

Python

Data & Automation

Daten

- Daten begegnen uns in vielen Formen, teilweise **wohlgeformt und vorbereitet** und teilweise noch **sehr roh oder gar fehlerbehaftet**.
- Nachdem wir nun wissen, wie wir z. B. aus Webseiten Daten beziehen können, wollen wir nun Daten einlesen und **vollautomatisch** wiedergeben.

Szenario: Auf der Suche nach Daten

The image displays three separate web interfaces side-by-side:

- kaggle**: A sidebar menu on the left with options like Create, Home, Competitions, Datasets, and Models. The main area shows a "Datasets" section with a search bar and a "+ New Dataset" button.
- Stadt Zürich Open Data**: A header with the city logo and "Stadt Zürich Open Data". Below it is a navigation bar with links to Startseite, Datensätze, Kategorien, and Showcases. A large blue rectangular area is positioned to the right of the navigation bar.
- opendata.swiss**: A teal-colored landing page with the text "Finden Sie Schweizer Open Government Data" and a button "Erfahren Sie mehr über opendata.swiss".

Szenario: Auf der Suche nach Daten

Standorte ZüriVelo

Offizielle Standorte der Veloverleihstationen von Züri Velo (PubliBike) Zweck: Dieser Datensatz dient zur Übersicht der Standorte. Die IDs können zur Verknüpfung von...

[csv](#) [dxf](#) [gpkg](#) [json](#) [shp](#) [wfs](#) [wms](#) [wmts](#)

- CSV:
Comma-Separated Values.
- DXF: Drawing Interchange File Format.
- GPKG: GeoPackage.
- JSON: GeoJSON / JSON.
- SHP: Shapefile.
- WFS: Web Feature Service.
- WMS: Web Map Service.
- WMTS: Web Map Tile Service.

Szenario: Auf der Suche nach Daten

- CSV: Comma-Separated Values.
- DXF: Drawing Interchange File Format.
- GPKG: GeoPackage.
- JSON: GeoJSON / JSON.
- SHP: Shapefile.
- WFS: Web Feature Service.
- WMS: Web Map Service.
- WMTS: Web Map Tile Service.

Standorte ZüriVelo

Offizielle Standorte der Veloverleihstationen von Züri Velo (PubliBike) Zweck: Dieser Datensatz dient zur Übersicht der Standorte. Die IDs können zur Verknüpfung von...

[csv](#) [dxf](#) [gpkg](#) [json](#) [shp](#) [wfs](#) [wms](#) [wmts](#)

Leider ist die Zeit zu kurz für GEO-Daten

Szenario: Auf der Suche nach Daten

- CSV: Comma-Separated Values.
- DXF: Drawing Interchange File Format.
- GPKG: GeoPackage.
- JSON: GeoJSON / JSON.
- SHP: Shapefile.
- WFS: Web Feature Service.
- WMS: Web Map Service.
- WMTS: Web Map Tile Service.

Standorte ZüriVelo

Offizielle Standorte der Veloverleihstationen von Züri Velo (PubliBike) Zweck: Dieser Datensatz dient zur Übersicht der Standorte. Die IDs können zur Verknüpfung von...

[csv](#) [dxf](#) [gpkg](#) [json](#) [shp](#) [wfs](#) [wms](#) [wmts](#)

Leider ist die Zeit zu kurz für GEO-Daten.

Szenario: Auf der Suche nach Daten

Von GeoJSON erhalten wir folgenden Link:

[https://www.ogd.stadt-zuerich.ch/wfs/geoportal/Standorte_ZueriVelo?
service=WFS&version=1.1.0&request=GetFeature&outputFormat=GeoJSON&typ
ename=view_zuerivelocommon_publibike](https://www.ogd.stadt-zuerich.ch/wfs/geoportal/Standorte_ZueriVelo?service=WFS&version=1.1.0&request=GetFeature&outputFormat=GeoJSON&typename=view_zuerivelocommon_publibike)

```
// 20240117162054
// https://www.ogd.stadt-zuerich.ch/wfs/geoportal/Standorte_ZueriVelo?
// service=WFS&version=1.1.0&request=GetFeature&outputFormat=GeoJSON&typename=view_zuerivelocommon_publibike

{
  "type": "FeatureCollection",
  "bbox": [
    8.46781,
    47.325159,
    8.650765,
    47.452387
  ],
  "features": [
    {
      "geometry": {
        "coordinates": [
          8.547903,
          47.384902
        ],
        "type": "Point"
      },
      "id": "view_zuerivelocommon_publibike.1",
      "properties": {
        "adresse": "Culmannstrasse 100",
        "datum": "2024-01-17T00:00:03.000",
        "name": "Publibike"
      }
    }
  ]
}
```

Szenario: Auf der Suche nach Daten

Ganz schön kompliziert... lässt uns nach MetaDaten suchen:

Standorte ZüriVelo

Offizielle Standorte der Veloverleihstationen von Züri Velo (PubliBike)

Datensatz frei von Nutzungsbeschränkungen

[Komplette Metadaten ansehen](#)

Szenario: Auf der Suche nach Daten

Ganz schön kompliziert... lässt uns nach MetaDaten suchen:

Standorte ZüriVelo

Offizielle Standorte der Veloverleihstationen von Züri Velo (PubliBike)

Datensatz frei von Nutzungsbeschränkungen

[Komplette Metadaten ansehen](#)

<https://www.geocat.ch/geonetwork/srv/ger/catalog.search#/metadata/a7e477f1-75ce-46d2-a32a-c547e37ccfaf/formatters/xsl-view?root=div&view=advanced>

Szenario: Auf der Suche nach Daten



```
1      "id": "view_zuerivelopublibike.9",
2      "properties": {
3          "adresse": "Bahnhof Stettbach",
4          "datum": "2024-01-17T00:00:03.000",
5          "id1": 353643.0,
6          "id_publibike": 514.0,
7          "lat": 47.397885,
8          "lon": 8.596202,
9          "name": "Bahnhof Stettbach",
10         "objectid": 9.0,
11         "plz": 8051,
12         "stadt": "Zürich",
13         "status": "Active"
14     },
15     "type": "Feature"
16 },
17 {
18     "geometry": {
19         "coordinates": [
20             8.549831,
21             47.421914
22         ],
23         "type": "Point"
24     },
```

Fragen:

- Wo steht das Fahrrad wir?
- Von wann sind die Daten?
- Was für eine ID hat das Objekt?

