Python Kurs 2019/2020

1: Einführung, Shell, Strings, Variablen

Bernhard Mallinger

`b.mallinger [at] gmx.at`

https://totycro.github.io/python-kurs

Kurs

- Keine Vorkenntnisse notwendig
- Programmieren ist auch Handwerk
  - Selbst programmieren notwendig
  - Keine Angst vor Fehlern, alles ausprobieren!
- Modus: Vortrag + Übungsbeispiele
- Kurs soll Grundlagen vermitteln und befähigen, sich in speziellere Gebiete einzuarbeiten

Kurs

- Keine Vorkenntnisse notwendig
- Programmieren ist auch Handwerk
  - Selbst programmieren notwendig
  - Keine Angst vor Fehlern, alles ausprobieren!
- Modus: Vortrag + Übungsbeispiele
- Kurs soll Grundlagen vermitteln und befähigen, sich in speziellere Gebiete einzuarbeiten

---

- Vorschläge zu konkreten Anwendungen von Python erwünscht!
- Was möchtet ihr lernen?

Python

- Elegant
  - High-Level
  - Grunddatentypen (Listen, Dictionaries)
- Wächst seit 1991 beständig durch gutes Design
- Plattformunabhängig
- Viele Anwendungsbereiche
- Sprache VS Interpreter

Python

- Elegant
  - High-Level
  - Grunddatentypen (Listen, Dictionaries)
- Wächst seit 1991 beständig durch gutes Design
- Plattformunabhängig
- Viele Anwendungsbereiche
- Sprache VS Interpreter

- Viele Programmiersprachen ähnlich
  → Wissen übertragbar
- Alle Programmiersprachen gleich mächtig

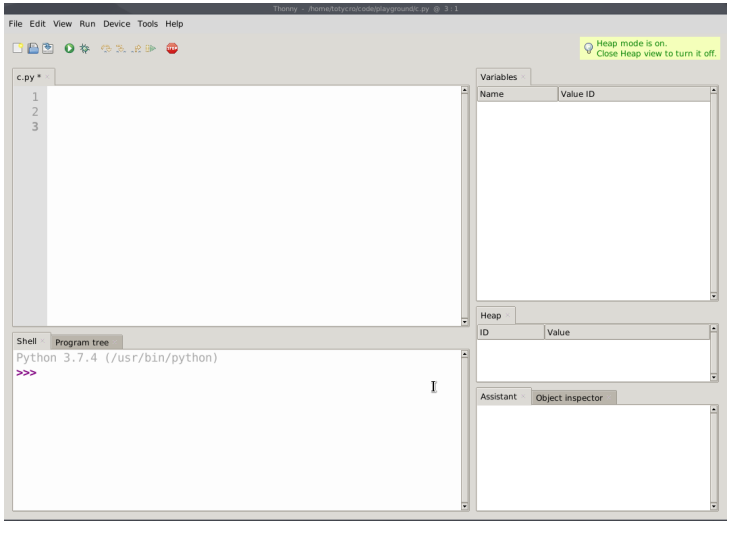
Weitere Einführungen

Einführungen in Python für Programmierneulinge:

- https://thepythonguru.com
- https://overiq.com/python-101/
- https://cscircles.cemc.uwaterloo.ca/

Thonny

- Einfache Entwicklungsumgebung zum Beginnen mit Python
- https://thonny.org/



Die Python Shell

Zahlen

- Ganze Zahlen (`int`): `0`, `4`, `-27`
- Fließkommazahlen (`float`): `1.0`, `4.1238`, `-27e4`
- "Taschenrechner"

```
>>> 4
4
>>> 4 + 2
6
>>> 5 / 3
1.6666666666666667
>>> 5 // 3
1
>>> ((4 + 7) * 27 / 14 ) ** 6
91152898.25473033
```

Funktionsaufrufe

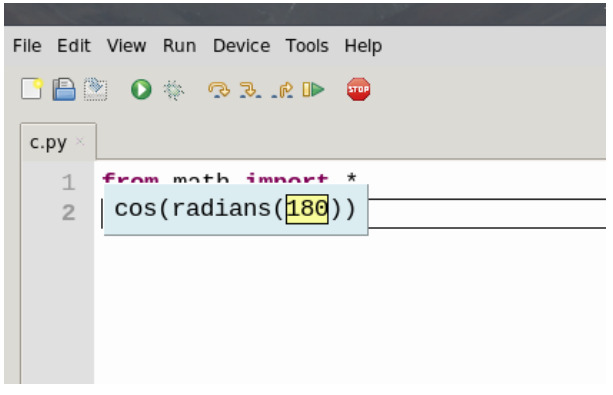
- Inspiriert von Mathematik, ist aber nicht das selbe

```
>>> min(4, 3, 5)
3
>>> max(4, 3, 5)
5
>>>        math
>>> math.log(10)
2.302585092994046
>>> math.radians(180)
3.141592653589793
>>> math.cos(3.141592653589793)
-1.0
>>> math.cos(math.radians(180))
-1.0
```

