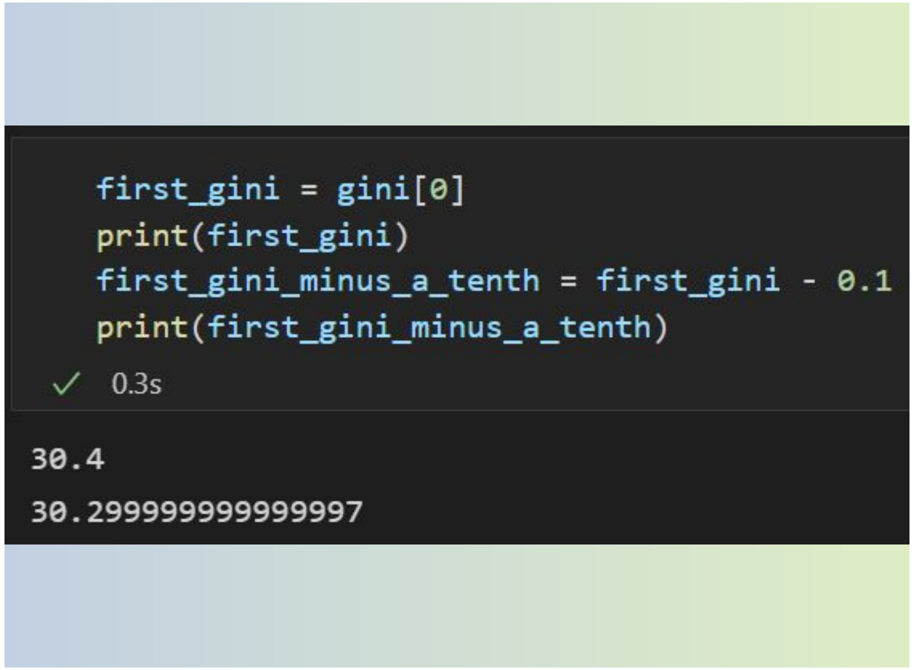
```
number = 100
print(type(number))
print(100 + 201)
print(int(3.1))
```

✓ 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 -∞ gerundet

3. Datentypen - Floats



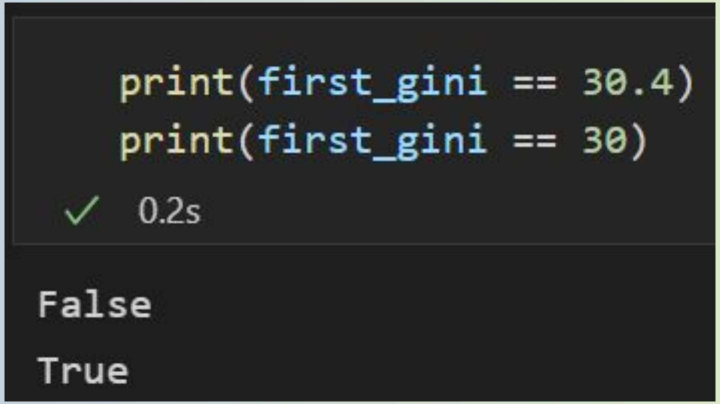
```
first_gini = gini[0]
print(first_gini)
first_gini_minus_a_tenth = first_gini - 0.1
print(first_gini_minus_a_tenth)
```

✓ 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
```

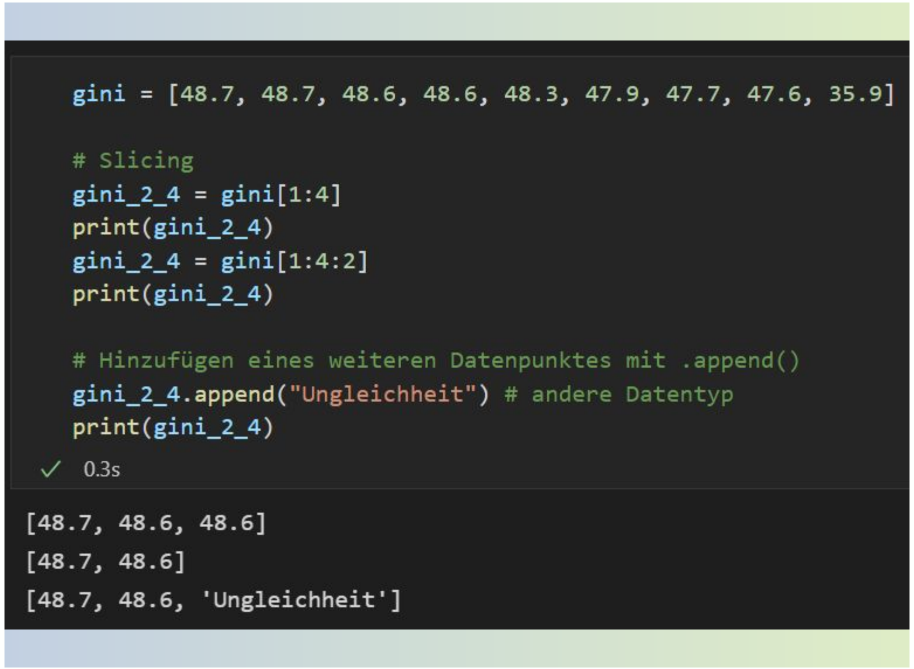
- Wahrheitswerte
- Grundsätzlich nur True (1) oder False (0)

4. Datencontainer

- Listen
- Tuples
- Sets
- Dictionaries

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

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)
```

