Python Basics

Martin Dröge

26. Februar 2021

Einstieg

Vorstellungsrunde und Erwartungen

Zeitplan

- Teil 1 | 9-11 Uhr:
  - Python Basics
- Teil 2 | 11-13 Uhr:
  - Anwendungsbeispiele
    - Text Mining
    - Web Scraping

Ziele

- Einstiegsschwelle überwinden
- Neugierig machen
- Einblicke in Python geben
- Eigene Skripte schreiben

Grenzen

- Sehr knapper Zeitplan
- Nur Grundlagen werden angesprochen
- Kein vollständiger Programmierkurs

Fragen sind jederzeit erwünscht!

Vorgehen

- Kurze Inputs
- Kleine Skripte gemeinsam schreiben

Zwei „Lern-Stränge“

1. Python-Skripte coden
2. Jupyter Notebooks, Deepnote und Google Colab kennenlernen

Block 1: Historie, Konzepte, Einsatz

Warum Python?

- einfach zu lernen
- einfach zu lesen
- riesige Community
- viele Bibliotheken verfügbar
- laut TIOBE im Jahr 2020 auf Platz 3
- Open Source Software

Entwicklung

- zu Beginn der 1990er-Jahre von Guido van Rossum entwickelt
- 1994 erschien die Version 1.0
- ursprünglicher Zweck: Programmieren vermitteln
- Name geht nicht auf die Schlange, sondern auf Monty Python zurück
- Dezember 2008: Version 3.0 -> vollständig auf Unicode umgestellt
- aktuelle Version Python 3.9.1

Python als Skriptsprache

- interpretiert
- höhere Programmiersprache
- Python-Skripte erkennbar an der Endung: `skript_name.py`
- Jupyter Notebooks erkennbar an der Endung: `notebook_name.ipynb`

Einsatzgebiete I

- Universitäten und Forschungseinrichtungen
- Technologie-Branche
- Industrie
- Data Science

Einsatzgebiete II

- Data and Text Mining
- Daten-Analyse
- Visualisierung
- Web Entwicklung
- System-Administration
- Rapid Prototyping

Hilfe zur Selbsthilfe

- Python Tutorial
- learnpython.org
- Python Dokumentation
- Python Community
- Python Forum
- Stack Overflow

Block 2: Erste Schritte

Jupyter Notebook unter Anaconda

- Anaconda sollte vorab installiert sein!
- https://www.anaconda.com

lokales Arbeitsverzeichnis erstellen

- Neuen Ordner anlegen
- Umbenennen in `python-workshop`
- Speicherort merken

Jupyter Notebook starten

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- `Jupyter Notebook` anklicken
- Zum Ordner `python-workshop` navigieren

Neues Jupyter Notebook

- Rechts oben auf 'New' klicken und Python3 auswählen
- Links 'File' auswählen und 'Rename' anklicken
- Notebook umbenennen in `python-workshop-notebook`

Notebook unter Google Colaboratory

- mit Google Account angemeldet sein
- Vorteil:
  - kein Setup und keine Installationen notwendig
  - Notebooks können über einen Link geteilt werden
  - Blick aus der Ferne in das Notebook wird möglich

Google Colab starten

- https://colab.research.google.com aufrufen
- unten rechts `NEUES NOTEBOOK` auswählen
- Notebook umbenennen in `*NAME*_python-workshop-notebook`

Jupyter Notebook im Browser

- https://jupyter.org/try aufrufen
- 'Try Classic Notebook' anklicken
- Links 'File', dann 'New Notebook' und 'Python3' anwählen
- Links 'File' auswählen und 'Rename' anklicken
- Notebook umbenennen in `python-workshop-notebook`

Hallo Welt!

```
print("Hallo Welt!")
```

Jupyter Notebook Short Cuts

- STRG + Enter = Zelle ausführen
- SHIFT + Enter = Zelle ausführen und neue Zelle einfügen
- ESC = Wechsel in den Command Mode
- Enter = Wechsel in den Edit Mode
- B = neue Zelle unten einfügen
- M = Zelle in Markdown umwandeln