Thonny Debugger

```
     math        *
cos(radians(180))
```

- "Bug"-Button um bei der Ausführung zuzuschauen (*Debuggen*)
- Step over, Step into



Selber entdecken (1/2)

```
>>> help(math.cos)
Help on built-   function cos    module math:

cos(x, /)
    Return the cosine of x (measured    radians).

>>> help(dir)
Help on built-   function dir    module builtins:

[..]
          an alphabetized list of names comprising
    (some of) the attributes of the given object,
        of attributes reachable      it.
[..]

>>> dir(math)
[..., 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', ...]

>>>
```

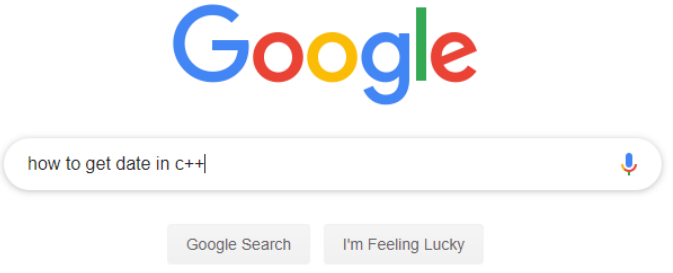
Selber entdecken (2/2)

AUFGABE

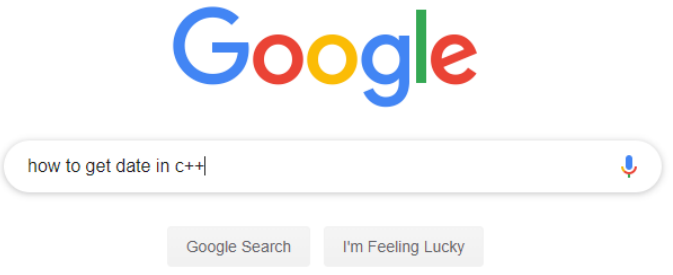
- Welche Funktionen findest du in `math`?
- Was tun sie genau?
- Wie kannst du sie verwenden?

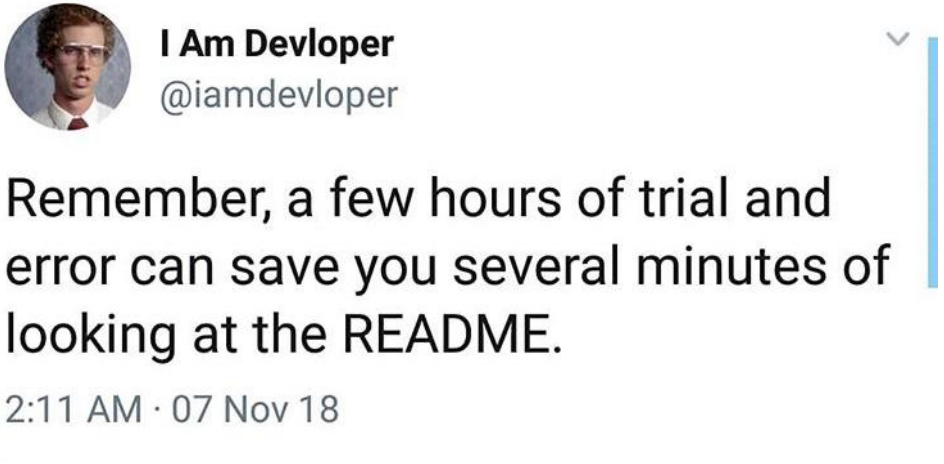
Zusätzlich zu `dir` und `help` ist die offizielle Dokumentation oft sehr nützlich:
https://docs.python.org/3/library/math.html

when you start working
as a programmer



10 years of experience later





https://www.reddit.com/r/ProgrammerHumor/comments/ag3u6c/im_so
https://www.reddit.com/r/ProgrammerHumor/comments/bo1ggo/google
https://www.reddit.com/r/ProgrammerHumor/comments/covgb5/lamo/

Fehlermeldungen

Bei Problemen versucht Python euch genaue Hilfestellung zu geben

```
>>> math.cos(0)
Traceback (most recent call last):
  File "<pyshell>", line 1,    <module>
NameError: name 'math'       defined
>>> 4 / 0
Traceback (most recent call last):
  File "<pyshell>", line 1,    <module>
ZeroDivisionError: division by zero
>>> 3 +
  File "<pyshell>", line 1
    3 +
      ^
SyntaxError: invalid syntax
```

String (Zeichenkette)

```
>>> "hello world"
'hello world'
>>> "hello" + " " + 'wo' + 'rld'
'hello world'
>>> len("hello")
5
```

`"` und `'` gleichbedeutend, jeder String darf aber nur eine Variante verwenden:

```
>>> "boku'
  File "<pyshell>", line 1
    "boku'
         ^
SyntaxError: EOL while scanning string literal
```

- Syntax: Grammatik
- EOL: End of line

Webbrowser

```
antigravity
```

```
webbrowser
webbrowser.open("boku.ac.at")
```