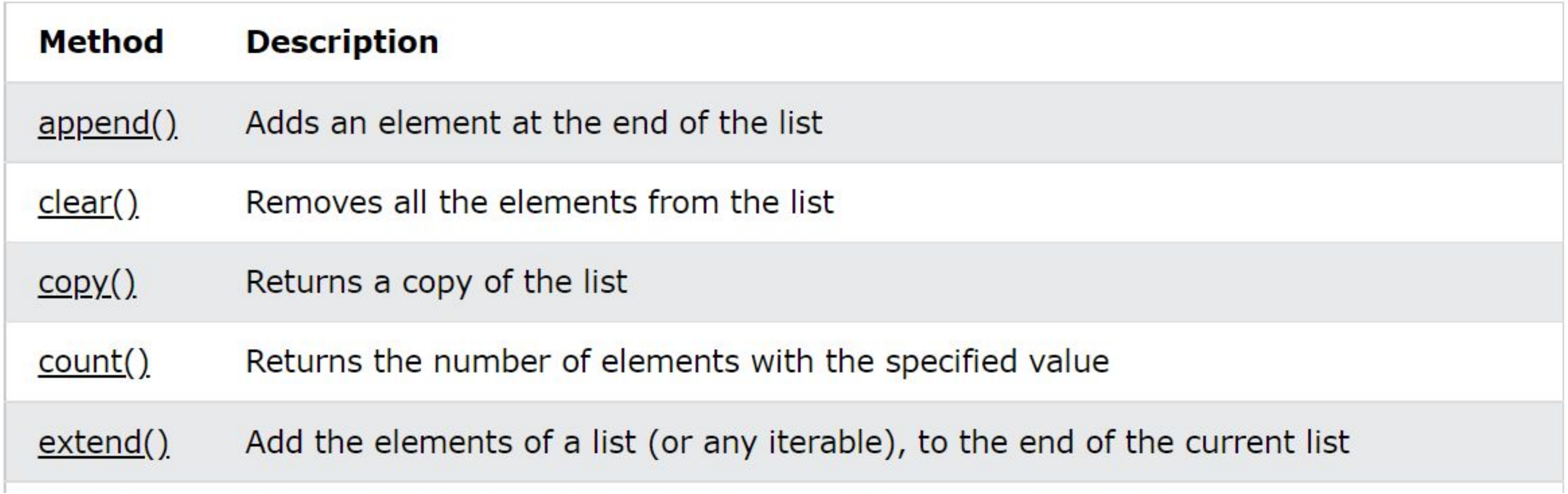
✓ 0.3s

```
[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



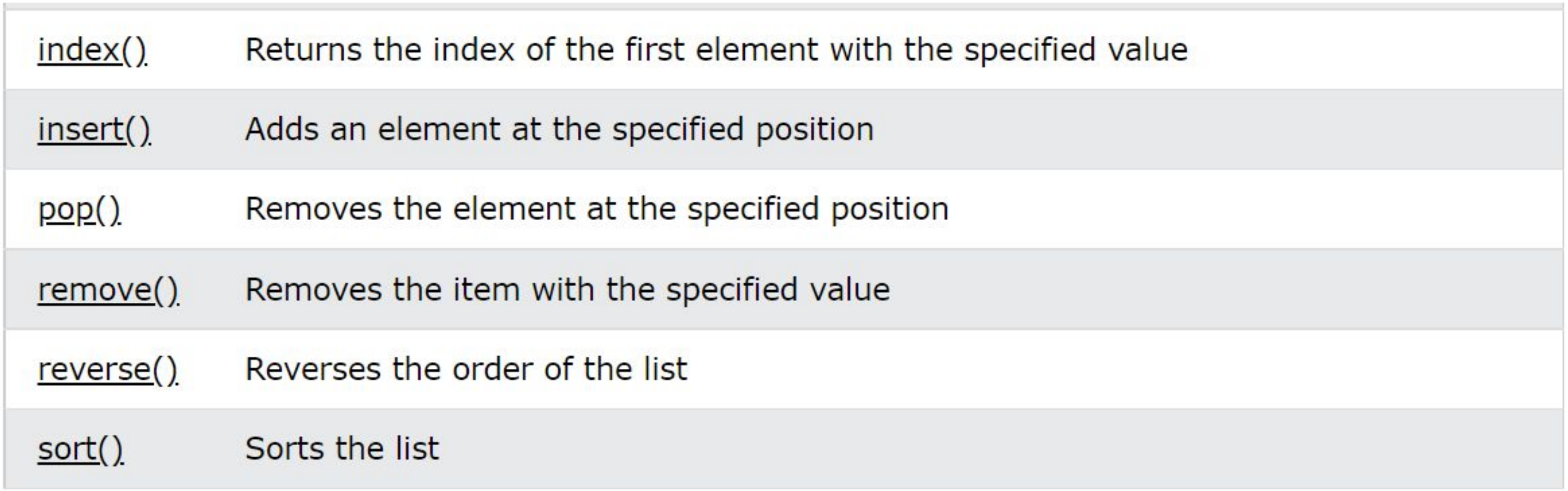
Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

https://www.w3schools.com/python/python_lists_methods.asp

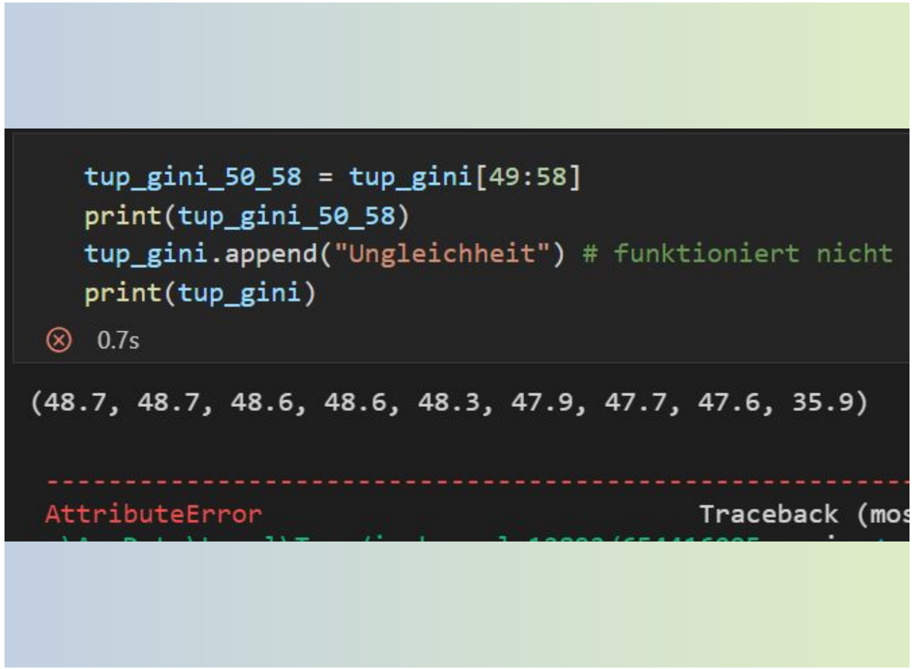
