CHVorl2

November 12, 2016

1 Vorlesung 2: Gruppen von Individuen

1.1 Data mit Pandas Dataframes

Python Einführung: http://www.diveintopython.net/toc/index.html
Reguläre Ausdrücke: http://www.regexe.de/hilfe.jsp https://pymotw.com/2/re
Pandas: http://www.data-analysis-in-python.org/3_pandas.html : https://bitbucket.org/hrojas/learn-pandas

1.2 Pandas Dataframes

```
In [1]: import json
        import pandas as pd
        import re
        import numpy as np
        import matplotlib.pyplot as plt

In [2]: # json.load erzeugt ein dictionary der JSON Daten
        with open('chapter1.json') as json_data:
            PoleisRawData = json.load(json_data)

In [3]: # Abfrage des Datentyps
        type(PoleisRawData)

Out[3]: dict

In [4]: # Ausgabe des Dictionaries
        #PoleisRawData

In [5]: # list() erstellt eine Liste der keys
        PoleisKeyList = list(PoleisRawData)
        PoleisKeyList

Out[5]: ['10. Akrai ',
         '28. Kamarina ',
         '49. Tyndaris ',
         '41. Naxos ',
         '12. Alontion ',
         '48. Tauromenion ',
```

```
 '19. Henna ',
 '22. Herbes(s)os ',
 '32. Kephaloidion ',
 '50. (Tyrrhenoi)',
 '26 *Imachara ',
 '38. Mylai ',
 '18. Heloron ',
 '44. Selinous ',
 '9. Akragas ',
 '14. Engyon ',
 '25. Hippana ',
 '46. (Stielanaioi)',
 '7. Agyrion ',
 '16. Galeria ',
 '43. Piakos ',
 '24. Himera ',
 '31. Kentoripa ',
 '47. Syrakousai ',
 '35. *Longane ',
 '17. Gela ',
 '33. Leontinoi ',
 '11. Alaisa ',
 '36. Megara ',
 '27. Kallipolis ',
 '21. Herakleia 2 ',
 '29. Kasmenai ',
 '20. Herakleia 1',
 '15. Euboia ',
 '51. Zankle ',
 '42. Petra ',
 '40. Nakone ',
 '30. Katane ',
 '23. Herbita ',
 '8. Aitna ',
 '39. Mytistratos ',
 '6. Adranon ',
 '5. Abakainon ',
 '34. Lipara ',
 '37. Morgantina ',
 '13. Apollonia ',
 '45. (Sileraioi)']
```

```
In [6]: # Listenelemente werden mit eckigen Klammern ausgewählt. Eine Index-Zahl gi
        PoleisKeyList[1]
```

```
Out[6]: '28. Kamarina '
```

```
In [7]: for i in PoleisKeyList:
```

```
            if "Megara" in PoleisRawData[i]:
                print(i)
```

```
18. Heloron
44. Selinous
46. (Stielanaioi)
47. Syrakousai
33. Leontinoi
36. Megara
15. Euboia
```

```
In [8]: # Der key in eckigen Klammern gibt den Wert des Elements mit dem Key aus
        PoleisRawData[PoleisKeyList[4]]
```

```
Out[8]: "(Alontinos) Map  47.  Lat. 38.05,long.  14.40. Size of  territory:  ? Type
```

```
In [9]: # Liest das Dictionary als Dataframe ein. Namen der Poleis werden als Index
        dfPoleis = pd.DataFrame([PoleisRawData]).transpose()
        dfPoleis.head(4)
```

```
Out[9]:                                                                    0
        10. Akrai       (Akraios) Map  47.  Lat. 37.05,long.  14.55.  ...
        11. Alaisa      (Alaisinos) Map  47.  Lat. 38.00,long.  14.15...
        12. Alontion    (Alontinos) Map  47.  Lat. 38.05,long.  14.40...
        13. Apollonia   (Apolloniates) Map  47.  Lat. 38.00,long. 14.3...
```

```
In [10]: # Umnennen der Spalte von 0 zu 'full_text'
         dfPoleis = dfPoleis.rename(columns={0: 'Beschreibung'})
         dfPoleis.head()
```

```
Out[10]:                                                        Beschreibung
         10. Akrai       (Akraios) Map  47.  Lat. 37.05,long.  14.55.  ...
         11. Alaisa      (Alaisinos) Map  47.  Lat. 38.00,long.  14.15...
         12. Alontion    (Alontinos) Map  47.  Lat. 38.05,long.  14.40...
         13. Apollonia   (Apolloniates) Map  47.  Lat. 38.00,long. 14.3...
         14. Engyon      (Engyinos)  Map  47.  Lat. 37.45,long.  14.35...
```

