

Python Demo

January 31, 2019

1 Python3 Demo

Dies sind die Python Demos die während des Workshops entstanden sind.

Um Code lesbarer zu machen, fügen Informatiker gerne Kommentare ein. Diese werden in Python mit einer Raute (#) markiert und werden von Python ignoriert - nur der Code wird ausgeführt.

Zwischen Codezeilen können beliebig viele leere Zeilen stehen. Dies dient vor allem dazu den Code übersichtlich zu halten.

1.1 Basics

```
In [2]: # Python ist in der Lage viele mathematische Rechnungen durchzuführen und wird
        # gerne als überdimensionierter Taschenrechner missbraucht.
        print(1 + 7)
```

8

```
In [3]: # Aber auch Texte (in der Informatik "String" genannt) können mit "+" zu einem
        # einzelnen Text zusammengeführt werden
        print("Das! " + "ist " + "Python!")
```

Das! ist Python!

1.2 Listen und For-Schleifen

```
In [5]: # Zunächst erstellen wir eine Liste. Listen werden in Python durch eckige
        # Klammern erstellt. Die Liste "erwin" hier enthält 4 Elemente, die jeweils
        # mit einem Komma separiert sind. Listen können verschiedene Datentypen
        # (Zahlen, Texte oder komplexe Objekte) in einer vorgegebenen Reihenfolge
        # speichern.
        erwin = [1, 2, 4, 5.5]

        # Um den Inhalt der Liste unten in der Konsole auszugeben, können wir die
        # print-Funktion nutzen.
        print(erwin)
```

[1, 2, 4, 5.5]

```
In [6]: # Es folgt ein Beispiel wie wir die Zahlen aus der oben genannten Liste
# summieren können.

# Zunächst müssen wir eine Variable ("summe") definieren, die am Ende
# die Summe enthalten soll. Außer Variablen haben wir keine Möglichkeit
# Zwischenergebnisse zu speichern.
summe = 0

# Mit Hilfe der For-Schleife holen wir nun jedes Element der Liste heraus
# und addieren es zu der Variable "summe".
for zahl in erwin:

    # Diese Zeile mag im ersten Moment verwirrend wirken, macht aber nichts
    # anderes als den Wert der Variable "summe" zu aktualisieren und die
    # Variable "zahl" (also das aktuelle Element aus der Liste "erwin") hinzu
    # zu addieren. Kürzer könnten Sie auch schreiben: summe += zahl .
    # Python würde auch das akzeptieren.
    summe = summe + zahl

    # Hier wird uns das Zwischenergebnis ausgegeben nachdem jeweils eine
    # Zahl addiert wurde
    print(summe)

1
3
7
12.5
```

1.3 Funktionen aufrufen

```
In [12]: # Zunächst definieren wir eine Variable "t", die einen String enthält.
t = "das! ist! sparta!"

# Die Funktion upper() kapitalisiert jeden Buchstaben des Strings, verändert
# den Originalstring aber nicht, sondern gibt nur das Ergebnis zurück. Wenn
# Sie mit dem Ergebnis weiterarbeiten wollen, muss es in einer Variable (hier
# die Variable "gross") gespeichert werden.
gross = t.upper()
print('t hat den Wert"', t, '"')
print('gross hat den Wert "', gross, '"')

t hat den Wert" das! ist! sparta! "
gross hat den Wert " DAS! IST! SPARTA! "
```

1.4 Dateien öffnen

1.4.1 Der alte (nicht empfohlene) Weg

```
In [ ]: # Die Datei selber (nicht ihr Inhalt!) wird mit dem open-Befehl im
# "read"(r) Modus geöffnet.
f = open("1-abbydemo.xml", "r")

# Mit read() können wir nun den tatsächlichen Inhalt der Datei auslesen.
print(f.read())

# Anschließend muss die Datei manuell geschlossen werden.
f.close()
```

1.4.2 Der neue und bessere Weg

```
In [ ]: # Das "with"-Keyword zeigt Python an, dass wir die Datei in einem sogenannten
# "Kontext" öffnen wollen. Kontext heißt hier nichts anderes, als das die Datei
# automatisch (und in jedem Fall) geschlossen wird. Wir müssen uns also nicht
# mehr darum kümmern.
with open("1-abbydemo.xml", "r") as datei:
    print(datei.read())
```

1.5 ABBYY-Prozessierung

Disclaimer: Im folgenden wird ein (!) Weg aufgeführt Klartext aus ABBYY-XML zu extrahieren. Das heißt NICHT, dass es nicht auch anders, schneller, schöner oder kreativer geht. Viele Wege führen zum Ziel und Python versucht Sie dabei bestmöglich zu unterstützen.