Addieren

5 + 2

3 + 6

42 + 1337

Substrahieren

5 - 2

7 - 4

1337 - 42

Multiplizieren

7 * 5

6 * 3

6 * 7

Dividieren

15 / 3

42 / 7

1337 / 12

Ganzzahl und Gleitkommazahl

- **Ganzzahl**

Integer, int(): 8, 42, 1337

- **Gleitkommazahl**

Float, float(): 1.5, 8.7, 123.45

Block 3: Datentypen

Zeichenketten / Strings

```
print("Hallo Welt!")
```

Konkatenieren

```
print("Hallo " + "Welt!")

print("Digital History Tagung " + "2021")

print("foo" * 2 + "bar " * 3)
```

Variablen

```
event = "Digital History Tagung"
year = 2021
```

f-Strings

```
print(f"Die {event} findet {year} statt.")

print("Die {} findet {} statt.".format(event, year))
```

Funktionen und Methoden bei Strings

```
len(event)

len("Digital History Tagung")

event.lower()

"Digital History Tagung".upper()
```

Listen / Lists

```
list_1 = [1, 2, 3, 4]

list_2 = ["Banane", "Apfel", "Birne"]

list_3 = [17, "Obst", 23, "Ball"]

list_4 = [[1, 2, 3,], ["a", "b", "c"], ["Hallo", "Welt"]]
```

Zuordnungen / Dictionaries

```
dict_1 = {"Banane": 5,
          "Apfel": 7,
          "Birne": 9}

dict_2 = {"Vorname": "Monty",
          "Nachname": "Python",
          "Alter": 48}
```

Mengen / Sets

```
menge = {1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 5}

print(menge)

type(menge)
```

Tuple

```
word_freq = ("Wort A", 42)

type(word_freq)
```

Indexieren von Strings

```
"Digital History Tagung"[1]

"Digital History Tagung"[0]

"Digital History Tagung"[8]
```

Slicing

```
"Digital History Tagung"[0:10]
```

```
"Digital History Tagung"[3:6]
```

Indexieren von Listen

```
list_2 = ["Banane", "Apfel", "Birne"]
```

```
list_2[2]
```

```
list_2[0]
```

Funktionen und Methoden bei Listen

```
len(list_2)
```

```
list_2.append("Ananas")
```

```
list_2.pop()
```

Zugriff auf Dictionaries

```
dict_1 = {"Banane": 5, "Apfel": 7, "Birne": 9}
```

```
dict_1["Banane"]
```

```
dict_1.keys()
```

```
dict_1.values()
```

```
sum(dict_1.values())
```

Logische Ausdrücke

```
5 > 3
```

```
5 > 7
```

```
"Mauer" == "Haus"
```

```
"Mauer" != "Haus"
```

Logische Ausdrücke: and, or

```
5 > 3 and "Mauer" != "Haus"
```

```
5 > 3 and "Mauer" == "Haus"
```

```
5 > 3 or "Mauer" == "Haus"
```

```
5 > 7 or "Mauer" == "Haus"
```

Block 4: error handling und Kontrollstrukturen

User-Input Skript

```
# Eingabe Name
name = input("Wie heißt du? >>>")
# Eingabe Alter
alter = input("Wie alt bist du? >>>")
# Eingabe Wohnort
ort = input("Wo wohnst du? >>>")
jahr = 2021 - int(alter)
# Ausgabe
print(f"""
\nHallo {name}, schön, dass du da bist!\n
Du bist {jahr} geboren.\n
{ort} ist der beste Ort auf dem Planeten.""")
```

Fehlerbehandlung / error handling

```
print("Hallo Welt!"
```

```
File "<ipython-input-5-e1aa54892f20>", line 1
    print("Hallo Welt!"
                       ^
SyntaxError: unexpected EOF while parsing
```

Fehlerbehandlung / error handling

```
for x in range(10)

File "<ipython-input-7-a12e79ce3754>", line 1
    for x in range(10)
                      ^
SyntaxError: invalid syntax
```

Fehlerbehandlung / error handling

```
for x in range(10):
print(x)

File "<ipython-input-8-790af6e00d9b>", line 2
    print(x)
        ^
IndentationError: expected an indented block
```

help() Function

```
help(len)
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

Kontrollstrukturen

For-Schleife

```
for x in range(10):
    print(x)
```

While-Schleife

```
x = 10
while x > 0:
    print(x)
    x -= 1
```

If-Anweisung

```python
for x in range(1, 11):
    if x % 2 == 0:
        print(f"{x} ist eine gerade Zahl.")
    else:
        print(f"{x} ist eine ungerade Zahl.")
```

Dateien lesen 1