4. Datencontainer - Listen

Übersicht Methoden II



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

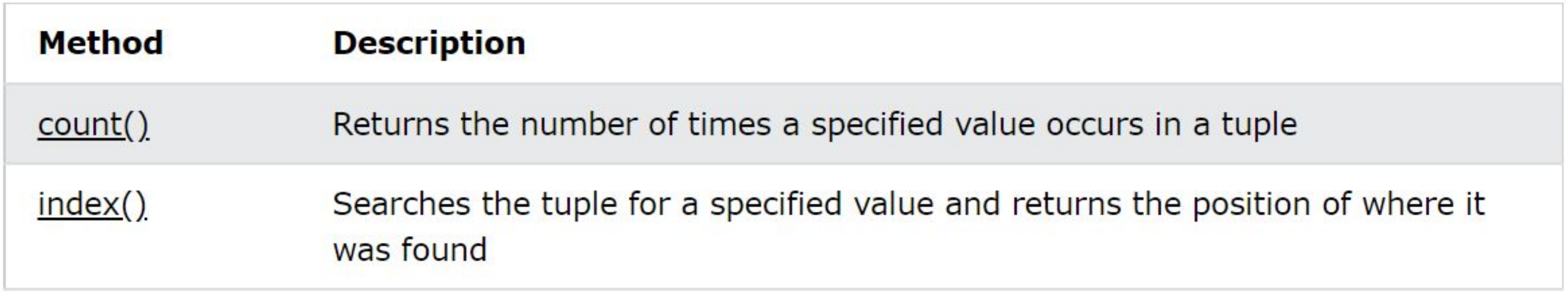
0.7s

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

---------------------------------------------------------
AttributeError                        Traceback (mos
```