2 Konstruktion neuer Merkmale

2.1 Textmuster mit regulären Ausdrücken

http://www.regexe.de/hilfe.jsp https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/
http://www.coli.uni-saarland.de/courses/python1-10/folien/PythonI10-07.pdf

```
In [11]: # Konstruktion einer Liste mit sogenannten List-Comprehensions
         ListCities = [x[4:] for x in dfPoleis.index]
         ListCities
```

```
Out[11]: ['Akrai ',
          'Alaisa ',
          'Alontion ',
          'Apollonia ',
          'Engyon ',
          'Euboia ',
          'Galeria ',
          'Gela ',
          'Heloron ',
          'Henna ',
          'Herakleia 1',
          'Herakleia 2 ',
          'Herbes(s)os ',
          'Herbita ',
          'Himera ',
          'Hippana ',
          'Imachara ',
          'Kallipolis ',
          'Kamarina ',
          'Kasmenai ',
          'Katane ',
          'Kentoripa ',
          'Kephaloidion ',
          'Leontinoi ',
          'Lipara ',
          '*Longane ',
          'Megara ',
          'Morgantina ',
          'Mylai ',
          'Mytistratos ',
          'Nakone ',
          'Naxos ',
          'Petra ',
          'Piakos ',
          'Selinous ',
          '(Sileraioi)',
          '(Stielanaioi)',
          'Syrakousai ',
          'Tauromenion ',
          'Tyndaris ',
          'bakainon ',
          '(Tyrrhenoi)',
          'Zankle ',
          'dranon ',
          'gyrion ',
          'itna ',
          'kragas ']
```

```
In [12]: # Extrahiere Name der Polis aus Index
         dfPoleis['city'] = [x[4:] for x in dfPoleis.index]

In [13]: # Extrahiere Nummer des Polis Eintrags
         dfPoleis['city_index'] = [int(re.findall('\d{1,2}', x)[0]) for x in dfPole

In [14]: # Sortiere die Zeilen nach der Spalte
         dfPoleis = dfPoleis.sort_values(by='city_index')

In [15]: dfPoleis.head(4)

Out[15]:                                                    Beschreibung       cit
         5. Abakainon   (Abakaininos) Map  47.  Lat. 38.05,long. 15.05...  bakainor
         6. Adranon     (Adranites) Map  47.  Lat. 37.40,long.  14.50...    dranon
         7. Agyrion     (Agyrinaios) Map  47.  Lat. 37.40,long.  14.30...    gyrion
         8. Aitna       (Aitnaios) Map  47.Location  of  Aitna  I  as ...      itna

                        city_index
         5. Abakainon            5
         6. Adranon              6
         7. Agyrion              7
         8. Aitna                8
```

2.2 Textmustersuche in der Beschreibung einer Polis

2.2.1 Neue Funktionen

```
In [16]: def ListePattern(value,pattern):
             x = re.findall(pattern, value)
             if x:
                 return(x[0][-5:])
```

2.2.2 Geographische Koordinaten

```
In [17]: ListePattern(dfPoleis["Beschreibung"][0],"Lat\.\s?\d+\.\d+")

Out[17]: '38.05'

In [18]: # gleiches auch für long.
         listLong=ListePattern(dfPoleis['Beschreibung'][0],"long\.\s*\d+\.\d+")
         listLong

Out[18]: '15.05'

In [19]: dfPoleis['Latitude'] = dfPoleis['Beschreibung'].apply(ListePattern,patterr

         dfPoleis['Longitude'] = dfPoleis['Beschreibung'].apply(ListePattern,patter

In [20]: dfPoleis
```

Out[20]:

