Python für Linguisten

Dozentin: Wiebke Petersen & Co-Dozentin: Esther Seyffarth

Weiterführende Packages in Python

Pakete in Python

- Bisher haben wir im Unterricht nur Standardfunktionen und mitgelieferte Pakete von Python verwendet.
- Standardfunktionen sind ohne weiteres verfügbar, wie z.B. `sort()` oder `print()`.
- Mitgelieferte Pakete sind Sammlungen von Modulen, die bestimmte Aufgabenbereiche abdecken. Wir haben schon das Paket für reguläre Ausdrücke kennengelernt:

```
1 >>> import re
2 >>> pattern = re.compile("...")
```

- Man kann auch selbstgeschriebene Module importieren, wenn man ein Projekt bearbeitet, das sich in verschiedene Teilbereiche aufteilen lässt (wie z.B. in einigen Projektgruppen im Seminar).
- Wenn man versucht, ein Modul zu importieren, wird zunächst das Verzeichnis mit den Standardpaketen durchsucht (`<Pythonpath>/Lib/` und `<Pythonpath>/Lib/site-packages/`). Erst danach wird das aktuelle Arbeitsverzeichnis durchsucht!

Pakete installieren

- Selbstgeschriebene Module, die wiederkehrende Aufgaben erfüllen, kann man auch als Paket im Pythonpfad ablegen, dann kann man sie jederzeit in anderen Projekten importieren.
- Viele kluge Köpfe entwickeln permanent geniale Pythonpakete, die man sich einzeln je nach Bedarf installieren kann.
- Zum Installieren verwendet man in der Kommandozeile/Terminal folgenden Befehl: `pip install [packagename]`
- Einige Packages kann man sich auch von den GitHub-Seiten der Entwickler/innen herunterladen.
- Nach dem Installieren kann man sofort die Pakete importieren und verwenden. Zur Benutzung am besten immer die Dokumentation des jeweiligen Pakets lesen!

sqlite3 (mitgeliefert)

- Gutes Modul zum Erstellen und Bearbeiten von lokalen Datenbanken im Sqlite-Format.

```
>>> # Modul importieren
>>> import sqlite3
>>> # Verbindung zur Datenbankdatei herstellen
>>> conn = sqlite3.connect("german.sqlite")
>>> # Cursor zum Lesen der DB erzeugen
>>> c = conn.cursor()
>>> # mithilfe des Cursors eine SQL-Abfrage ausfuehren
>>> row = c.execute('''SELECT Wort, POS FROM wort WHERE
                       Wort LIKE "%python%";''')
>>> # fetchall() holt alle Ergebnisse der Anfrage, fetchone()
    # holt ein einziges Ergebnis
>>> print(row.fetchall())
[('Baumpython', 'NN'), ('Felsenpython', 'NN'), ('Fleckenpython',
'NN'), ('Netzpython', 'NN'), ('Python', 'NN'), ('Pythons', 'NN'),
('Ringpython', 'NN'), ('Schwarzkopfpython', 'NN'),
('Teppichpython', 'NN'), ('Wasserpython', 'NN')]
```

csv (mitgeliefert)

- Modul zum Erstellen und Bearbeiten von CSV-Dateien.

```
>>> # Modul importieren
>>> import csv
>>> with open('corpus.csv', 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile, delimiter=';', \
                    quotechar='|', quoting=csv.QUOTE_MINIMAL)
        csvwriter.writerow(["1. Spalte", "Zweite", "Dritte"])
```

urllib (mitgeliefert)

- Gutes Modul zum Öffnen von Webseiten. Auch in der Lage, Cookies zu speichern und Formulare auszufüllen.

```
>>> # Modul importieren
>>> import urllib.request
>>> url = "https://en.wikipedia.org/wiki/Special:Random"

>>> # Webseite anfragen und in r speichern
>>> r = urllib.request.urlopen(url)

>>> # Achtung: Der Quellcode wird nicht als normaler String
    # geliefert, sondern als Bytestring. Er kann aber einfach
    # umgewandelt werden.
>>> r = r.read()
>>> r = r.decode("utf-8")

>>> # Der Webseitentext kann nun wie ein String verarbeitet
    # werden:
>>> print(r.split("<head>", 1)[1].split("<body>", 1)[0])
```

Weitere Pakete

- simplejson: Zum Schreiben und Lesen von JSON-Dateien oder zum Verarbeiten von API-Rückgaben.
- xml: Zum Schreiben und Lesen von XML-Dateien.
- collections: Ermöglicht zusätzliche Funktionen bei Dictionaries, Tupeln, Listen und Mengen.
- time: Zum Messen der Dauer eines Programms, aber auch zum Erzeugen von Timestamps usw., z.B. beim Schreiben von Logdateien
- tweepy: Zum Abfragen und Posten von Tweets, extrem gut geeignet zum Erstellen einer Datenbasis (die man z.B. als Sqlite-Datenbank oder als CSV-Datei speichert)
- pep8: Ein Tool, das prüft, ob der Code eines selbstgeschriebenen Moduls den Vorgaben nach PEP8 entspricht.
- traceback: Ermöglicht eine weiterführende Behandlung von Fehlermeldungen, z.B. nützlich beim Schreiben von Logdateien.

Übung: Erstellen eines Wikipedia-Korpus

- Laden Sie sich die Datei wikipedia_corpus.py von der Kurswebseite herunter.
- Machen Sie sich mit der Funktionalität von `get_website_content(url)` vertraut. Öffnen Sie mehrere Webseiten mithilfe dieser Funktion und lassen Sie sie anzeigen.
- Welche Funktion hat die Zeile `page.decode("utf-8")`?
- Welche Funktion hat die Zeile `html.unescape(...)`?
- Wir wollen ein Korpus aus zufälligen Wikipedia-Artikeln erstellen. Die Funktion `make_corpus()` ist der Einstiegspunkt in das Programm.
- Ergänzen Sie den notwendigen Code, damit jeweils der erste Absatz jedes Wikipedia-Artikels in die CSV-Datei geschrieben wird.