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

4. Datencontainer - Tuples

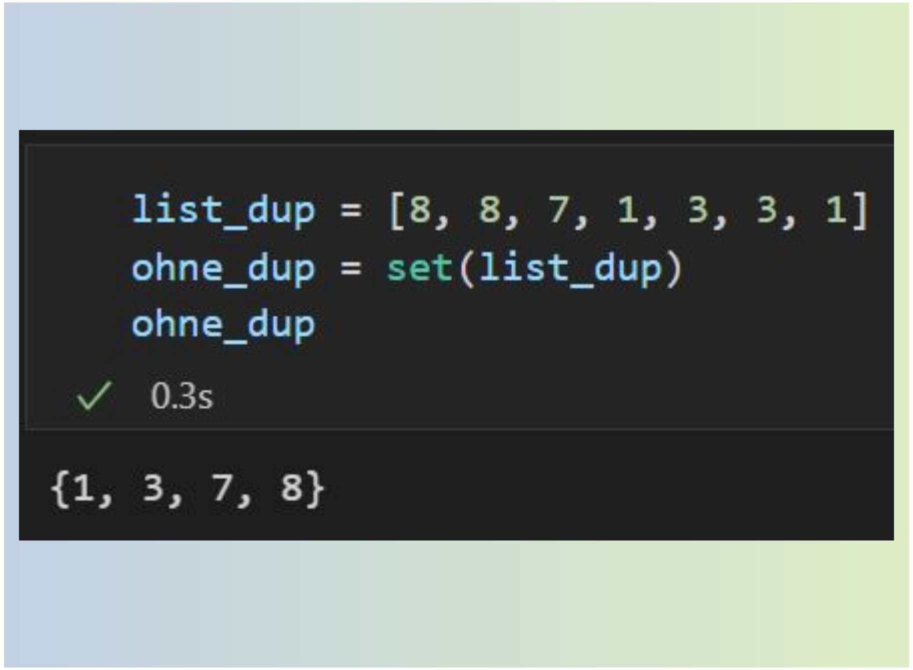
Übersicht Methoden



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

https://www.w3schools.com/python/python_tuples_methods.asp

4. Datencontainer - Sets



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

✓ 0.3s

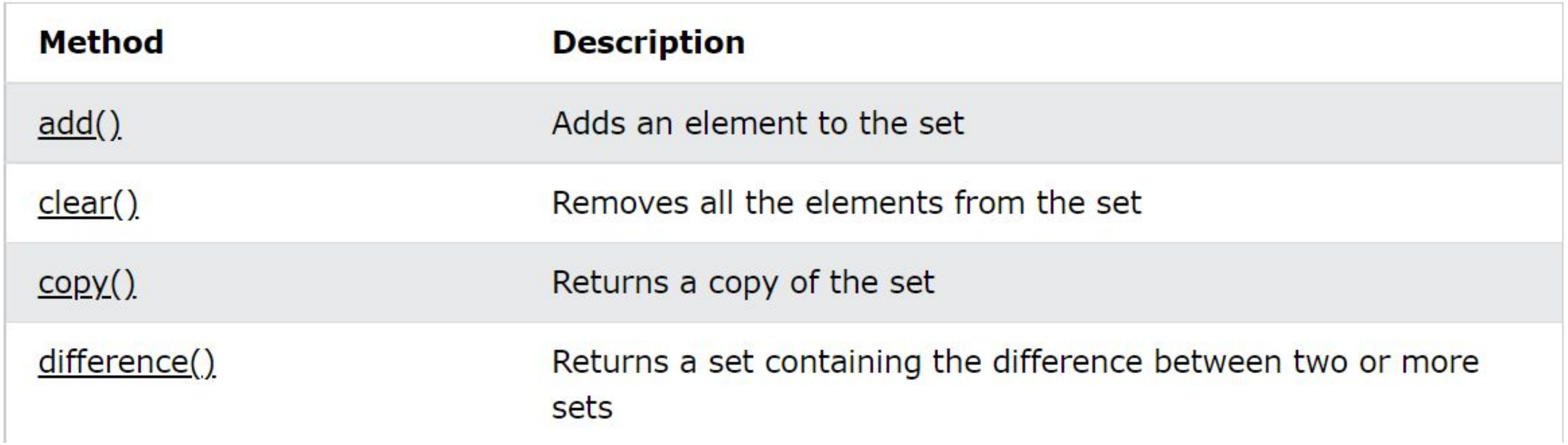
```
{1, 3, 7, 8}
```

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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