		Beschreibung \
5.	Abakainon	(Abakaininos) Map 47. Lat. 38.05,long. 15.05...
6.	Adranon	(Adranites) Map 47. Lat. 37.40,long. 14.50...
7.	Agyrion	(Agyrinaios) Map 47. Lat. 37.40,long. 14.30...
8.	Aitna	(Aitnaios) Map 47.Location of Aitna I as ...
9.	Akragas	(Akragantinos) Map 47. Lat. 37.20,long. 13...
10.	Akrai	(Akraios) Map 47. Lat. 37.05,long. 14.55. ...
11.	Alaisa	(Alaisinos) Map 47. Lat. 38.00,long. 14.15...
12.	Alontion	(Alontinos) Map 47. Lat. 38.05,long. 14.40...
13.	Apollonia	(Apolloniates) Map 47. Lat. 38.00,long. 14.3...
14.	Engyon	(Engyinos) Map 47. Lat. 37.45,long. 14.35...
15.	Euboia	(Euboeus) Map 47. Unlocated. Type: C: .Th...
16.	Galeria	(Galarinos) Map 47.Unlocated (Manni (1981)...
17.	Gela	(Geloios, Geloaios) Map 47. Lat. 37.05,long...
18.	Heloron	(Ailoros) Map 47. Lat. 36.50,long. 15.05. ...
19.	Henna	(Hennaios) Map 47. Lat. 37.35,long. 14.15. ...
20.	Herakleia 1	(Herakleotes) Map 47. Lat. 37.25,long. 13.15...
21.	Herakleia 2	Map 47.Unlocated site in western Sicily, i...
22.	Herbes(s)os	(Herbessinos) Map 47. Unlocated, but presum...
23.	Herbita	(Herbitaios) Map 47.Unlocated (cf. C. Boeh...
24.	Himera	(Himeraios) Map 47. Lat. 37.55,long. 13.50...
25.	Hippana	(Hipanatas) Map 47. Lat. 37.40,long. 13.25...
26	*Imachara	(Imacharaios) Map 47. Unlocated. Barr. tent...
27.	Kallipolis	(Kallipolites) Map 47. Unlocated. Type: A...
28.	Kamarina	(Kamarinaios) Map 47. Lat. 36.50,long. 14.25...
29.	Kasmenai	(Kasmenaios) Map 47. Lat. 37.05,long. 14.50...
30.	Katane	(Katanaios) Map 47. Lat. 37.30,long. 15.05...
31.	Kentoripa	(Kentoripinos) Map 47. Lat. 37.35,long. 14.4...
32.	Kephaloidion	(Kephaloiditas) Map 47. Lat. 38.00, long. 14...
33.	Leontinoi	(Leontinos) Map 47. Lat. 37.15,long. 15.00...
34.	Lipara	(Liparaios) Map 47. Lat. 38.30 long. 14.55...
35.	*Longane	(Longenaios) Map 47. Lat. 38.05,long. 15.10...
36.	Megara	(Megareus) Map 47. Lat. 37.10,long. 15.10. ...
37.	Morgantina	(Morgantinos) Map 47. Lat. 37.25,long. 14.30...
38.	Mylai	(Mylaios) Map 47. Lat. 38.15, long 15.15. ...
39.	Mytistratos	(Mytiseratinos) Map 47. Lat. 37.35,long. 14...
40.	Nakone	(Nakonaios) Map 47. Unlocated (Tegon (199...
41.	Naxos	(Naxios) Map 47. Lat. 37.50,long. 15.15. S...
42.	Petra	(Petrinos) Map 47. Unlocated (cf. Bejor (...
43.	Piakos	(Piakinos) Map 47. Unlocated, but possibly...
44.	Selinous	(Selinousios) Map 47. Lat. 37.35,long. 12.5...
45.	(Sileraioi)	Map 47. Unlocated (cf. Manni (1981) 225)...
46.	(Stielanaioi)	Map 47. Lat. 37.10,long. 14.55: the loca- ...
47.	Syrakousai	(Syrakosios) Map 47. Lat. 37.05,long. 15.15...
48.	Tauromenion	(Tauromenitas) Map 47. Lat. 37.50, long. 15...
49.	Tyndaris	(Tyndarites) Map 47. Lat. 38.10,long. 15.05...
50.	(Tyrrhenoi)	Map 47. Unlocated. Type: C: .The Tyr- rhen...
51.	Zankle	(Zanklaios)/Messana

