Block 3: Datentypen

IPython-REPL II

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- IPython anklicken

Navigieren im Dateiensystem

```
In [1]: pwd
```

```
Out[1]: 'C:/Users/Martin/Documents'
```

```
In [2]: cd ..
```

```
C:/Users/Martin
```

```
In [3]: cd Documents
```

```
C:/Users/Martin/Documents
```

Magische Befehle

```
%magic
```

```
%cpaste
```

Zeichenketten / Strings

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

Konkatenieren

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

```
print("Historikertag " + "2018")
```

```
print("Hallo " * 2 + "Historikertag")
```

Variablen

```
event = "Historikertag"
year = 2018
```

Interpolieren

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

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

Funktionen und Methoden bei Strings

```
len(event)
```

```
len("Historikertag")
```

```
event.lower()
```

```
"Historikertag".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": "Jan", "Nachname": "Meier", "Alter": 48}
```

Mengen / Sets

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

```
print(menge)
```

```
type(menge)
```

Tuple

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

```
type(word_freq)
```

Indexieren von Strings

```
"Historikertag"[1]
```

```
"Historikertag"[0]
```

```
"Historikertag"[8]
```

Slicing

```
"Historikertag"[0:10]
```

```
"Historikertag"[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: kleine Skripte

VS Code einrichten

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- Anaconda Navigator anklicken
- VS Code installieren

VS Code starten

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- Anaconda-Prompt anklicken

- ‘code’ eingeben

Python Extension einbinden

- ganz links viertes Symbol von oben klicken
- im Suchfeld ‘Python’ eingeben
- ersten Treffer auswählen und installieren
- Code neu starten

Python-Datei anlegen

- speichern unter: Dateiname eingeben
- Dateiendung: .py anhängen

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 = 2018 - 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.""")
```

Skript aufrufen

- STRG + ö
- ins Verzeichnis navigieren
- python DATEINAME.py

Anweisungsaufbau und Blöcke

ANWEISUNGSKOPF:**DOPPELPUNKT**

EINRÜCKUNG ANWEISUNG

Fehlerbehandlung / error handling

```
for x in range
```

```
  File "<ipython-input-1-a89b35aad551>", line 1
    for x in range
                  ^
SyntaxError: invalid syntax
```

help() Function

```
help(len)
```

```
Help on built-in function len in module builtins:

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

Juypter Notebooks einrichten

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- Anaconda Navigator anklicken
- Jupyter Lab installieren

Jupyter Lab starten

- Windows-Startmenu aufrufen
- unter 'A': Anaconda auswählen
- Anaconda-Prompt anklicken
- 'juypter lab' eingeben

Jupyter Notebook starten

- ganz links 'Files' klicken
- in das Arbeitsverzeichnis navigieren
- im Launcher Notebook Python 3 auswählen

Jupyter Notebook Short Cuts

- STRG + Eingabe: = Zelle ausführen
- B = neue Zelle unten einfügen

Kontrollstrukturen

For-Schleife

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

While-Schleife

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

If-Anweisung

```
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

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

Dateien lesen 2

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

I/O Parameter

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

block5-hsozkult

September 24, 2018

1 Web Scraping von Hsozkult

Das Ziel besteht darin, eine Suchanfrage an Hsozkult zu schicken und die erste Rezension, die wir erhalten, in einer Text-Datei abzuspeichern.

Als erstes binden wir die benötigten Bibliotheken ein:

```
In [1]: import re
        from urllib.request import urlopen
        from bs4 import BeautifulSoup
```

`re` stellt Funktionen für sogenannte 'Reguläre Ausdrücke' zur Verfügung. `urllib` ist eine Standardbibliothek für den Umgang mit dem Internetprotokoll `http`. `bs4` steht für *BeautifulSoup* – die beste Bibliothek, um HTML zu parsen.