```python
f = open("DATEI.txt", "r", encoding="utf-8")
text = f.read()
f.close()
```

Dateien lesen 2

```python
with open("DATEI.txt", "r", encoding="utf-8") as f:
    text = f.read()
```

I/O Parameter

- "r" = read / lesen
- "w" = write / schreiben
- "a" = append / anhängen

Block 5: Pseudo-Code

algorithmisches Denken

- Problem in einzelne Schritte zerlegen
- Schritte ausformulieren
- einzelne Bauteile zusammenstellen

Pseudo Code

```
TEXT =  "text"

LOOP OVER TEXT

    IF ELEMENT IS VOWEL
```

```
        PRINT ELEMENT
```

Vorteil

- Vorgehen wird klarer
- Konzepte verstehen
- Übertragbar auf andere Programmiersprachen

Problemstellung

Für die Zahlen zwischen 1 und 100 soll:

- FizzBuzz ausgegeben werden, wenn die Zahl durch 3 und 5 teilbar ist
- Fizz ausgegeben werden, wenn die Zahl durch 3 teilbar ist
- Buzz ausgegeben werden, wenn die Zahl durch 5 teilbar ist

Pseudo-Code Beispiel

```
FOR i from 1 TO 100 DO
    IF i is divisible by 3 AND i is divisible by 5 THEN
        OUTPUT "FizzBuzz"
    ELSE IF i is divisible by 3 THEN
        OUTPUT "Fizz"
    ELSE IF i is divisible by 5 THEN
        OUTPUT "Buzz"
    ELSE
        OUTPUT i
```

Pseudo-Code in Python

```python
for i in range(1,100):
    if i % 3 == 0 and i % 5 == 0:
        print('FizzBuzz')
    elif i % 3 == 0:
        print('Fizz')
    elif i % 5 == 0:
        print('Buzz')
    else:
        print(i)
```

Block 6: Funktionen Textprocessing

Text-Datei runterladen

```
import requests

gg_raw_url = "https://raw.githubusercontent.com/levinalex/deutsche_verfassungen/master/grund
response = requests.get(gg_raw_url)
grundgesetz = response.text

grundgesetz[:1000]
```

Text-Datei speichern

```
with open('grundgesetz.txt', 'w', encoding='utf-8') as f:
    f.write(grundgesetz)
```

Kleinschreibung vereinheitlichen

```
grundgesetz = grundgesetz.lower()
print(grundgesetz[:250])
```

Zeichensetzung entfernen

```
import string

def remove_punctuation(text):
    punctuation = string.punctuation
    for marker in punctuation:
        text = text.replace(marker, "")
    return text

grundgesetz = remove_punctuation(grundgesetz)
print(grundgesetz[:250])
```

Liste erstellen

```
grundgesetz_words = grundgesetz.split()

print("Anzahl aller Worte des Textes: ")
print(len(grundgesetz_words)))
```

```
print("=======")
print(grundgesetz_words[:25])
```

Zählen eines bestimmten Worts

```
def count_item_in_text(item_to_count, list_to_search):
    number_of_hits = 0
    for item in list_to_search:
        if item == item_to_count:
            number_of_hits += 1
    return number_of_hits

print(count_item_in_text("freiheit", grundgesetz_words))
```

Alle Wörter zählen mit Hilfe eines Dictionarys

```
def counter_dict(list_to_search):
    counts = {}
    for word in list_to_search:
        if word in counts:
            counts[word] = counts[word] + 1
        else:
            counts[word] = 1
    return counts

print(counter_dict(grundgesetz_words))
```

Worthäufigkeiten sortieren

```
def freq_sort(list_to_search):
    counts = counter_dict(list_to_search)
    counts = [(counts[key], key) for key in counts]
    counts.sort()
    counts.reverse()
    return counts

print(freq_sort(grundgesetz_words)[:25])
```

Entfernen von Stoppwörtern

```
import requests
```

```
def remove_stopwords(list_to_search):
    stopword_url = "http://members.unine.ch/jacques.savoy/clef/germanST.txt"
    response = requests.get(stopword_url)
    stopwords = response.text
    stopwords = stopwords.split()
    return [w for w in list_to_search if w not in stopwords]

print(remove_stopwords(grundgesetz_words)[:25])
```

Funktionsaufrufe

```
grundgesetz = grundgesetz.lower()
grundgesetz = remove_punctuation(grundgesetz)
grundgesetz_words = grundgesetz.split()
grundgesetz_words = remove_stopwords(grundgesetz_words)
print(freq_count(grundgesetz_words)[:25])
```

Block 7: NLTK

collections

