

Python: 10. Werken met databronnen

Dr. Cornelis Stal

April 27, 2022

1 Werken met databronnen

1.1 Mappen en bestanden

Om te kunnen werken met bestanden maken we gebruik van de functionaliteiten van de standaard `os`-bibliotheek. Deze module biedt een eenvoudige manier om besturingssysteemafhankelijke functionaliteit te gebruiken. Voor het bevragen en manipuleren van referenties op een computer gebruiken we de `os.path`-bibliotheek.

```
[ ]: # Bibliotheek importeren
import os
# Controleer of een gegeven map bestaat
print(os.path.exists('C:\\Windows'))           # True op Windows
print(os.path.exists('C:\\bestaatNiet'))        # False op Windows
# Controleer of een map of bestand bestaat
print(os.path.isdir('C:\\Windows\\System32'))   # True op Windows
print(os.path.isfile('C:\\Windows\\System32'))  # False op Windows
print(os.path.isdir('C:\\Windows\\System32\\calc.exe')) # False op Windows
print(os.path.isfile('C:\\Windows\\System32\\calc.exe')) # True op Windows
```

1.2 Bestanden inlezen

Gegeven is de onderstaande data, die worden opgeslagen in een bestand `inwoners.csv`. In dit bestand wordt voor iedere provincie en het Vlaams gewest de NIS-code en het aantal inwoners gegeven.

NIS	NAAM	AANTAL
2000	Vlaams Gewest	6444127
10000	Antwerpen	1813282
20001	Vlaams-Brabant	1114299
30000	West-Vlaanderen	1178996
40000	Oost-Vlaanderen	1477346
70000	Limburg	860204

Nadat de `os`-bibliotheek geïmporteerd is en de locatie van het bestand gedefinieerd, volgen we de volgende stappen om de inhoud van het bestand te verkrijgen. Als we alleen een bestand willen lezen of schrijven gebruiken we de standaard *built-in* `open`-methode:

1. Bestand openen met de `open`-methode. De verwijzing naar het bestand en de modus worden als argumenten meegegeven. Voor deze modus zijn er nu drie types relevant:
 - `r`: lezen of *read*;
 - `w`: schrijven of *write*;
 - `a`: aanvullen of *append*;
2. De inhoud van het bestand wordt ingelezen met:
 - de `read`-methode (gehele inhoud als een lange string)
 - de `readlines`-methode (gehele inhoud in een lijst);
3. Als we klaar zijn met het bestand, moeten we deze weer afsluiten met de `close`-methode.

Opmerking: binnen de Python console of de IDE kan hulp opgevraagd worden over het werken met bestanden met behulp van `help(open)`.

```
[ ]: # Importeer de bibliotheek
import os
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './inwoners.csv')
# Open het bestand en lees de inhoud
bestand = open(pad, 'r')
bestandContent = bestand.read()
# Sluit het bestand weer
bestand.close()
# Print de inhoud van het bestand
print(bestandContent)
```

De inhoud van het bestand kan ook regel per regel behandeld worden. Hiervoor kunnen we een `for`-lus gebruiken nadat het bestand geopend is. Merk op dat de `rstrip`-methode gebruikt wordt om de nieuwe regel in te tekst te negeren:

```
[ ]: # Importeer de bibliotheek
import os
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './inwoners.csv')
# Open het bestand en lees de inhoud
with open(pad, 'r') as bestand:
    for regel in bestand:
        # Druk de inhoud van het bestand rij per rij af
        print(regel.rstrip('\n'))
# Sluit het bestand weer
bestand.close()
```

Als we de inhoud van het bestand toe willen voegen aan een lijst, bijvoorbeeld ter voorbereiding van statistische analyse, kunnen we regel per regel de `strip`-methode toepassen op de lijnstring:

```
[ ]: # Importeer de bibliotheek
import os
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './inwoners.csv')
# Open het bestand en lees de inhoud
provLijst = []
with open(pad, 'r') as bestand:
    bestand.readline()
    for regel in bestand:
        # Maak een lijst van de string met een komma als separator
        provLijst.append(regel.rstrip('\n').split(","))
# Sluit het bestand weer
print(provLijst)
bestand.close()
```

De bovenstaande methoden werken met alle type bestanden. In het specifieke geval wordt er echter gewerkt met een CSV-bestand. Hiervoor bestaat er binnen Python de `csv`-bibliotheek. Het inlezen van de data verloopt dan als volgt:

```
[ ]: # Importeer de bibliotheek
import csv
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './inwoners.csv')
with open(pad, newline='\n') as csvBestand:
    dataString = csv.reader(csvBestand, delimiter=',')
    # Print de elementen af als een rij
    for rij in dataString:
        print(rij)
```