AnschlieSSend bauen wir uns eine URL zusammen: Sie besteht aus der eigentlichen Domain, aus dem Query-String, über den sich Suchanfragen ausführen lassen, und aus den Suchbegriffen.

```
In [2]: base_url = "http://www.hsozkult.de"
        query_string = "/searching/page?q="
        search_string = "rezension grundwissenschaft"
        search_string = search_string.replace(" ", "+")
        url = base_url + query_string + search_string
```

Da keine Leerzeichen in einer URL vorhanden sein dürfen, müssen wir im Such-String alle Leerzeichen durch '+' ersetzen. Danach wird die URL aus den einzelnen Bestandteilen zusammengesetzt.

Als nächstes senden wir mithilfe von `urlopen` eine Anfrage an Hsozkult. Die Antwort des Servers übergeben wir an *BeautifulSoup*, lassen es durch einen HTML-Parser verarbeiten und speichern das Ergebnis in der Variable `bs` ab.

```
In [3]: search = urlopen(url)
        bs = BeautifulSoup(search, 'html.parser')
```

Daraufhin müssen wir einen Blick auf eine entsprechende Seite von Hsozkult werfen. Dabei stellen wir fest, dass die Tabelle mit den einzelnen Suchergebnissen sich in einem `div`-Kontainer der `class` "hfn-list-itemtitle" befindet. Aus der Ergebnis-Liste nehmen wir uns den ersten Treffern vor.

```
In [4]: results = bs.find_all('div', {'class': 'hfn-list-itemtitle'})

        first_hit = results[0].find('a')['href']
        first_hit

Out[4]: '/searching/id/rezbuecher-28479?title=t-kahlert-unternehmungen-grossen-stils&q=rezensio
```

Wir suchen in diesem ersten Treffer nach einem Link <a> mit dem Attribut `href`, lesen das aus und speichern es in der Variable `first_hit` ab. Damit wissen wir, hinter welchem Link sich die erste gefundene Rezension verbirgt.

Wir bauen anschlieSSend erneut eine URL zusammen, diesmal aus der `base_url` und unserem `first_hit`. Da sich darin aber ein Leerzeichen befindet, müssen wir es wieder durch den entsprechenden Escape-Code '%20' ersetzen.

Haben wir das erledigt, schicken wir erneute eine Anfrage an Hsozkult, diesmal für die Rezension, und parsen das Ganze mit *BeautifulSoup*.

```
In [5]: link_url = base_url + first_hit
        link_url = link_url.replace(" ", "%20")

        review_html = urlopen(link_url)
        bs_review = BeautifulSoup(review_html, 'html.parser')
```

Unser Ziel ist es, den eigentlichen Text der Rezension zusammen mit den dazugehörigen bibliographischen Metadaten in einer Datei zu speichern. Um mit *BeautifulSoup* auf die einschlägigen HTML-Tags zugreifen zu können, werfen wir einen Blick auf den Quelltext der Seite. Auf diese Weise stellen wir fest, dass sich alle bibliographischen Metadaten in einer Tabelle jeweils in einem `div` mit dem Klassen-Attribut `class="hfn-item-metarow"` befinden. Eine Spalte gibt Auskunft darüber, um welchen Datentyp es sich handelt, also bspw. 'Autor(en)', 'Titel', 'Erschienen' etc. Die andere Spalte enthält die entsprechenden Angaben. Es bietet sich daher an, anhand dieser Angaben ein *Dictionary* zu erstellen.

