Python Hackathon

Nur auf der vGen-Konferenz

Stefan, 04.02.2022

Einige Grundlagen

- Formatierung ist auch Semantik
  - Leerzeichen / Tabs egal, aber einheitlich
- Module, Virtual environment & pip
  - Für (fast) alles ein Modul vorhanden
  - Eigenes Modul erstellen: Datei (Name ist Modulname)
  - Suchen: https://pypi.org/ bzw. bei uns: apt search ….
- Python Interpreter (auch interaktiv nutzbar)

Einige Grundlagen

- Python Interpreter (auch interaktiv nutzbar)
  - Aufruf mit „python3“ (Version 3, wir machen kein Python 2 heute)
    - Ohne Parameter: Interaktiv
    - Mit Dateiname: Führe Datei aus
  - Linux: Shebang + chmod +x …
    - #!/usr/bin/python3

Sprache

- Variablen nicht deklarieren, sondern zuweisen
  - v = 45
  - a = [0, 1, 2, 3, v]

- Es gibt Typen, aber zunächst nicht sichtbar
  - Referenz: https://docs.python.org/3/library/stdtypes.html
  - Funktion type(…) nutzen

Sprache: Wichtige Typen

- Zahlen (int, float), Strings, boolean
- Liste: a = [1, 2, 3, ‚huhu‘, 456.4]
  - Geordnet und veränderbar
- Tupel: t = (1, 2, 3)
  - Geordnet und nicht veränderbar, Zugriff auch über t[0], ….
- Set: a = {1, 2, 3, 4}
  - Ungeordnet und eindeutig
- Dictionary: d = {‚code‘: 3, ‚value‘:’three’}
  - Zugriff mit: d[‚code‘], d[‚value‘]

Sprache: Listen

- Wie Arrays
- Mehrere Dimensionen möglich
- Aber: initialisieren mit Werten, keine „leeren“ Listen möglich

```
a = [0 for x in range(5)]
print(a)

[0, 0, 0, 0, 0]
```

Sprache: Kontrollfluss

- Grundsätzlich: Einrückung ergibt semantischen Block
- Funktion

```
def start():
    print('Starting')
    return True
```

- Schleife

```
for i in range(0, 10):
    print(i)
```

- Gleiches Prinzip für while, if, …

Sprache: Kontrollfluss

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

ready = True

if i == 0 or i < -5 or i != 5:
    print(0)
elif i == 10 and not ready:
    print(10)
else:
    print('Was anderes')
```

Sprache: Klassen

- Auch Klassen existieren, Nutzung aber nicht so häufig, wie bei Java, …

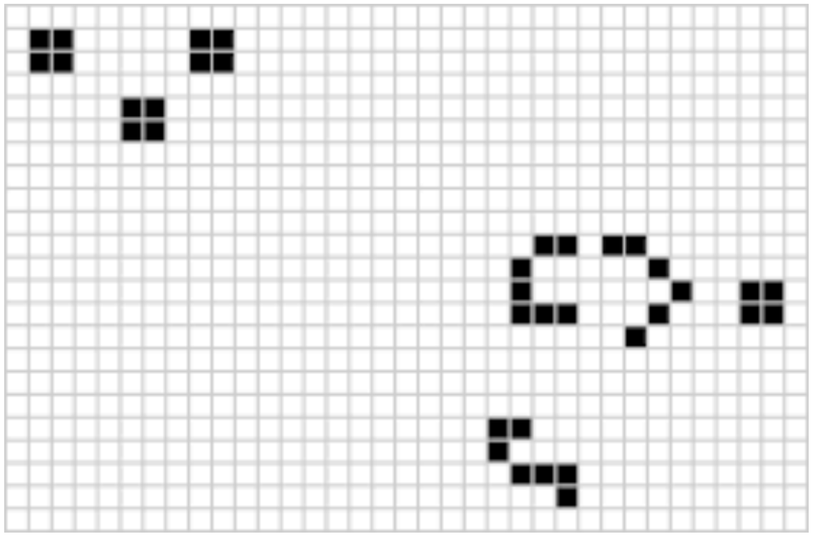
```
class MyClass:
    def __init__(self):
        return

    def member_function(self, a):
        print(a)

c = MyClass()
c.member_function('Hallo Klasse')
```

Umgebung
IDE, Env, …

- IDE: PyCharm (Visual Studio Code, IDLE, …)
  - Legt bei Bedarf direkt Umgebung an (virtual env)
    - Erlaubt eigene Versionen von Python und Modulen pro Projekt
    - Manuell: siehe auch https://docs.python.org/3/library/venv.html
    -

Conways Game of Life

Gemeinsame Übung



Conways Game of Life

Mit pygame

- Simulation von Individuen (Zellen)
- Regeln
  - Zellen werden geboren, wenn sie drei lebende Nachbarn haben
  - Zellen sterben, wenn sie weniger als zwei lebende Nachbarn haben (Einsamkeit)
  - Zellen sterben, wenn sie mehr als drei lebende Nachbarn haben

  - Siehe auch: https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

Conways Game of Life

Mit numpy

- Verbesserte Version: manuelle Listen-Behandlung sperrig
- Lösung: numpy

Modul: numpy

- Eigene Klassen für Listen / Arrays, viele nützliche Funktionen

Jupyter Notebook
Web IDE

- Web IDE für Python-Projekte
  - Fokus auf mathematischen Anwendungen / Analysen / Auswertungen / AI / ML
  - Direkte Ergebnisse im Browser
  - Starten mit „jupyter-notebook“

Daten Sammeln
CSV

- CSV lesen mit pandas
-

Daten Sammeln

Scrapy, BS4

- Scrapy

- Beautiful Soup 4

Diagramme: matplotlib

Fokus: Mathematische Darstellungen

- Kurven