

Introduction to programming in Python

Jules Kreuer

Uni Tübingen

fsi@fsi.uni-tuebingen.de

contact@juleskreuer.eu

February 17, 2022

Based on:

Ana Bell, Eric Grimson, and John Guttag.

6.0001 Introduction to Computer Science and Programming in Python.

Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare

<https://ocw.mit.edu>.

License: Creative Commons BY-NC-SA.

Nick Parlante, John Cox, Steve Glassman, Piotr Kaminski, Antoine Picard.

Google's Python Class.

July 2015. Google LLC

License: Creative Commons BY 2.5.

Part 1: Hello World

- Introduction
- Installation
- REPL

Break

Part 1: Hello World

- Introduction
- Installation
- REPL

Break

Part 2: Basics

- Common operators
- Data types, type-casting
- Lists, dicts
- Control flow: for, while, break, continue

Break

Part 3: Abstraction

- Functions, variable scope, lambda
- Exceptions
- recursion
- Objects, Classes

Break

Part 3: Abstraction

- Functions, variable scope, lambda
- Exceptions
- recursion
- Objects, Classes

Break

Part 4: Development

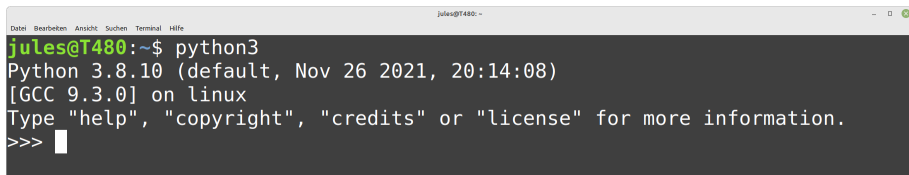
- Arguments
- Modules / imports
- Virtual-Envs
- Tests

```
jules@T480:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license"...

>>> a = 5
>>> a
5
>>> a = "Hello World"
>>> a
'Hello World'
>>> a + "!"
'Hello World!'
>>>
```

<https://www.python.org/downloads/>
Debian / Ubuntu: *sudo apt install python3*

Type in your shell: *python3*

A screenshot of a terminal window titled 'jules@T480: ~'. The window has a menu bar with 'Datei', 'Bearbeiten', 'Ansicht', 'Suchen', 'Terminal', and 'Hilfe'. The terminal content shows the command 'python3' being executed, resulting in the Python 3.8.10 shell. The prompt is 'jules@T480:~\$' and the output is 'Python 3.8.10 (default, Nov 26 2021, 20:14:08) [GCC 9.3.0] on linux'. Below this, it says 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt changes to '>>>' with a cursor.

```
jules@T480:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Figure: Python3 REPL

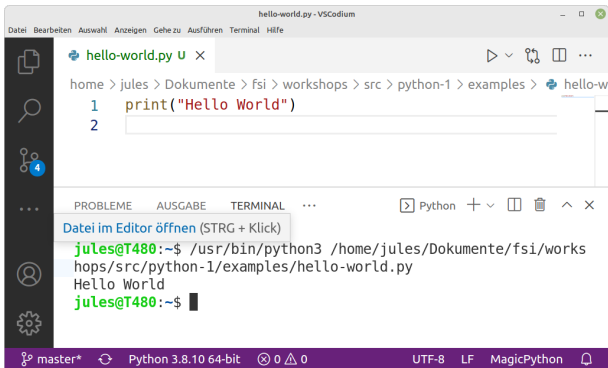
Running code

- REPL
- python3 file args

Example

```
python3 hello-world.py
```

Combining Editor and Interpreter



The screenshot displays the VS Codium interface. The top menu bar includes 'Datei', 'Bearbeiten', 'Auswahl', 'Anzeigen', 'Gehe zu', 'Ausführen', 'Terminal', and 'Hilfe'. The editor window, titled 'hello-world.py - VSCodium', shows a Python script with two lines: `1 print("Hello World")` and `2`. The file explorer on the left shows the path `home > jules > Dokumente > fsi > workshops > src > python-1 > examples > hello-w`. The terminal panel at the bottom, titled 'Python', shows the command `jules@T480:~$ /usr/bin/python3 /home/jules/Dokumente/fsi/workshops/src/python-1/examples/hello-world.py` and its output `Hello World`. The status bar at the bottom indicates the current state: `master*`, `Python 3.8.10 64-bit`, `0 0 0`, `UTF-8`, `LF`, `MagicPython`, and a bell icon.