Typen von Werten sind entscheidend

```
>>> "Python " + 2019
Traceback (most recent call last):
  File "<pyshell>", line 1,    <module>
TypeError: can only concatenate str (    "int") to str
>>> 1 + "3"
Traceback (most recent call last):
  File "<pyshell>", line 1,    <module>
TypeError: unsupported operand type(s)     +: 'int'      'str'
```

Es gibt aber Konvertierungsfunktionen:

```
>>> "Python " + str(2019)
'Python 2019'
>>> 1 + int("3")
4
```

Stringfunktionen: `replace`

- Viele Objekte bieten Funktionen direkt an
- Aufruf durch `obj.funktion()` (im Gegensatz zu `min(1,2,3)`)

```
>>> "hello world".replace("world", "boku")
'hello boku'
```

AUFGABE

Bei `"IOOIO"` wurden die Buchstaben `"I"` und `"O"` statt `1` und `0` verwendet. Verwende `replace` um hier wieder die richtigen Ziffern einzusetzen und `int` um schließlich eine Zahl daraus zu machen.

Das ist übrigens in auch in einer Zeile möglich.

Stringfunktionen: `find`

Gibt Position von gesuchtem "Substring" zurück (oder `-1`):

```
>>> "hello world".find("wor")
6
>>> "hello world".find("work")
-1
```

Computer beginnen bei 0 zu zählen:

```
>>> "holzwirtschaft".find("h")
0
```

Stringfunktionen: `find`

```
>>> help("".find)
Help on built-   function find:

find(...) method of builtins.str instance
    S.find(sub[, start[, end]]) -> int

    Return the lowest index    S where substring sub    found,
    such that sub    contained within S[start:end].  Optional
    arguments start     end are interpreted      slice notation.

    Return -1 on failure.
```

```
>>> "holzwirtschaft".find("h", 4)
10
```

Stringfunktionen

- Es gibt noch viele weitere Stringfunktionen
- In der Praxis muss man ständig hier nachsehen:

```
>>> dir(str)
>>> help(str)
>>> help(str.find)
>>> help("".find)
```

https://docs.python.org/3.8/library/stdtypes.html#string-methods

AUFGABE

- Finde und verwende eine Stringfunktion, die unnötigte Leerzeichen links und rechts entfernt (`" world "`).
- Finde und verwende eine Stringfunktion, die einen String in Großbuchstaben umwandelt (`"world"` → `"WORLD"`).

Wahrheitswerte (`bool`)

Ja/Nein-Abfragen geben `True` oder `False` zurück

```
>>> "boku.ac.at".endswith(".ac.at")

>>> "orf.at".endswith(".ac.at")

>>>

>>> "123".isdecimal()

>>> "orf.at".isdecimal()
```

Typen herausfinden

`type` zeigt den Typ eines Ausdrucks an

```
>>> type("orf.at")
<class 'str'>
>>> type(    )
<class 'bool'>
>>> type("orf.at".isdecimal())
<class 'bool'>
>>> type(1 + 2)
<class 'int'>
>>> type(1 / 3)
<class 'float'>
>>> type("world".find)
<     '                          '>
```

Variablen (1/3)

- Variablen speichern Werte
- Können wie ihre Werte verwendet werden
- Können sich jederzeit ändern
- `=` ist Zuweisung, nicht mathematisches Gleichzeichen

```
>>> boku_address = "boku.ac.at"
>>> boku_address
'boku.ac.at'
>>> number_of_students = 15
>>> number_of_students
15
>>> number_of_students + 5
20
>>> number_of_students + 5
20
>>> number_of_students = number_of_students + 5
>>> number_of_students
20
>>> number_of_students + 5
25
```

Variablen (2/3)

- Erlaubte Namen:
  - Groß- und Kleinbuchstaben und `_`
  - keine Leerzeichen, kein `-`
- Richtline für Variablennamen in Python:
  snake case (`number_of_students`)
  (im Gegensatz zu camel case `numberOfStudents`)
- Rechts kann ein beliebiger Ausdruck stehen
- Name muss immer links stehen:

```
>>> 5 = number_of_students
File "<pyshell>", line 1
SyntaxError: can't assign to literal
```

Variablen (3/3)

- Neuer Wert kann neuen Typ haben
- Nicht unbedingt empfohlen!

```
>>> weight = 4.54
>>> weight + 3.2
7.74
>>> type(weight)
<class 'float'>
>>> weight = "heavy"
>>> weight + 3.2
Traceback (most recent call last):
  File "<pyshell>", line 1,    <module>
TypeError: can only concatenate str (    "float") to str
>>> type(weight)
<class 'str'>
```

Wert austauschen

AUFGABE

Schreibe ein Programm, das die Werte von 2 Variablen vertauscht:

```
>>> a = "apple"
>>> b = 2.3
[dein Programm hier]
>>> a
2.3
>>> b
'apple'
```

Hilfestellung hier unter "Exchange Program":
https://cscircles.cemc.uwaterloo.ca/1-variables/