Het inlezen van een bestand met de `reader`-methode zal regel per regel resulteren in een lijst met alle elementen. Voor het inlezen van de data geven we hierbij wel een scheidingsteken of `delimiter` mee. Het voordeel van de `csv`-bibliotheek is dat de data op gestandaardiseerde wijze wordt ingelezen. Het wordt nog interessanter wanneer we het bestand inlezen met de `DictReader`-methode. Hierbij wordt de hoofding of `header` van het bestand automatisch omgezet tot de sleutels van een dictionary. De bijbehorende waarden worden aan iedere sleutel toegekend:

```
[ ]: # Importeer de bibliotheek
import csv
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './inwoners.csv')
with open(pad, newline='\n') as csvBestand:
    dataString = csv.DictReader(csvBestand, delimiter=',')
    # Print per rij enkel de geografische naam en het aantal inwoners af
    for rij in dataString:
        print(rij['NAAM'], rij['AANTAL'])
```

In sectie het vervolg van deze cursus wordt het inlezen van CSV- en andere databestanden kort besproken met behulp van de **pandas**-bibliotheek. Deze bibliotheek heeft naast het inlezen van data nog een groot aantal aanvullende functionaliteiten om data te analyseren.

1.3 Bestanden schrijven

Het schrijven van bestanden met Python gebeurt op gelijkaardige wijze als het inlezen van bestanden. Een nieuw bestand kan worden aangemaakt, of nieuwe data kunnen aan een bestaand bestand worden toegevoegd. In beide gevallen wordt de **open**-methode gebruikt, maar respectievelijk in de **w**- en **a**-modus. Python zal in deze modii standaard data verwerken als tekst. Bij het verwerken van binaire data, zoals afbeeldingen, wordt de **b** toegevoegd aan de modus, zoals **wb** voor het wegschrijven van een PNG-bestand. Met de **write**-methode kunnen we data aan het bestand toevoegen. Nadat alle data zijn toegevoegd, sluiten we het bestand af.

Opmerking: wanneer, met het oog op het wegschrijven van data, de **write**-modus van een bestand wordt gebruikt, zal de reeds aanwezige data in het bestand worden overschreven.

```
[ ]: # Importeer de bibliotheek
import os
# Definieer de locatie en de naam van het bestand
huidige_map = os.path.dirname("D://")
pad = os.path.join(huidige_map, './output.csv')
# Open het bestand en lees de inhoud
bestand = open(pad, 'w')
# Weg te schrijven data
provincies = ["VlaamsGewest", "Antwerpen", "Vlaams-Brabant", "West-Vlaanderen", "Oost-Vlaanderen", "Limburg"]
# Data lijn er lijn wegschrijven
for provincie in provincies:
    bestand.write('%s%s' % (provincie, '\n'))
# Sluit het bestand weer
bestand.close()
```

In de bovenstaande code worden de elementen uit een lijst rij per rij opgeslagen in een nieuw bestand.

1.4 Verplaatsen, kopiëren en verwijderen van bestanden

De **shutil**-bibliotheek biedt een aantal functionaliteiten voor het beheren bestanden en mappen. In het bijzonder worden functies geboden die het kopiëren en verwijderen van bestanden ondersteunen. Zie ook de **os**-module voor bewerkingen op individuele bestanden, zoals zojuist besproken. Voor het verplaatsen van bestanden of mappen gebruiken we de **move**-methode:

```
[ ]: # Importeer bibliotheek
import shutil
# Verplaatsen bestand met of zonder hernoemen
# Als de map bestaat, wordt het bestand verplaatst, anders wordt het bestand
hernoemd
```

```
shutil.move('D:\\output.csv', 'D:\\data')
# Bestand wordt verplaatst en hernoemd
shutil.move('D:\\output.csv', 'D:\\data\\provincies.csv')
```

Het kopiëren van bestanden en mappen verloopt met behulp van respectievelijk de `copy`- en de `copytree`-methode:

```
[ ]: # Importeer bibliotheek
import shutil
# Het bestand wordt naar een andere map gekopieerd
shutil.copy('D:\\output.csv', 'D:\\data')
# Het bestand wordt gekopieerd en hernoemd
shutil.copy('D:\\output.csv', 'D:\\data\\provincies.csv')
# De volledige inhoud van de map wordt gekopieerd naar een nieuwe map
shutil.copytree('D:\\data', 'D:\\data_backup')
```