```
hello-world.py U X
home > jules > Dokumente > fsi > workshops > src > python-1 > examples > hello-w
1 print("Hello World")
2

PROBLEME AUSGABE TERMINAL ... Python + - [ ] [X] ^ X
Datei im Editor öffnen (STRG + Klick)
jules@T480:~$ /usr/bin/python3 /home/jules/Dokumente/fsi/workshops/src/python-1/examples/hello-world.py
Hello World
jules@T480:~$
```

master* Python 3.8.10 64-bit 0 0 0 UTF-8 LF MagicPython

Figure: VS Codium

Possible IDEs / Editors:

- VS Codium: <https://vscodium.com/>
- PyCharm: <https://www.jetbrains.com/pycharm/>
- Atom: <https://atom.io/>
- ...

- Content: `print("Hello World")`
- Run it!

Basic operators and types

Just like 'any other' language.

Math

```
s = (a + b - c) / d * e
p = a ** 2 # a to the power of 2
b = a^2    # bitwise shifting
m = a%2    # mod
```

Numeric types

```
int, float, complex
i = 1 = int("1") = int(1.0)
f = 4.2
c = 4+2j
```

Strings

```
s = "Hello " + "World"
c = "A" * 10 + "HHHH"
S = s.upper()
length = len(S)    # Returns Integer
pos = s.find("W")  # Return Integer (Position of first W)
```

Text types

```
str
s = str(1)
```

Booleans

```
a = (True or False) and not False
```

Boolean types

```
bool  
t = bool(1) = bool("Not Empty")  
f = bool(0) = bool("")
```


Comparison

<, >, ==, !=, <=, >=

Example

```
t = 3 < 5
```

```
t = not "A" == "B"
```

```
f = 4.2 == 2
```

```
f = 0 == "Hello" # Comparision in between types is possible
```

Exercise

Desired output: 'The sum of 41.8 and 0.2 is 42'.

Use following variables:

```
i = 41.8
```

```
f = 0.2
```

```
prefix = "The sum of "
```

Lists

Mutable, dynamic in length, non-homogenous, ordered

```
aList = [1, 2, 3, 4, "What?", 6]
aList[0]           # -> 1
aList[4:]          # -> ['What?', 6]
aList[1::2]        # -> [2, 4, 6]
aList[-1]          # -> 6
aList.append(7)     # -> [..., 6, 7]
aList.extend([8,9]) # -> [..., 6, 7, 8, 9]
aList[0] = "New Zero"
general form: [from:to:step]
```

Lists

Mutable, dynamic in length, non-homogenous, ordered

```
aList = [1, 2, 3, 4, "What?", 6]
aList[0]           # -> 1
aList[4:]          # -> ['What?', 6]
aList[1::2]        # -> [2, 4, 6]
aList[-1]          # -> 6
aList.append(7)     # -> [..., 6, 7]
aList.extend([8,9]) # -> [..., 6, 7, 8, 9]
aList[0] = "New Zero"
general form: [from:to:step]
```

Tuples

Non-Mutable, fixed length, non-homogenous

```
aTuple = ("A", "a", 1)
a[0]     # -> "A"
```

Dicts

Mutable, dynamic in size, non-homogenous, unordered^a

```
d = {"key": "value", 1: 3}
d["key"]      # -> "value"
d["new"] = 2  # Insert new value to d
d.keys()      # -> ["key", 1, "new"]
d.values()    # -> ["value", 3, 2]
d.items()     # -> [("key", "value"), (1, 3), ("new", 2)]
```

^aSomehow..

See: <https://docs.python.org/3/tutorial/datastructures.html>

Control flow: if, for, while, break, continue

Regular control flow with if:

```
if condition:
    doThis()
elif cond2:
    doThat()
else:
    otherWise()
```

Looping has two different approaches:

`while / condition`

```
i = 0
while i < 10:
    print(i)
    i = i + 1
```

Looping has two different approaches:

while / condition

```
i = 0
while i < 10:
    print(i)
    i = i + 1
```

for / iterable

```
for element in iterable:
    print(e)
```


Iterables: something with an order and members.

Example

```
tuples = (0, 1,2,3,4)
lists  = [0, 1,2,3,4]
string = "Hello World"
dicts  = {"a", 2}
range(0, 6, 2) # start, stop, interval
               # somehow comparable to [0,2,4]
file objects
...
```

for / iterable

```
for element in iterable:  
    print(e)
```

for / iterable

```
for element in iterable:  
    print(e)
```

Example

```
for i in range(5):  
    print(e) # 0, 1, 2, 3, 4  
for c in "Hello World":  
    print(e) # Every char  
for k in {"k": "v", "k2": "v2"}:  
    print(k) # Only the keys  
for k, v in {"k": "v", "k2": "v2"}.items():  
    print(k, v) # Unpacking
```

Unpacking:

- Object with ordered members
- Number of vars equal to members¹.

Example

```
a, b, c = (1, 2, 3)
```

```
a, b,    = [1,2]
```

¹ $x, *xs = [1, 2, 3, 4] \rightarrow xs = [2,3,4]$

Exit the loop early?

Break

```
while True:    # works for "for i in .." aswell
    doThis()
    if exitCondition:
        break
```

Skip to the next element?

Continue

```
for i in range(4):
    if i == 2:
        continue
    print(i)
-> 0, 1, 3
```

Exercise

Implement a basic python number guessing game.

1. Generate a random number.
2. Ask for a guess.
3. Check if guess was correct.
4. If not, say if number was smaller / larger
5. Repeat from step 2, but only 7 times max.

Use following functions:

```
from random import randint(a, b)
randint(0,1024)  # random integer N such that  $a \leq N \leq b$ 
input("Number?") # Takes input from user
```