Grundlagen

Richard Müller, Tom Felber

20. Oktober 2021

Python-Kurs

Gliederung

Über diesen Kurs

Über diesen Kurs

- 12 Kurseinheiten
- setzt grundlegende Programmierkenntnisse voraus
- Ressourcen
  - die Kursseite
  - offizielle Dokumentation
  - die Github-Seite für diesen Kurs
- Hinweis: SCM's sind hilfreich (git)

Der Python Interpreter

Der Python Interpreter

- Die zwei verbreitet verwendeten Python Versionen sind 2.7 und 3, wir werden 3 benutzen, da 2.7 nicht mehr unterstützt wird
- Python kann hier heruntergeladen und installiert werden oder mit dem Paketmanager eurer Wahl. (Das Paket sollte `python3` und `python3-dev` sein, außer unter Arch)
- Python funktioniert am besten unter UNIX (ist aber okay unter Windows)
- Den Interpreter startet man mit `python3` im Terminal oder mit `Python.exe`
- Der Interpreter stellt die volle Funktionalität von Python bereit, einschließlich dem Erstellen von Klassen und Funktionen

Python Scripte

Python Scripte

Editor empfohlen

- atom (benutzen wir im Kurs)

IDEs hilfreich bei größeren Projekten, Vorstellung gegen Ende des Kurses oder auf Anfrage

- PyCharm (free + professional für Studenten)

Python Scripte

Struktur

- Python Scripte sind Textdateien, die auf `.py` enden
- Python Packages sind Ordner mit einer `__init__.py` Datei (behandeln wir später)

Grundlagen der Sprache

Grundlagen der Sprache

Python ist eine schwach typisierte Scriptsprache (weakly typed scripting language). Es gibt Typen (anders als in JavaScript), aber Variablen haben keine festen Typen.

Beispiel (erzeugt keinen Fehler):

```
1 my_var = "Das ist ein String"
2 my_var = 42
3 my_var = 4.2
4 my_var = 4.222214124
5 my_var = True
6 my_var = None
```

builtin Datentypen:

Name	Funktion
`object`	Basistyp, alles erbt von `object`
`int`	Ganzzahl "beliebiger" Größe
`float`	Kommazahl "beliebiger" Größe
`bool`	Wahrheitswert (`True`, `False`)
`None`	Typ des `None`-Objektes
`str`	Zeichenkette
`type`	Grundtyp aller Typen (z.B. `int` ist eine Instanz von `int`)
`list`	standard Liste
`tuple`	unveränderbares n-Tupel
`set`	(mathematische) Menge von Objekten
`frozenset`	unveränderbare (mathematische) Menge von Objekten
`dict`	Hash-Map

Das erste Programm

Das erste Programm

Ein simples 'Hallo Welt'-Programm:

```
1 # normaler einzeiliger Kommentar
2 print('Hallo Welt!')
```

gute Konvention:

```
1 def my_function():
2     """
3     Docstring, aber nur zu Beginn einer
4     Funktions- oder Klassendefinition
5     """
6     print('Hallo Welt!')
7
8 if __name__ == '__main__':
9     my_function()
```

Das erste Programm

Wichtige Eigenschaften:

- Keine Semikolons
- Keine geschweiften Klammern für Codeblöcke
- Einrückungen zeigen Codeblöcke an
- Funktionsaufrufe immer mit runden Klammern
- Funktionen definieren mit
  `def <funktionsname>([parameter_liste, ...]):`
- Variablen mit der Struktur `__name__` sind spezielle Werte
  (gewöhnlich aus `builtin` oder Methoden von Standardtypen)

Operatoren

Operatoren

mathematisch `+`, `-`, `*`, `/`, `%`

vergleichend `<`, `>`, `<=`, `>=`, `==` (Wert gleich), `is` (gleiches Objekt/gleiche Referenz)

logisch `and`, `or`, `not` `(a && b) || (!c)` aus C oder Java entspricht `(a and b) or not c` in Python

bitweise `&`, `|`, `<<`, `>>`, `^` (xor), `~` (invertieren)

Accessoren `.` (für Methoden und Attribute), `[]` (für Datenstrukturen mit Index)

Namenskonvention

Namenskonvention

Klassen *PascalCase*, alles direkt zusammen, groß beginnend und jedes neue Wort groß

Variablen, Funktionen, Methoden *snake_case*, alles klein und Wörter mit Unterstrich getrennt
Merke: Da - ein Operator ist, ist es in Namen von Variablen, Funktionen etc. **nicht** zulässig (damit Python eine Kontextfreie Sprache ist)

protected Variablen, Funktionen, Methoden beginnen mit einem Unterstrich _ oder mit zweien __ für private

Merke Python hat kein Zugriffsmanagement. Die Regel mit dem Unterstrich ist nur eine Konvention um zu verhindern, dass andere Teile des Codes nutzen, der eine hohe Wahrscheinlichkeit hat in Zukunft verändert zu werden.

Strings

Strings - Grundlagen

- Der Typ eines Strings ist `str`.
- Strings sind in Python immutable (nicht veränderbar). Jede String Operation erzeugt einen neuen String.
- Ein String kann erzeugt werden mit einer Zeichenkette in Anführungszeichen, `''` oder `""` (beide sind äquivalent).
- rohe Strings mir dem Präfix `r`, `r"mystring"` oder `r'mystring'`
- Strings in Python3 sind UTF-8 encoded.

Strings - Verknüpfen

- Strings können durch Konkatenation verknüpft werden

```
1 'Hallo' + '_' + 'Welt' #  => 'Hallo_Welt'
```

- Listen, Tupel etc. von Strings können via ‘str.join‘ verknüpft werden

```
1 '_'.join(['Hallo', 'Welt']) #  => 'Hallo_Welt'
```

  Dabei ist der String, auf welchem die Methode aufgerufen wird, der Separator.

Strings - Formatierung

Wir wollen den String 'my string 4 vier' erzeugen.

```
# mit `str.format()`

'my string {} {}'.format(4, 'vier')
# in Reihenfolge der Argumente

'my string {number} {name}'.format(name='vier', number=4)`
# via Name, Reihenfolge egal

'my string {number} {}'.format('vier', number=4)
# oder beides kombiniert
```

Strings - Formatierung

Wir wollen den String 'my string 4 vier' erzeugen.

```
# und mit dem %-Operator

'my string %d %s' % (4, 'vier')
# in Reihenfolge

'my string %(number)d %(name)s' % {'number':4, 'name':'vier'}
# via Name
```

Strings - Formatierung

Wir wollen den String 'my string 4 vier' erzeugen.

```
# mit f strings

f'my string {4} {"vier"}'
# mit f vor dem String, und geschweiften Klammern"

f"my string {4} {'vier'}"
# oder so
```