Dafür lesen wir jede Zeile der Tabelle aus und übergeben die Werte einem *Dictionary* namens `meta_data`. Wir interessieren uns dabei nur für die ausgegebenen Texte innerhalb der *Tags*. Und wir entfernen den unnötigen *White space*. Wenn eine Spalte keine Angaben enthält, übernehmen wir sie auch nicht ins *Dictionary*.

```
In [6]: meta_data = {}

        for key, val in bs_review.find_all('div', {'class': 'hfn-item-metarow'}):
            k = key.get_text()
            k = k.strip()
            v = val.get_text()
            v = v.strip()
            v = re.sub(r"[\n\t]+", " ", v)
            if k != '':
                meta_data[k] = v

        meta_data
```

```
Out[6]: {'Autor(en)': 'Kahlert, Torsten',
         'Erschienen': 'Berlin  2017: be.bra Verlag',
         'ISBN': '978-3-95410-089-7',
         'Preis': ' 40,00',
         'Titel': 'Unternehmungen groen Stils.  Wissenschaftsorganisation, Objektivität und His
         'Umfang': '384 S.'}
```

Nun kümmern wir uns um den Rezensenten. Die Angaben dazu verstecken sich im `div` mit der Klasse `hfn-item-creator`. Auch hier interessieren wir uns nur für den Text. Das Institut etc. spielt für uns aber keine Rolle. Wir zerlegen also den String `review_author` an den Kommas und greifen nur auf den ersten Abschnitt der geschaffenen Liste zurück.

```
In [7]: review_author = bs_review.find('div', {'class': 'hfn-item-creator'})
        review_author = review_author.get_text()
        review_author = review_author.strip().split(',')[0]

        review_author

Out[7]: 'Jan Ruhkopf'
```

Jetzt lesen wir den eigentlichen Text der Besprechung aus. In dem `div` mit der Klasse `hfn-item-fulltext` befinden sich gleich mehrere Paragraphen. Diese suchen wir mit zwei `find`- bzw. `find_all`-Methoden des *BeautifulSoup*-Objekts. Das Ergebnis ist eine Liste mit den einzelnen Paragraphen, die wir in der Variable `review` abspeichern. Danach nutzen wir eine besondere Python-Konstruktion, die *list comprehension*. Mit ihrer Hilfe erstellen wir eine neue Liste `review_content`, in der der Text der einzelnen Paragraphen-*tags* hinterlegt wird.

```
In [8]: review = bs_review.find('div', {'class': 'hfn-item-fulltext'}).find_all('p')

        review_content = [paragraph.get_text() for paragraph in review]
```

Zu guter letzt speichern wir das alles in der Datei `rezension.txt` ab. Wir öffen es mit der Funktion `with open(...) as x:`, die sicherstellt, dass das *File*-Objekt am Ende auch wieder geschlossen wird.

In dem Block unterscheiden wir, ob es sich um einen Sammelband handelt oder um eine Monographie. Dementsprechend formatieren wir den String, den wir durch Interpolation aus den Metadaten und `review_author` zusammensetzen. Danach kommen zwei *newlines* und der Rezensionstext.

```
In [9]: with open('rezension.txt', 'w', encoding='utf-8') as f:
            if 'Hrsg. v.' in meta_data.keys():
                f.write("{}: Rezension von: {} (Hg.): {}, {}.".format(
                    review_author,
                    meta_data['Hrsg. v.'],
                    meta_data['Titel'],
                    meta_data['Erschienen']))
            else:
                f.write("{}: Rezension von: {}: {}, {}.".format(
                    review_author,
```

```
                meta_data['Autor(en)'],
                meta_data['Titel'],
                meta_data['Erschienen']))
        f.write("\n\n")
        f.write("\n".join(review_content))
```

Fertig ist unser Hsozkult-Scraper!

block6-anno-scraper

September 20, 2018

1 Block 6 Web Scraping Zeitungsportal Anno

URL: Übersicht der verfügbaren Zeitungen:

- http://anno.onb.ac.at/alph_list.htm

Das Skript kann verschiedene txt-Dateien aus dem österreichischen Zeitungsportal Anno herunter. Die txt-Dateien sind per OCR erfasst; Erkennungsfehler sind reichlich vorhanden.

