OpenStreetMap

- Daten Parsen
  - openstreetmap tools

Daten Parsen

openstreetmap tools

Beschreibung:

Man kann mit dem Linux tool openstreetmaptolls, Daten aus einer downgeloadeten Openstreetmap Datei pbf exportieren in eine csv.

Unter https://download.geofabrik.de lassen sich Ganze Kontinente oder nur Regionen oder einzelne Länder downloaden

Vorrausetzung:

```
sudo apt install osmctools python3 python3-pandas python3-geopy python3-geopandas
```

Parsen

Nach dem man seine gewünschte Datei downgeloadet hat, kann man die Adressen exportieren.
uns interessieren Straße, Hausnummer PLZ, Ort und Ortsteil.
Der Parameter -o gibt die Ausgabe Datei an.

```
osmconvert germany-latest.osm.pbf --csv="@id addr:street addr:housenumber addr:postcode addr:city addr:suburb" -o=addresses.csv
```

Unvollständige Daten entfernen

Python Script um nur Vollständige Adressen zu behalten in der csv
Denn sonst sieht das so aus, mit grep Wunderburgpark

ID	Straße	Hausnummer	Postleitzahl	Ort
540052	Am Wunderburgpark	5b	26135	Oldenburg
1582540056	Am Wunderburgpark	6	26135	Oldenburg
1582540114	Am Wunderburgpark	7	26135	Oldenburg
1582540118	Am Wunderburgpark	8	26135	Oldenburg

ID	Straße	Hausnummer	Postleitzahl	Ort
1582540163	Am Wunderburgpark	9	26135	Oldenburg
1582540165	Am Wunderburgpark	9b	26135	Oldenburg
35624089	Am Wunderburgpark		26135	Oldenburg
35624098	Am Wunderburgpark		26135	Oldenburg
35624100				
35624102				

Das Script

```
import pandas as pd
import argparse
import sys

def clean_csv(input_file, output_file):
    """Bereinigt eine CSV-Datei, indem unvollständige Datensätze entfernt werden."""
    try:
        # CSV-Datei laden
        df = pd.read_csv(input_file, sep="\t", names=["id", "street", "housenumber", "postcode", "city", "suburb"])

        # Filter für vollständige Datensätze
        filtered_df = df.dropna(subset=["street", "housenumber", "postcode", "city"])  # Entfernt Zeilen mit leeren Werten
        filtered_df = filtered_df[filtered_df["housenumber"].notnull() & (filtered_df["housenumber"] != "")]

        # Bereinigte Daten speichern
        filtered_df.to_csv(output_file, index=False, sep="\t", encoding="utf-8")
        print(f"Bereinigte Datei wurde gespeichert: {output_file}")
    except Exception as e:
        print(f"Fehler bei der Verarbeitung: {e}")
        sys.exit(1)

def main():
    # Argument-Parser einrichten
    parser = argparse.ArgumentParser(description="Bereinigt eine CSV-Datei, indem unvollständige Datensätze entfernt werden.")
    parser.add_argument("input_file", metavar="INPUT", type=str, help="Pfad zur Eingabedatei (CSV, Tab-separiert)")
    parser.add_argument("output_file", metavar="OUTPUT", type=str, help="Pfad zur Ausgabedatei (bereinigte
```

```
CSV)")
    args = parser.parse_args()

    # CSV-Bereinigung durchführen
    clean_csv(args.input_file, args.output_file)

if __name__ == "__main__":
    if len(sys.argv) == 1 or "--help" in sys.argv:
        print("""
Verwendung:
    python script.py INPUT OUTPUT

Beschreibung:
    Dieses Skript bereinigt eine CSV-Datei, indem unvollständige Datensätze (z. B. fehlende Hausnummern)
    entfernt werden. Die bereinigte Datei wird als neue CSV gespeichert.

Parameter:
    INPUT   Pfad zur Eingabedatei (CSV, Tab-separiert)
    OUTPUT  Pfad zur Ausgabedatei (bereinigte CSV)

Beispiel:
    python script.py addresses_raw.csv addresses_cleaned.csv
        """)
        sys.exit(0)

    main()
```

Ausführen:

```
python3 adresses.py addresses.csv addresses_clean.csv
```

Da das cleanen sehr Recheninstensiv ist, einfach geduld.