```
from collections import Counter

freq = Counter(grundgesetz_words)
```

most common

```
print(freq.most_common(25))
```

NLTK

```
import nltk

with open('grundgesetz.txt', 'r' encoding='utf-8') as infile:
    text_raw = infile.read()

text = text_raw.lower()
text = text.split()
text = nltk.Text(text)
```

concordance

```
text.concordance("freiheit")
```

similar

```
text.similar("freiheit")
```

dispersion plot

```
text.dispersion_plot(["artikel", "gesetz", "freiheit"])
```

Block 8: Web Scraping

Ziel des Web Scraping

- Posts vom Blog der AG Digitale Geschichtswissenschaft bei hypotheses.org
- https://digigw.hypotheses.org/

Zutaten für das Skript

- Webseite aufrufen
- html erfassen
- html parsen
- Text extrahieren
- Text in Datei speichern
- Datei benennen

Inspektion der Webseite

- Gibt es eine serielle URL?
- Wie ist diese aufgebaut?
- In welchem html-div findet sich Text?

serielle URL

https://digigw.hypotheses.org/3653

Webseiten Inspektor

STRG + SHIFT + I

Bausteine des Skripts I

- Import der benötigten Module und Bibliotheken
- URL-Root einrichten Variablen benennen
- Startseite bzw. Indexseite anfragen
- html in Variable speichern
- IDs der Blogposts aus html extrahieren
- IDs der Blogposts in Liste speichern
- for-Schleife zur wiederholten Anwendung

Bausteine des Skripts II

- aus der Liste der IDs URL zusammensetzen und anfragen
- html in Variable speichern
- Text des Posts aus html extrahieren
- Text in Datei speichern
- for-Schleife zur wiederholten Anwendung

Import der Module

```
import request
import time
from bs4 import BeautifulSoup
```

URL-Root und Variablen

```
url_root = 'https://digigw.hypotheses.org/'
blog_ID_list = []
article_length_list = []
```

Schleife IDs

```
for x in range(1,19):
    url = url_root + 'page/' + str(x)
    response = requests.get(url)
    response = response.text
    soup = BeautifulSoup(response, 'html.parser')
```

```
    blog_IDs = soup.find_all('h2',
                             {'class': 'entry-title'})
    for element in blog_IDs:
        blog_ID = element.find('a')['href']
        blog_ID = blog_ID.split('/')
        blog_ID = blog_ID[3]
        blog_ID_list.append(blog_ID)
    time.sleep(5)
```

Schleife Text

```
for x in blog_ID_list:
    try:
        response = requests.get(url_root + str(x))
        response = response.text
        soup = BeautifulSoup(response, "html.parser")
        article_text = soup.find(
                        attrs={"class": "entry-content"}
                        ).text.strip()
        article_length_list.append(len(article_text))
        with open(r'./data-blog-posts/digigw-blog_'
                    + str(x) + '.txt',
                    'w', encoding='utf-8') as infile:
            infile.write(article_text)
        time.sleep(8)
    except:
        continue
```

Ausgabe Ergebnisse

```
print(blog_ID_list)
print('Gesamtanzahl der Blogbeiträge: ')
print(str(len(blog_ID_list)))
print('Anzahl der Zeichen des längsten Blogbeitrags: ')
print(str(max(article_length_list)))
print('Anzahl der Zeichen des kürzsten Blogbeitrags: ')
print(str(min(article_length_list)))
print('Anzahl der Zeichen der Blogbeiträge im Durchschnit: ')
print(str(sum(article_length_list) / len(article_length)))
```

Block 9: Hilfe zur Selbsthilfe

Tutorials

learnpython.org

https://realpython.com/

https://docs.python.org/3/tutorial/

https://www.python-kurs.eu/

https://programminghistorian.org/lessons/?topic=python

Links

https://www.python.org/doc/

https://www.python.org/community/

https://www.python-forum.de/

https://stackoverflow.com/

Bücher: Online

https://automatetheboringstuff.com/

https://www.nltk.org/book/

offene Fragen

Feedback-Runde

Zum Abschluss

What people think programming is vs. how it actually is