1.1 Ermittlung der URL

- Zeitung auswählen
- Jahresübersicht auswählen
- Ausgabe auswählen
- Anzeige als txt auswählen
- am Ende der URL die Seitenzahl durch x ersetzen; auf diese Weise wird die gesamte Ausgabe ausgewählt
- in der URL findet sich ein Datumformat in der Form yyyymmdd; auf diese Weise kann in einer Schleife das Datum genutzt werden

1.2 Das Skript

```
In [ ]: import requests
        import time
        import pandas as pd
        from datetime import date
```

URL

```
In [ ]: url_root = "http://anno.onb.ac.at/cgi-content/annoshow?text=dar|" # URL der Zeitung ei
        url_root_2 = "|x"  # steht für alle Seiten einer Ausgabe
        name_zeitung = "die-arbeit"
```

Beginn und Ende der Schleife im Datumsformat

```
In [ ]: start_date = date(1886, 1, 1) # Start des Zeitraums
        end_date = date(1886, 1, 22) # Ende des Zeitraums
```

Mit Pandas ein date_range-Objekt erstellen

```
In [ ]: daterange = pd.date_range(start_date, end_date)
```

Über das date_range-Objekt iterieren

```
In [ ]: for date in daterange:

            date_id = date.strftime("%Y%m%d")

            response = requests.get(url_root + date_id + url_root_2)
            text = response.text

            print(date_id)

            if len(text)!= 0:

                # speichern der einzelnen Ausgabe
                with open(date_id + "-" + name_zeitung + ".txt", "w", encoding="utf-8") as file
                    file.write(text)

                # speichern eines gesamten Jahrgangs
                with open("1886-jahrgang-" + name_zeitung + ".txt", "a", encoding="utf-8") as f
                    file.write(text + "\n")

            time.sleep(2)
```

block6-funktionen-textpreprocessing

September 20, 2018

1 Block 6: eigene Funktionen schreiben - Textpreprocessing

1.1 Text-Datei einlesen

```
In [ ]: with open("DATEINAME.txt", encoding="utf-8") as infile:
            rezension = infile.read()
```

1.2 Kleinschreibung vereinheitlichen

```
In [ ]: def text_lower(text):
            return text.lower()

        rezension = text_lower(rezension)
        print(rezension[:250])
```

1.2.1 Punktation entfernen

```
In [ ]: import string

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

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

1.3 Liste erstellen

```
In [ ]: rezension_words = rezension.split()
```

```
In [ ]: print("Anzahl aller Worte des Textes: ", (len(rezension_words)))
        print("=======")
        print(rezension_words[:25])
```

1.4 Zählen eines bestimmten Worts

```
In [ ]: 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("information", rezension_words))
```

1.5 Alle Wörter zählen mit Hilfe eines Dictionarys

```
In [ ]: 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(rezension_words))
```

1.6 Worthäufigkeiten sortieren

```
In [ ]: 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(rezension_words)[:25])
```

1.7 Entfernen von Stoppwörtern

```
In [ ]: 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(rezension_words))
```

1.8 Funktionsaufrufe

In [ ]:
```
rezension = text_lower(rezension)
rezension = remove_punctuation(rezension)
rezension_words = rezension.split()
rezension_words = remove_stopwords(rezension_words)
print(freq_count(rezension_words)[:25])
```

block6b-nltk-textprocessing

September 24, 2018

1 Block 6b: NLTK

```
In [ ]: import collections

        freq = collections.Counter(text)

In [ ]: print(freq.most_common(25))

In [ ]: import string

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

In [ ]: import nltk

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

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

In [ ]: text.concordance("freiheit")

In [ ]: text.similar("freiheit")

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

1.1 NLTK-Beispiel

(aus: http://www.nltk.org/book/ch02.html)

```
In [ ]: import nltk
        from nltk.corpus import inaugural

        cfd = nltk.ConditionalFreqDist(
                  (target, fileid[:4])
                  for fileid in inaugural.fileids()
                    for w in inaugural.words(fileid)
                       for target in ['america', 'citizen']
                           if w.lower().startswith(target))
        cfd.plot()
```