4. Datencontainer - Sets

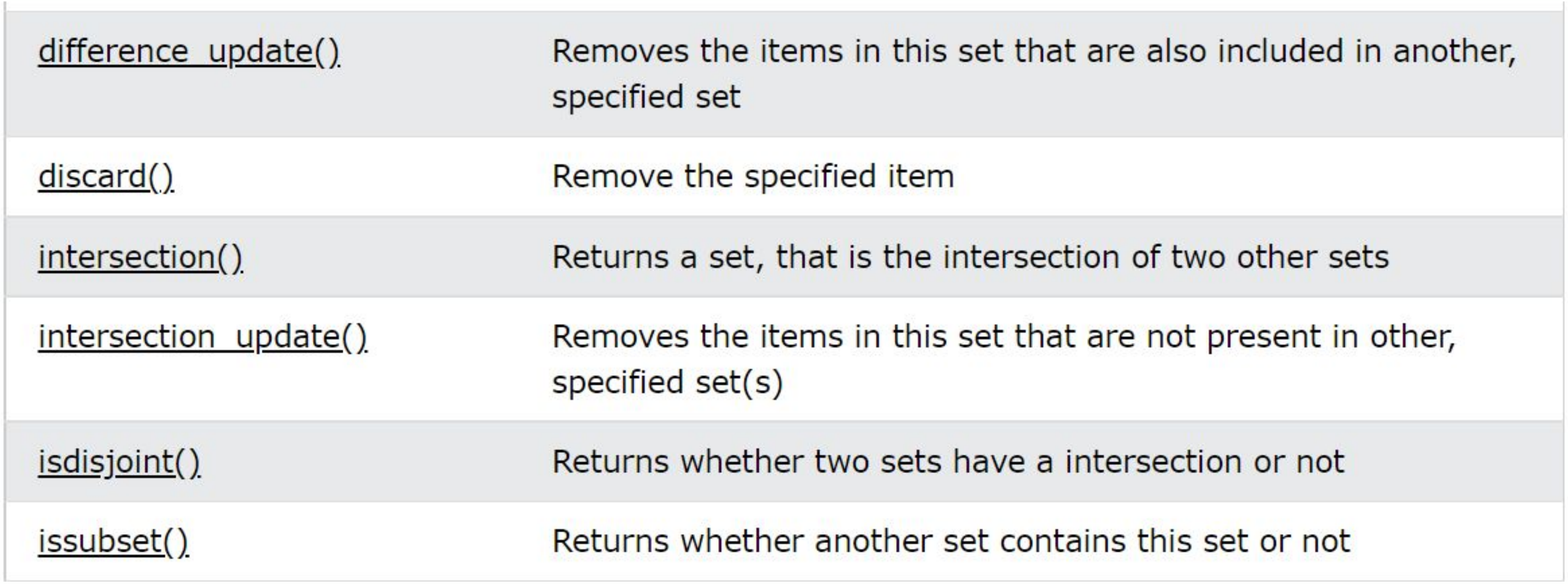
Übersicht Methoden I



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

4. Datencontainer - Sets

Übersicht Methoden II



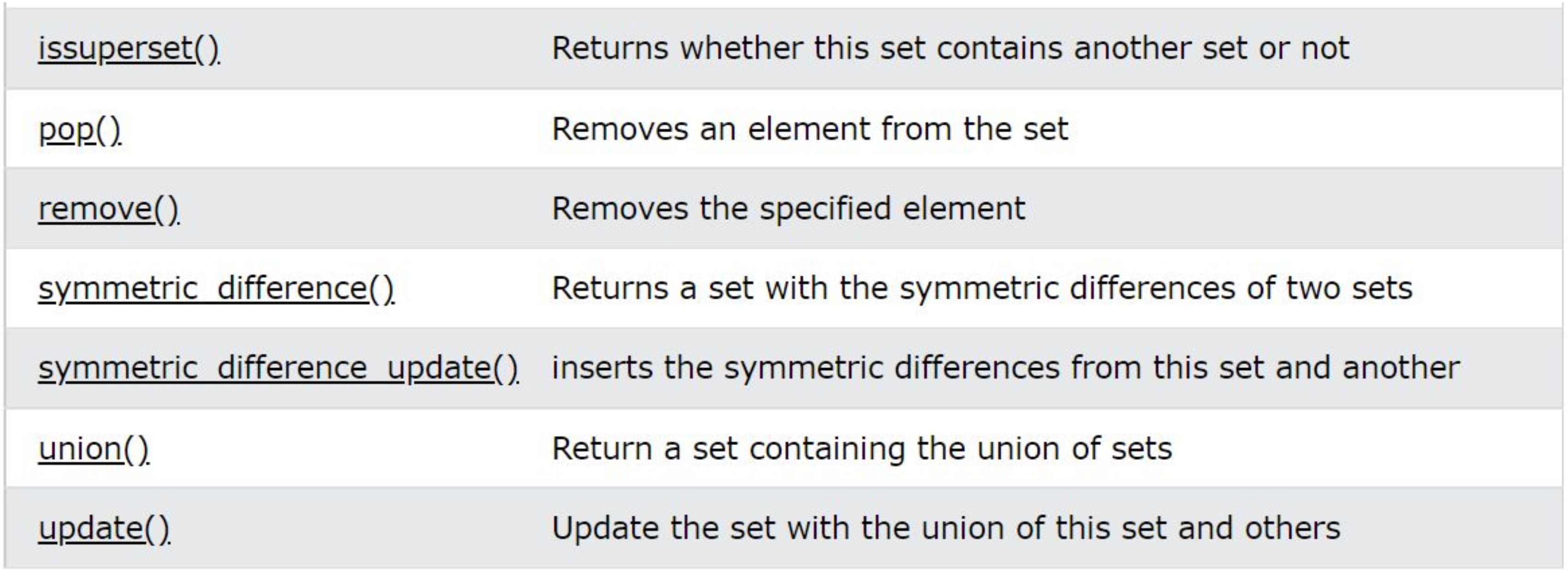
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