1.6 Der 1. Ansatz

Für die Bearbeitung von XML-Dateien im allgemeinen und ABBYY-XML im speziellen können wir das Paket [BeautifulSoup](#) nutzen. Im Allgemeinen gilt, dass Sie mehr Zeit damit verbringen werden Lösungen für Ihr Problem zu googlen als dass Sie am Ende wirklich programmieren.

```
In [ ]: # Wir importieren BeautifulSoup und nennen es der Einfachheit halber "Soup"
from bs4 import BeautifulSoup as Soup

# Wie oben gesehen lesen wir den Inhalt der XML-Datei ein.
with open("1-abbydemo.xml", 'r') as xml_datei:
    xml_daten = xml_datei.read()

# Hier bauen wir ein BeautifulSoup-Objekt indem wir einfach Soup() aufrufen
# und in die runden Klammern die XML-Daten (bzw. die Variable, die diese Daten
# enthält) schreiben, die wir gerne bearbeiten wollen.
suppe = Soup(xml_daten)

# Mit suppe.text (wobei text keine Funktion sondern eine Klassen-Variable ist;
# aber das führt hier jetzt zu weit) können wir den reinen Text (ohne alle
```

```
# XML-Tags) aus den Daten extrahieren.
print(suppe.text)
```

Unser 1. Ansatz hat leider dazu geführt, dass wir zwar den Text unserer digitalisierten Seite bekommen haben (im PDF nicht gezeigt, weil zu lang), aber auch sämtliche Zeilenumbrüche, die zwar in der XML-Datei vorhanden sind, aber im Originaltext nicht. Daher müssen wir das Problem etwas anders angehen.

1.7 Der 2. Ansatz

Wir müssen wohl unsere Strategie anpassen! Nun lassen wir uns alle **charparams**-Tags in der XML-Datei ausgehen. **charparams** enthalten die Metadaten eines einzelnen Zeichens (d.h. Buchstaben, Leerzeichen, Symbole etc.) und das Zeichen selber als Text.

Hier ist ein Beispiel für die ersten beiden Buchstaben "P" und "l" in unserer ABBYY-XML-Datei:

```
<charParams l="196" t="192" r="260" b="274" wordStart="1" wordFromDictionary="1"
wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="100"
serifProbability="4" wordPenalty="0" meanStrokeWidth="167">P</charParams>
```

```
<charParams l="276" t="192" r="292" b="274" wordStart="0" wordFromDictionary="1"
wordNormal="1" wordNumeric="0" wordIdentifier="0" charConfidence="100"
serifProbability="255" wordPenalty="0" meanStrokeWidth="167">l</charParams>
```

Aus den einzelnen Tags extrahieren wir nun das jeweilige Zeichen und hängen sie aneinander.

WICHTIG! Beachten Sie, dass die Tags eigentlich den Namen "charParams" haben, also mit einem groSSen "P". BeautifulSoup hingegen macht alle GroSSbuchstaben zu Kleinbuchstaben und finden nichts, wenn Sie nach "charParams" suchen; Sie müssen nach "charparams" suchen!

```
In [2]: from bs4 import BeautifulSoup as Soup
```

```
with open("1-abbydemo.xml", 'r') as xml_datei:
    xml_datan = xml_datei.read()
```

```
suppe = Soup(xml_datan)
```

```
# Wir definieren eine Variable "volltext", die alle Zwischenergebnisse
# speichert.
```

```
volltext = ''
```

```
# Die Funktion find_all() von BeautifulSoup gibt uns alle Tags IN EINER
# LISTE zurück, die den gegebenen Namen haben. Diese Liste können wir nun
# einzeln mit einer For-Schleife durchgehen und den Text jedes Tags unserer
# Variable "volltext" hinzufügen (wie oben erwähnt führt man Strings einfach
# mit einem "+" zusammen).
```

```
for char in suppe.find_all('charparams'):
    volltext = volltext + char.getText()
```

```
print(volltext)
```

Planet EarthEach planet of the Solar system is unique in its own right, yet Earth has a whole se

Das funktioniert ja schonmal ganz gut. Aber auch nur mit Buchstaben, Symbolen und Leerzeichen. Zeilenumbrüche werden nicht eingefügt.

Das führt zu folgendem Problem: > Planet EarthEach planet
bei welchem zwischen "Earth" und "Each" nicht einmal ein Leerzeichen steht. Wir müssen also noch unseren Code optimieren.

1.7.1 Code Optimierung

In ABBYY-XML werden alle Zeichen einer Zeile in einem Tag zusammengefasst. Das heißt, statt alle Zeichen zu suchen, suchen wir nun alle Zeilen und anschließend lassen wir uns alle Zeichen IN DER ZEILE zurückgeben. Dazu verschachteln wir eine for-Schleife in einer for-Schleife.