Voor het verwijderen van bestanden gebruiken we de `os`-bibliotheek, waar eveneens de `open`-methode in terug te vinden. Het verwijderen van losse bestanden voeren we uit met de `remove`-methode uit deze bibliotheek. Ook het verwijderen van complete mappen kan met deze bibliotheek, meer bepaald met de `rmdir`-methode. Deze methode veronderstelt echter wel dat de map bestaat, maar ook dat de te verwijderen map volledig leeg is. Mochten er bestanden of onderliggende mappen in deze map aanwezig zijn, kan gebruik gemaakt worden van de `rmtree`-methode uit de `shutil`-bibliotheek:

```
[ ]: # Importeer bibliotheek
import shutil, os
# Verijder een bestand
os.remove("D:\\data\\provincies.csv")
# Verwijderen een map met de os-bibliotheek
os.rmdir("D:\\Data")
# Verwijder een map en de gehele inhoud
shutil.rmtree("D:\\Data")
```

Opmerking: met de `os`- en `shutil`-modules is het (in principe) niet mogelijk om verwijderde bestanden te herstellen. Een bevestiging of de bestanden of mappen daadwerkelijk aangepast of verwijderd moeten worden zal ook niet gevraagd worden. Wees er dus zeker van dat deze zaken correct gebruikt worden.

1.5 Online databronnen bevragen

Python laat toe om met behulp van standaardbibliotheken online databronnen aan te spreken. De `urllib`-bibliotheek is bij de installatie van Python standaard beschikbaar. Om de beschikbare data uit de basisregistratie voor Vlaamse adressen via de API aan te spreken voor de “Valentin Vaerwyckweg 1, 9000, Gent”, kunnen we de volledige URL gebruiken, zoals verder besproken zal worden in een vervolg op deze cursus. We zullen bij de uitvoering van deze operatie een (binair) JSON-bestand terugkrijgen dat we met Python kunnen verwerken:

```
[ ]: # Bibliotheek importeren
import urllib.request
# Complete URL naar de databron
url = 'https://api.basisregisters.vlaanderen.be/v1/adresmatch? \
      gemeentenaam=gent&straatnaam=valentin%20vaerwyckweg& \
      huisnummer=1& format=json'
# Maar verbinding en print data
req = urllib.request.Request(url)
with urllib.request.urlopen(req) as response:
    print(response.read())
```

Het aanpassen van de URL, bijvoorbeeld om een totaal ander adres op te vragen, is met de bovenstaande code niet heel flexibel. De standaard `urllib`-methode is nuttig om simpele referenties te bevragen, maar wanneer er data of parameters verzonden moeten worden, is niet de meest overzichtelijke werkwijze. Ook de verschillende HTTP-methodes worden niet direct ondersteund. Vandaar het nu van de `requests`-bibliotheek, welke te installeren is via zowel Anaconda als via Pip:

```
[ ]: # Bibliotheek importeren
import requests
# Referentie naar de data opmaken
url = 'https://api.basisregisters.vlaanderen.be/v1/adresmatch'
params = {'gemeentenaam': 'gent',
          'straatnaam': 'valentin vaerwyckweg',
          'huisnummer': '1',
          'format': 'json'
}
# Maar verbinding en print data
r = requests.get(url, params=params)
print(r.json())
```

De `requests`-bibliotheek beschikt over methodes om verschillende HTTP-requests uit te voeren, zoals de GET- en POST-methode. In bovenstaande geval hebben we de parameters voor de query verwerkt in een dict-object, en deze als attribuut meegegeven aan de GET-request. Het coderen van de waarden is hierbij niet vereist. Een soortgelijke bevraging kunnen we uitvoeren om een snede uit de WMS van het GRB te verkrijgen, en het resultaat op te slaan in een afbeeldingsbestand, of deze rechtstreeks visualiseren:

```
[ ]: # Bibliotheek importeren
import requests, shutil
# Referentie naar de data opmaken
url = 'https://geoservices.informatievlaanderen.be/raadpleegdiensten/
      ↪GRB-basiskaart/wms'
params = {'SERVICE': 'WMS',
          'VERSION': '1.3.0',
          'REQUEST': 'GetMap',
          'BBOX': '102750.0,191250.0,103750.0,192000.0',
          'CRS': 'EPSG:31370',
```

```

    'WIDTH': '1000',
    'HEIGHT': '750',
    'LAYERS': 'GRB_BSK',
    'FORMAT': 'image/png'
}
# Data ophalen en resultaat opslaan als een afbeelding
r = requests.get(url, params=params, stream=True)
with open('grb_wms.png', 'wb') as afbBestand:
    shutil.copyfileobj(r.raw, afbBestand)
from PIL import Image
from io import BytesIO
i = Image.open(BytesIO(r.content))
display(i)

```

Opmerking: om de requests-module te installeren, kan een van de volgende commando's gebruikt worden via de Opdrachtprompt of de prompt van Anaconda:

Met Anaconda: `conda install -c anaconda requests`

Met PyPi: `python -m pip install requests`

Gebruikshandleiding: voor meer informatie over de requests en het gebruik van de verschillende methodes wordt doorverwezen naar <https://requests.readthedocs.io> of de [Notebook](#) in deze reeks.