https://www.w3schools.com/python/python_sets_methods.asp

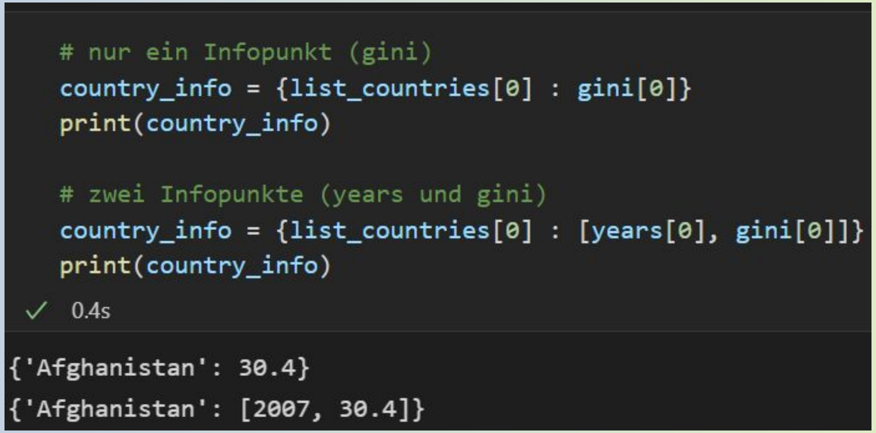
4. Datencontainer - Sets

Übersicht Methoden III



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

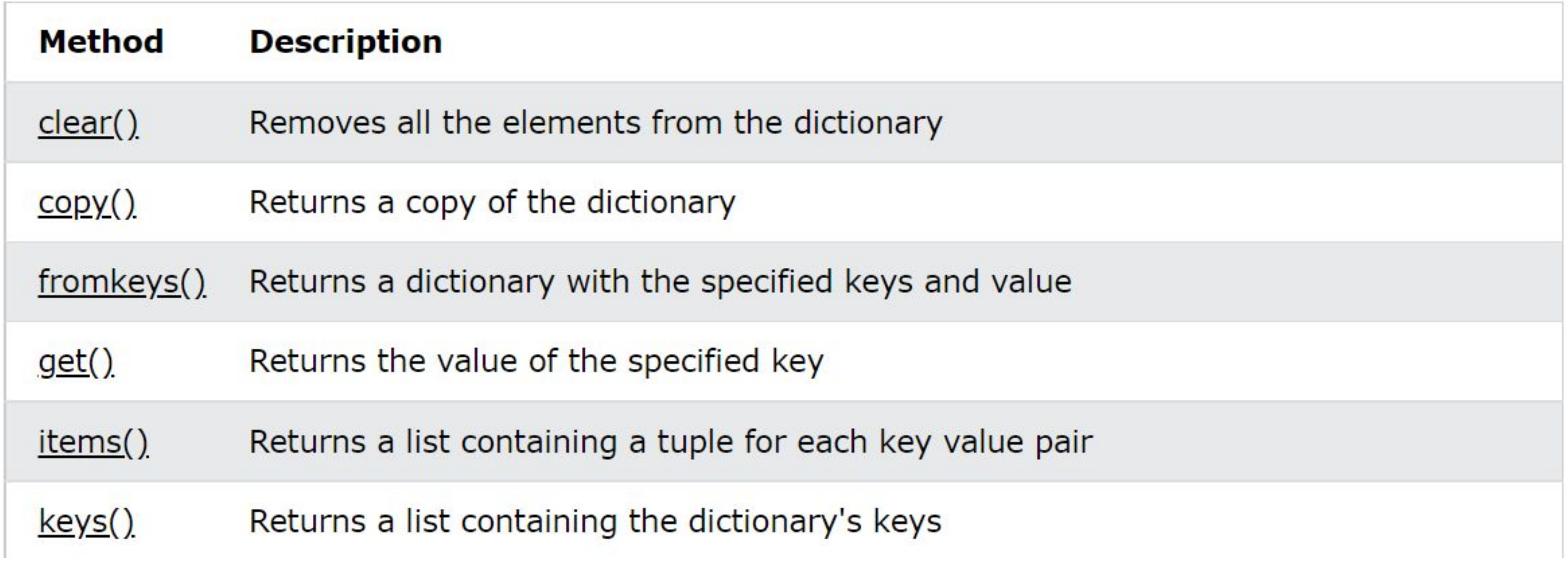
```
{'Afghanistan': 30.4}
{'Afghanistan': [2007, 30.4]}
```