	city	city_index	Latitude	Longitude
5. Abakainon	bakainon	5	38.05	15.05
6. Adranon	dranon	6	37.40	14.50
7. Agyrion	gyrion	7	37.40	14.30
8. Aitna	itna	8	NaN	NaN
9. Akragas	kragas	9	37.20	13.35
10. Akrai	Akrai	10	37.05	14.55
11. Alaisa	Alaisa	11	38.00	14.15
12. Alontion	Alontion	12	38.05	14.40
13. Apollonia	Apollonia	13	38.00	14.35
14. Engyon	Engyon	14	37.45	14.35
15. Euboia	Euboia	15	NaN	NaN
16. Galeria	Galeria	16	NaN	NaN
17. Gela	Gela	17	37.05	14.15
18. Heloron	Heloron	18	36.50	15.05
19. Henna	Henna	19	37.35	14.15
20. Herakleia 1	Herakleia 1	20	37.25	13.15
21. Herakleia 2	Herakleia 2	21	NaN	NaN
22. Herbes(s)os	Herbes(s)os	22	NaN	NaN
23. Herbita	Herbita	23	NaN	NaN
24. Himera	Himera	24	37.55	13.50
25. Hippana	Hippana	25	37.40	13.25
26 *Imachara	Imachara	26	NaN	14.20
27. Kallipolis	Kallipolis	27	NaN	NaN
28. Kamarina	Kamarina	28	36.50	14.25
29. Kasmenai	Kasmenai	29	37.05	14.50
30. Katane	Katane	30	37.30	15.05
31. Kentoripa	Kentoripa	31	37.35	14.45
32. Kephaloidion	Kephaloidion	32	38.00	14.00
33. Leontinoi	Leontinoi	33	37.15	15.00
34. Lipara	Lipara	34	38.30	14.55
35. *Longane	*Longane	35	38.05	15.10
36. Megara	Megara	36	37.10	15.10
37. Morgantina	Morgantina	37	37.25	14.30
38. Mylai	Mylai	38	38.15	NaN
39. Mytistratos	Mytistratos	39	37.35	14.00
40. Nakone	Nakone	40	NaN	NaN
41. Naxos	Naxos	41	37.50	15.15
42. Petra	Petra	42	NaN	NaN
43. Piakos	Piakos	43	NaN	NaN
44. Selinous	Selinous	44	37.35	12.50
45. (Sileraioi)	(Sileraioi)	45	NaN	NaN
46. (Stielanaioi)	(Stielanaioi)	46	37.10	14.55
47. Syrakousai	Syrakousai	47	37.05	15.15
48. Tauromenion	Tauromenion	48	37.50	15.15
49. Tyndaris	Tyndaris	49	38.10	15.05
50. (Tyrrhenoi)	(Tyrrhenoi)	50	NaN	NaN

```
         51. Zankle                Zankle           51       NaN        NaN
```

2.2.3 Zitatnachweise, Namen, Jahreszahlen

```
In [21]: dfPoleis["Beschreibung"].iloc[0]
```

```
Out[21]: '(Abakaininos) Map  47.  Lat. 38.05,long. 15.05.  Size  of  territory:  ?
```

2.3 Muster (Pattern) zur Erkennung der Literaturreferenzen

- Primärquellen

(Polyb. 1.18.2) (Diod. 13.85.4 (r 406)) (Diod. 13.108.2) (Hdt. 7.165; IGDS no. 182a) (Pind. Pyth. 6) (Thuc. 6.4.4: $\mu\mu$) (Xanthos (FGrHist 765) fr. 33; Arist. fr. 865)

- Sekundärquellen

(Karlsson (1995) 161 (Waele (1971) 195; Hinz (1998) 79)

- Jahreszahlen (dddd)

```
In [22]: # Definiere Funktion die reguläre Ausdrücke auf den DataFrame anwendet.
         def ListePatternFull(value,pattern):
             x = re.findall(pattern, value)
             if x:
                 return(x)
```

2.3.1 Testen der regulären Ausdrücke

```
In [23]: dfPoleis['Beschreibung'].iloc[0]
```

```
Out[23]: '(Abakaininos) Map  47.  Lat. 38.05,long. 15.05.  Size  of  territory:  ?
```