Bei einer for-Schleifen-Verschachtelung wird zunächst das erste Element der ersten Schleife aufgerufen und dieses an die geschachtelte Schleife (also hier die eingerückte Schleife) weitergegeben. Die geschachtelte Schleife wird dann einmal komplett durchlaufen, bevor Python wieder zur ersten Schleife zurückspringt und dort das zweite Element herausholt, welches dann wieder in der geschachtelten Schleife komplett einmal durchlaufen wird.

```
In [1]: from bs4 import BeautifulSoup as Soup

with open("1-abbydemo.xml", 'r') as xml_datei:
    xml_daten = xml_datei.read()

suppe = Soup(xml_daten)

volltext = ''

# Wir suchen alle <line> Tags in der Datei
for line in suppe.find_all('line'):

    # Innerhalb eines gefundenen <line>-Tags suchen wir nun alle
    # <charparams> Tags, deren Text wir dann wieder an die Variable
    # "volltext" hängen. Beachten Sie, dass hier die find_all() Funktion
    # auf das <line> Tag angewandt wird, nicht mehr auf die Variable "suppe",
    # die die ALLE Tags enthält.
    for char in line.find_all('charparams'):
        volltext = volltext + char.getText()

volltext = volltext + '\n'

print(volltext)
```

Planet Earth

Each planet of the Solar system is unique in its own right, yet Earth has a whole set of really
First, it is the only active planet - the earthquakes and volcano eruptions constantly change it
Second, it is the only planet that boasts vast resources of liquid water; it's too hot for that
cold on Mars. The Earth's atmosphere is also very unlike any other planet's gaseous shells. Earth

atmospheres consist mainly of carbon dioxide, while the Earth's contains great amounts of oxygen and nitrogen, which form shield from the most dangerous components of solar radiation. The Earth's atmosphere also protects the planet from meteorites. More so, it is this unique combination of changing land surface, oceans and aerial shield that made it possible for another unique phenomenon to exist on Earth.

Our planet photographed from "Apollo - 17". You can notice the big ice polar cap in Antarctica, Grand Canyon: the stream cut through the layers of soft sandstone and limestone in Arizona (USA) into a gigantic valley.

The maximum depth is 1,9 km.

Mississippi Delta: the river Mississippi collects a great amount of sediments along its long run and then drops them in the Gulf of Mexico. This slow river leaves these sediments in the mouth, thus creating areas of new land that didn't exist before.

Seasons

Each 24 hours the Earth completes a full revolution around its axis, which is inclined by 23.5° to the vertical. This inclination is the reason why the seasons change on the Earth as it rotates around the Sun.

Structure

The central part of Earth is a metal core; it's very hot - some 4000°C. and it's surrounded by a shell of liquid iron that creates the magnetic field of Earth. Outer layers form the mantle made up of rocky substances, over which are lighter substances that form the crust. The atmosphere is made of nitrogen (77%), oxygen (21%), and a mixture of water vapor and other gases.

Magnetic bubble

The rotation of Earth around its axis generates forceful electrical currents in the iron core of the planet and this creates the magnetic field. This field forms a giant "bubble" in the near-Earth space called the "magnetosphere". Magnetosphere protects Earth from the solar wind - a flow of charged particles emitted by the Sun. These particles are trapped by the magnetic field in two huge rings - Van Allen's belts. When spacecrafts travel through the Van Allen's belts, the electrical equipment of the former may suffer malfunction caused by these particles.

Clashing Continents

The Earth crust is made from parts called plates, which float on its surface driven by the flows in the liquid mantle. The continents lie on these plates, and so their location is subject to constant change. Some 200 million years ago, all the dry land on Earth was a single continent called Pangea by the scientists, which further split into the continents we know now. The lava rises by millimeters around the mountain ridges located on the ocean floor, and moves the continents apart. When the continents clash, as they do around the shores of the Pacific, the Earth's surface swells up and mountain ridges rise up; if the plates go down into

the mantle, earthquakes and volcanic eruptions occur. This process, called plate tectonics, constantly changes the Earth's appearance.

Carved by water

Many tilings on Earth are carved by wind, yet even more - by water. The sea waves battle with shores and create high cliffs and steep slopes. The rivers carve their beds and cut wonderful valleys on Earth, like the Grand Canyon. The glaciers level the mountains. Yet the water may also create dry land: slow rivers, like Mississippi, create sediments in mouths.

FACTS AND FIGURES

Diameter

12 756 km

Mean distance from Sun

149 600 000 km

Orbital speed

29.79 km/sec

Orbital period

365.26 days

Day (from sunrise to sunrise)

24 hours

Average density

5.52

Temperature at surface

-70° - + 55°C

Der vorherige Code bringt uns ein annehmbares Ergebnis!