- 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

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

4. Datencontainer - Dictionaries

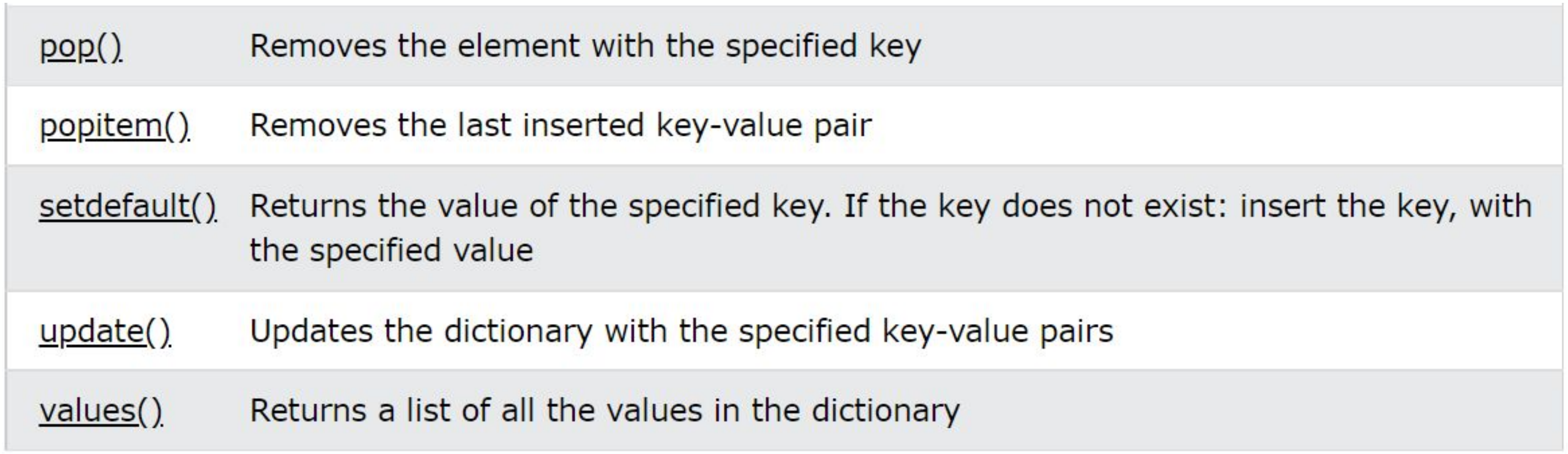
Übersicht Methoden I



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

4. Datencontainer - Dictionaries

Übersicht Methoden II



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

5. Loops und Conditional Statements

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

5. Conditional Statements

- We can use an if **statement** to implement a condition in our code.
- An elif clause executes if the preceding if statement (or the other preceding elif clauses) resolves to False and the condition specified after the elif keyword evaluates to

True.

- True and False are **Boolean values**.
- and and or are **logical operators**, and they unite two or more Booleans.

5. Conditional Statements

Using an if statement to control your code:

```
if True:
    print(1)
if 1 == 1:
    print(2)
    print(3)
```

Combining multiple conditions:

```
if 3 > 1 and 'data' == 'data':
    print('Both conditions are true!')
if 10 < 20 or 4 <= 5:
    print('At least one condition is true.')
```

Building more complex if statements:

```
if (20 > 3 and 2 != 1) or 'Games' == 'Games':
    print('At least one condition is true.')
```

Using the else clause:

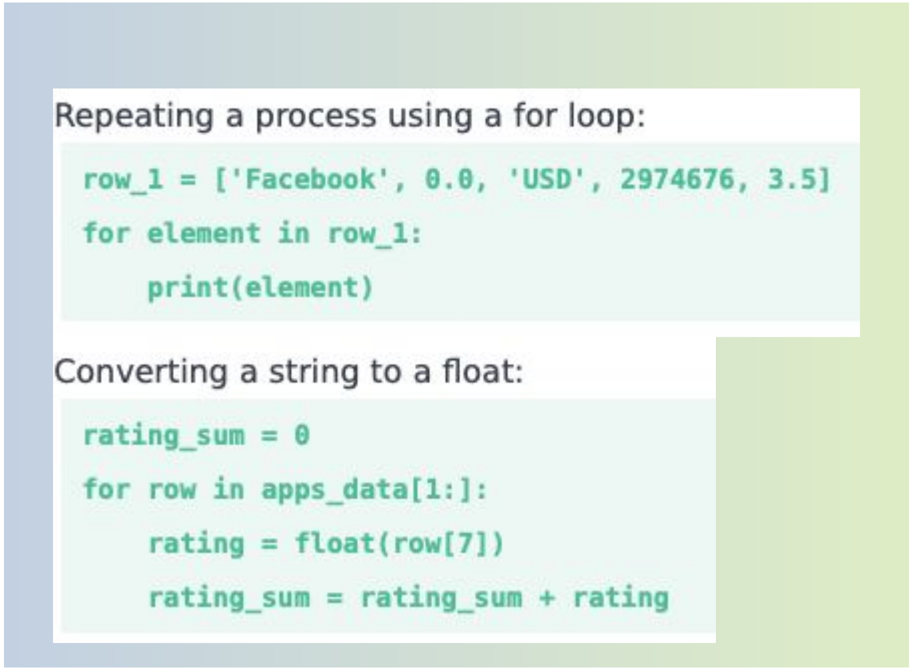
```
if False:
    print(1)
else:
    print('The condition above was false.')
```

Using the elif clause:

```
if False:
    print(1)
elif 30 > 5:
    print('The condition above was false.')
```

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

5. Loops und Conditional Statements - for-Loops



Repeating a process using a for loop:

```
row_1 = ['Facebook', 0.0, 'USD', 2974676, 3.5]
for element in row_1:
    print(element)
```

Converting a string to a float:

```
rating_sum = 0
for row in apps_data[1:]:
    rating = float(row[7])
    rating_sum = rating_sum + rating
```

- We can automate repetitive processes using **for loops**.
- We always start a for loop with **for** (like in **for element in app_ratings:**).
- The indented code in the **body** gets executed the same number of times as elements in the **iterable variable**. If the iterable variable is a list containing three elements, the indented code in the body gets executed three times. We call each code execution an **iteration**, so there will be three iterations for a list that has three elements. For each iteration, the **iteration variable** will take a different value.

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

6. Funktionen

Generally, a function displays this pattern:

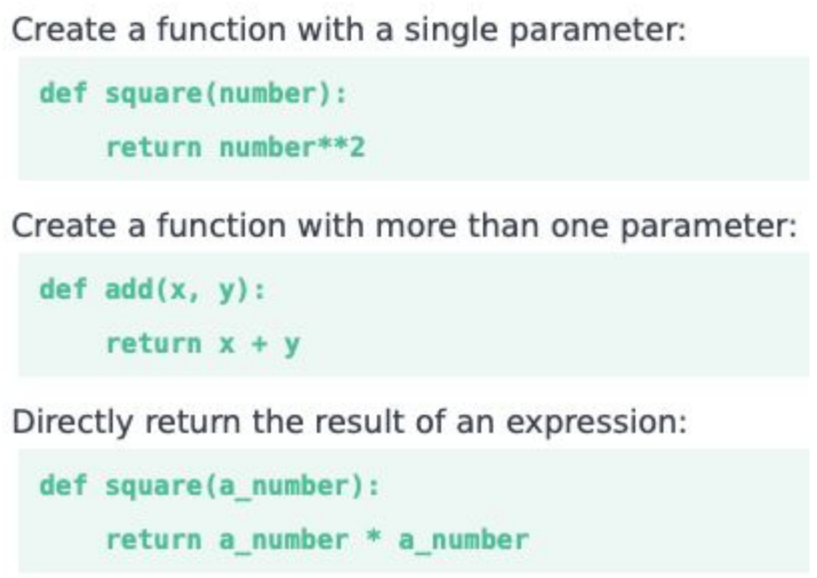
- It takes in an input.
- It processes that input.
- It returns output.

In Python, we have **built-in functions** like **min()** , **len(), sum()** ,and functions that we create ourselves.

Structurally, a function contains a header (which contains the **def** statement), a body, and a **return** statement.

We call input variables **parameters**, and we call the various values that parameters take **arguments**. In **def square(number)** , the **number** variable is a parameter. In **square(number=6)** , the value **6** is an argument that passes to the parameter **number** .

6. Funktionen



Create a function with a single parameter:

```
def square(number):
    return number**2
```

Create a function with more than one parameter:

```
def add(x, y):
    return x + y
```

Directly return the result of an expression:

```
def square(a_number):
    return a_number * a_number
```

Weiterführende Quellen

Lehrbuch: A Beginners Guide to Python 3 Programming (John Hunt)

YouTube: Corey Schafer, Socratica

Übungen: w3resource

Python Cheat-Sheet

CORRELAID
GOOD CAUSES. BETTER EFFECTS.

Vielen Dank für eure Aufmerksamkeit!

…und bis zum nächsten Mal

Feedback-Link: https://forms.gle/d6nojuhW9be7XdPC8

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