Finde alle groß-geschriebenen Wörter mit mindestens 3 nachfolgenden kleinen Buchstaben.

```
In [24]: re.findall('[A-Z][a-z]{3,10}',dfPoleis['Beschreibung'].iloc[0])
```

```
Out[24]: ['Abakaininos',
          'Size',
          'Type',
          'Diod',
          'Diod',
          'Steph',
          'Diod',
          'Steph',
          'Abakainon',
          'Diod',
          'Magon',
          'Agathokles',
          'Kamarina',
```

```
 'Leontinoi',
 'Katane',
 'Messana',
 'Diod',
 'Diod',
 'Dionysios',
 'Abakainon',
 'Tyndaris',
 'Diod',
 'Abakainon',
 'Tyndaris',
 'Tripi',
 'However',
 'Diodorus',
 'Carthaginia',
 'Manni',
 'Dionysios',
 'There',
 'Greek',
 'Greek',
 'Villard',
 'Leontinoi',
 'Bacci',
 'Spigo',
 'Greek',
 'Abakainon',
 'Zeus',
 'Apollo',
 'Demeter',
 'Persephone',
 'Head',
 'Bertino',
 'Sicily',
 'Probably',
 'Timoleon',
 'Head',
 'Bertino',
 'Sicily',
 'Dioskouros',
 'Tyndaris',
 'Italy',
 'Bertino']
```

Finde alle Ausdrücke wie oben, denen ein Punkt folgt, mit anschließenden Zifferfolgen der Form [Ziffern][Punkt][Ziffern][Punkt][Ziffern]

```
In [25]: re.findall('[A-Z][a-z]{1,10}\. \d{1,3}\.\d{1,3}\.\d{1,3}',dfPoleis['Beschr

Out[25]: ['Diod. 14.90.3',
```

```
 'Diod. 19.65.6',
 'Diod. 14.78.5',
 'Diod. 14.90.3',
 'Diod. 19.65.6',
 'Diod. 19.110.4']
```

Finde alle Ausdrücke wie oben, wobei statt des Punktes nach den kleinen Buchstaben auch zwei Leerzeichen und eine runde Klammer folgen können

```
In [26]: re.findall('[A-Z][a-z]{1,10}[\.| ] [(|\d{0,4}][\d{1,4}| ][\.|\d{}]\d{1,3}|
```

```
Out[26]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4',
          'Manni  ( 1976',
          'Villard  ( 1954)',
          'Spigo  ( 1997',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)',
          'Bertino  ( 1975)']
```

```
In [27]: #Alternativ kann man auch Gruppen definieren zB. (A(B|C))
         #re.findall('([A-Z][a-z]{1,10}(\. \d{1,3}\.\d{1,3}\.\d{1,3}|\s{2}\(\s\d{4}
```

```
In [28]: dfPoleis['Namen'] = dfPoleis['Beschreibung'].apply(ListePatternFull,patter
```

```
In [29]: dfPoleis['Quellen'] = dfPoleis['Beschreibung'].apply(ListePatternFull,patt
```

```
In [30]: dfPoleis.head(4)
```

```
Out[30]:                                                    Beschreibung       cit
         5. Abakainon   (Abakaininos) Map  47.  Lat. 38.05,long. 15.05...  bakainor
         6. Adranon     (Adranites) Map  47.  Lat. 37.40,long.  14.50...    dranon
         7. Agyrion     (Agyrinaios) Map  47.  Lat. 37.40,long.  14.30...    gyrion
         8. Aitna       (Aitnaios) Map  47.Location  of  Aitna  I  as ...      itna

                       city_index  Latitude  Longitude  \
         5. Abakainon           5     38.05      15.05
         6. Adranon             6     37.40      14.50
         7. Agyrion             7     37.40      14.30
         8. Aitna               8       NaN        NaN

                                                                Namen  \
         5. Abakainon   [Abakaininos, Size, Type, Diod, Diod, Steph, D...
         6. Adranon     [Adranites, Size, Type, Diod, Steph, Diod, Adr...
         7. Agyrion     [Agyrinaios, Size, Type, Diod, Ptol, Geog, Ste...
```

```
8. Aitna       [Aitnaios, Location, Aitna, Katane, Aitna, Dio...

                                                          Quellen
5. Abakainon   [Diod. 14.90.3, Diod. 19.65.6, Diod. 14.78.5, ...
6. Adranon     [Diod. 14.37.5, Diod. 16.68.9, Diod. 14.37.5, ...
7. Agyrion     [Byz. 23.19), Diod. 16.82.4, Moggi  ( 1976), D...
8. Aitna       [Diod. 11.49.1, Diod. 11.49.1, Diod. 11.66.4, ...
```

```
In [31]: #dfBesp = dfPoleis['Beschreibung'].str.replace('(','\n(').str.replace(')',
```

```
In [32]: #print(dfBesp.iloc[0])
```

