

## Python short reference

### Viabiles and value Assignment

```
name = "Darwin"
```

### Simple Datatypes

str - String, "Hallo Welt"

int - Integer, 234

float - Floating point number, 42.23

bool - Boolean- True/False

### Container data types

#### Lists

```
names = ["Noether", "Darwin", "Lovelace"]
```

The Elements of the list can be addressed by an index e.g. `names[0]`

#### Dictionaris

- Key / value pairs

```
person_and_birth_years = {"Noether": 1882, "Darwin": 1809, "Lovelace": 1815}
```

Values are addressed via keys e.g. `person_and_birth_years["Noether"]`

### Operators

- +, -, \*, /
- ==, !=, <, >, <=, >=
- not, and, or

### for-Loops

```
for <variable> in <list/iterable>:  
    <block to execute>
```

### Conditionals

Execution of a code block under a certain condition

```
if <condition>:  
    <block to execute>
```

Exececution of a code block under a certain condition or alternative code block if the conditions is not true

```
if <Bedingung>:  
    <Auszuführender Block>  
else:  
    <Auszuführender Block>
```

Execution of a code block under certain conditions and alternatives

```
if <condition 1>:
    <block to execute 1>
elif <condition 2>:
    <block to execute 2>
else:
    <block to execute 3>
```

## Comments

- Text right of a # is not interpreted

## Using libraries/packages

Importing a library

```
import csv
```

Importing a module of a library

```
import urllib.request
[...]
urllib.request.urlopen
```

Import a library with an alias

```
import pandas as pd
```

## Using functions and methods

### Functions

- Functions group several statements
- Functions can have zero to several parameters
- Functions are called by using their names and round brackets ()
- Examples:
  - `print("Hello World!")`
  - `type(counter)`
  - `len([5, 23, 52])`

### Methods

- Methods are functions that are bound to objects
- Examples
  - `name.upper()`
  - `name.replace("und", "oder")`

## Opening files

A so called file handle is generated with `open(<my_file_name>)`. The content of the file can be read and returned as string with the `read` methode.

```
my_file_handle = open("My_great_file.txt"):
file_content = my_file_handle.read()
```

Alternatively, the file can read line by line:

```
for line in open("My_great_file.txt"):
    print(line)
```

## Reading recommendation

- “Automate the Boring Stuff with Python”, Al Sweigart, <https://automatetheboringstuff.com/>

(CC-BY Till Sauerwein and Konrad Förstner)