Geodata

Adres naar coördinaten/ coördinaten naar adres

Functionaliteit bestaat vanuit Vlaanderen door middel van Geolocation Service dat beschikt over een REST API die kan opgeroepen worden. (https://overheid.vlaanderen.be/CRAB-Geolocation)

Python kan de URL ophalen van de API indien in de URL de nodige gegevens staan. Deze stuurt dan een JSON terug met de informatie die gekend is over dat adres gekoppeld aan het CRAB of POI. Ikzelf gebruik dit om de Lamber72 coördinaten op te halen als basis voor het ophalen van andere geodata.

```
import requests
import json
import urllib.parse


# adress = input()
adress = 'Singel 10 kaprijke'
def getcoordinates(adres):
    percentencodedadress = urllib.parse.quote(adress)
    url =
"https://loc.geopunt.be/v4/Location?q=url".replace("url",percentencodedadress)
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        json_string = json.dumps(data, ensure_ascii=False, indent=4)
        with open("./Output/location.json", "w") as write_file:
            write_file.write(json_string)
    else:
        response.raise_for_status()

    x = data['LocationResult'][0]['Location']['X_Lambert72']
    y = data['LocationResult'][0]['Location']['Y_Lambert72']

    return [x,y]
```

Bestaande kaarten ophalen

Ik maak gebruik van bestaande kaarten die door een instantie in WFS worden aangeboden. WFS is een vector gebaseerd open data endpoint waarmee geodata kan worden opgevraagd. Een WFS service bestaat in de vorm van een URL dat verschillende kaarten omvat en kan rechtstreeks ingeladen worden in een GIS software. Men kan echter ook verschillende operaties uitvoeren op de WFS door deze te specifiëren in de URL. De operatie GetFeature geeft de geometrie & attributen terug. Om een kaart te bevragen moet men wel de juiste naam van deze kaart in de url verwerken. Via de URL kan men ook het output format kiezen (GML, JSON, Shapefile of CSV).

Indien men deze URL opvraagt met de juiste notatie krijgt men een (in mijn geval JSON) terug met alle geometrie. Om hier op te filteren kan men in de URL gebruik maken van CQL filters. Ik maak gebruik

van de CQL_FILTER=CONTAINS methode met als input de coördinaten volgens de CRS van de WFS om alle geometrieën te ontvangen waar de coördinaten inliggen.

```
import requests
import urllib.parse
import json

# x = float(input())
# y = float(input())
x=100688
y=207134

percentencodedxy = urllib.parse.quote(str(x)+' '+str(y))
urlperceel =
"https://geo.api.vlaanderen.be/GRB/wfs?service=WFS&version=2.0.0&request=GetFe
ature&typeNames=GRB:ADP&outputFormat=application/json&CQL_FILTER=CONTAINS(SHAP
E,POINT(url))".replace('url',percentencodedxy)
bestandsnaamperceel = 'perceel'
urlwater =
"https://www.dov.vlaanderen.be/geoserver/wfs?service=WFS&version=2.0.0&request
=GetFeature&typeNames=gw_bescherming:beschermingszones_2016&outputFormat=appli
cation/json&CQL_FILTER=CONTAINS(geom,POINT(url))".replace('url',percentencoded
xy)
bestandsnaamwater = 'winwatergebied'

def getWFS(url,bestandsnaam):
    #omzeilen bescherming tegen script
    user_agent = 'Mozilla/5.0'
    response = requests.get(url, headers={'User-Agent': user_agent})

    if response.status_code == 200:
        data = response.json()
        json_string = json.dumps(data, ensure_ascii=False, indent=4)
        with open("Output/"+bestandsnaam+".json", "w") as write_file:
            write_file.write(json_string)
        return data
    else:
        response.raise_for_status()

getWFS(urlperceel,bestandsnaamperceel)
getWFS(urlwater,bestandsnaamwater)
```