2.4 Muster zur Erkennung von Namen

Empedokles (496) Theron (476) Timoleon c. 338

```
In [33]: # MALTE: RE für die Primärquellen; Sekundärquellen; Namen entwickeln und a
```

3 Datenvalidierung

3.1 Wertverteilungen, Test auf Dopplungen

Lese Werte der Spalte Quellen als Liste aus.

```
In [34]: mainList = dfPoleis['Quellen'].values.tolist()
```

Reduziere Unterlisten auf eine Gesamtliste.

```
In [35]: quellenListe = []
         for sublist in mainList:
             if sublist:
                 for k in range(len(sublist)):
                     quellenListe.append(sublist[k])
```

```
In [36]: quellenListe[:10]
```

```
Out[36]: ['Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 14.78.5',
          'Diod. 14.90.3',
          'Diod. 19.65.6',
          'Diod. 19.110.4',
          'Manni  ( 1976',
          'Villard  ( 1954)',
          'Spigo  ( 1997',
          'Bertino  ( 1975)']
```

Zähle die Häufigkeit der verschiedenen Quellen und speichere als Dictionary.

```
In [37]: quellenVerteilung = {x:quellenListe.count(x) for x in quellenListe}
         quellenVerteilung['Diod. 14.90.3']

Out[37]: 2
```

Erzeuge DataFrame, mit neuem Index und Namen der Spalten. Sortiere diesen Nach der Häufigkeit der Quelle.

```
In [38]: dfQuellenVerteilung = pd.DataFrame([quellenVerteilung])

In [39]: dfQuellenVerteilung = dfQuellenVerteilung.transpose().reset_index()

In [40]: dfQuellenVerteilung = dfQuellenVerteilung.rename(columns={'index': 'Quelle

In [41]: dfQuellenVerteilung.sort_values(by='Häufigkeit',ascending=False).head(10)

Out[41]:                    Quelle  Häufigkeit
         382    Manganaro  ( 1996          15
         331        Hinz  ( 1998)          13
         509     Talbert  ( 1974)          13
         83     Cavalier  ( 1991)          11
         344    Karlsson  ( 1995)           9
         226        Diod. 14.78.7           9
         113        Diod. 11.49.2           9
         47   Boehringer  ( 1998)           8
         478      Rutter  ( 1997)           8
         327      Hansen  ( 2000)           8
```

3.2 Visualisierungen

Zeige die Verteilung der Häufigkeiten als Histogram. Um auch Häufigkeiten zu erkennen, die nicht so oft auftreten, kann man in die Darstellung zoomen. Das Haus-Symbol zeigt wieder den ursprünglichen Zustand der Figur.

```
In [42]: %matplotlib notebook

In [45]: fig, ax = plt.subplots(figsize=(8,6))

         dfQuellenGrouped = dfQuellenVerteilung.groupby(by='Häufigkeit').plot(kind=

<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


In [44]: import folium
         from folium import plugins
         from folium.map import *
```

```
---------------------------------------------------------------------------

ImportError                               Traceback (most recent call last)

<ipython-input-44-9eae33d69195> in <module>()
----> 1 import folium
      2 from folium import plugins
      3 from folium.map import *


ImportError: No module named 'folium'
```

Remove all entries where no latitude or longitude is given.

In [ ]:
```
dfPoleisMap = dfPoleis.dropna(axis=0)
```

In [ ]:
```
poleis_map = folium.Map(location=[dfPoleisMap["Latitude"][0],dfPoleisMap["L

marker = FeatureGroup(name='Poleis')

marker_cluster = folium.MarkerCluster().add_to(marker)

for i in range(len(dfPoleisMap)):
    folium.Marker([dfPoleisMap['Latitude'][i], dfPoleisMap['Longitude'][i]]

poleis_map.add_children(marker)

poleis_map.add_children(folium.map.LayerControl())
```

In [ ]:
```

```