Szenario: Auf der Suche nach Daten

Fazit ist, Daten zu finden, ist nicht besonders schwer.

Nun geht es ans **einlesen und bearbeiten**.

Datenformate

Wir haben bereits gelernt, wie wir Web-Ressourcen abfragen können.
Tatsächlich haben wir mit Soup & request XML-Dateien (HTML) angefragt.

Nun wollen wir uns aber mit csv und json beschäftigen.

CSV

CSV steht für comma-separated-values und ist ein Dateiformat. Die Endung des Dateinamens lautet .csv.

Grob gesagt: Jedem "Datensatz" in der Tabelle entspricht eine Textlinie in der Textdatei. Jedes Feld ist durch ein Komma vom nächsten getrennt.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Der Code beinhaltet viel zu besprechen, bitte schaut diesen für 1–2 min an.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.'
```

Frage 0:

Warum müssen wir csv nicht installieren?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 0:

Warum müssen wir csv nicht installieren?

Antwort:

Weil es Teil der Python Standardbibliothek ist.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 1:

Warum wird hier nicht mit "öffnen, bearbeiten und schliessen" gearbeitet stattdessen mit "with"?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 1:

Warum wird hier nicht mit "öffnen, bearbeiten und schliessen" gearbeitet stattdessen mit "with"?

Antwort:

Die Schreibweise ist einfacher und es wird sicher geschlossen.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 2:

Was bedeutet der delimiter?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 2:

Was bedeutet der delimiter?

Antwort:

CSV-Files müssen nicht zwangsläufig mit , getrennt werden - andere Trennzeichen sind auch möglich.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 3:

Was bedeutet f' und \t?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 3:

Was bedeutet f' und \t?

Antwort:

f' ist die Formatierungsfunktion von Python \t ist ein Tab oder Indent.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 4:

Wie genau wird die Datei abgearbeitet?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 4:

Wie genau wird die Datei abgearbeitet?

Antwort:

Die Datei wird Linienweise abgearbeitet, mit einer Sonderbehandlung für die erste Zeile.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {" ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 5:

Ist `row[0]` und `row[2]` nicht ein wenig Fehleranfällig?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 00_employee_birthday.py
2
3 import csv
4
5 with open('00_employee_birthday.csv') as csv_file:
6     csv_reader = csv.reader(csv_file, delimiter=',')
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12         else:
13             print(f'\t{row[0]} works in the {row[1]} department, '
14                   f'and was born in {row[2]}.')
15             line_count += 1
16     print(f'Processed {line_count} lines.')
```

Frage 5:

Ist `row[0]` und `row[2]` nicht ein wenig Fehleranfällig?

Antwort:

Doch, hierfür müssen wir genau wissen, an welcher Position unsere Informationen stehen.

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 01_employee_birthday_dict.py
2
3 import csv
4
5 with open('00_employee_birthday.csv', mode='r') as csv_file:
6     csv_reader = csv.DictReader(csv_file)
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12             print(f'\t{row["name"]} works in the {row["department"]} '
13                   f'department, and was born in {row["birthday month"]}.')
14             line_count += 1
15             print(f'Processed {line_count} lines.'
```

Auf die Frage von vorhin wollen wir nun einiges verbessern, was wurde angepasst?

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 01_employee_birthday_dict.py
2
3 import csv
4
5 with open('00_employee_birthday.csv', mode='r') as csv_file:
6     csv_reader = csv.DictReader(csv_file)
7     line_count = 0
8     for row in csv_reader:
9         if line_count == 0:
10             print(f'Column names are {", ".join(row)}')
11             line_count += 1
12             print(f'\t{row["name"]} works in the {row["department"]} '
13                   f'department, and was born in {row["birthday month"]}.')
14             line_count += 1
15             print(f'Processed {line_count} lines.'
```

Auf die Frage von vorhin wollen wir nun einiges verbessern, was wurde angepasst?

Es wurde DictReader verwendet, welcher die csv-Datei wie ein Dictionary behandelt.

Entsprechend wurde aus row[0] auch das sprechende row[name].

CSV

Zusatzinformationen:

- **delimiter** gibt das Zeichen an, das zur Trennung der einzelnen Felder verwendet wird.
 - Die Vorgabe ist das Komma (',').
- **quotechar** gibt das Zeichen an, das verwendet wird, um Felder zu umschließen, die das Begrenzungszeichen enthalten.
 - Die Vorgabe ist ein doppeltes Anführungszeichen ("").
- **escapechar** gibt das Zeichen an, mit dem das Begrenzungszeichen umschlossen wird, falls keine Anführungszeichen verwendet werden.
 - Die Vorgabe ist kein Escape-Zeichen.

CSV

Mode:

Specify File Mode

Here are five different modes you can use to open the file:

Character	Mode	Description
'r'	Read (default)	Open a file for read only
'w'	Write	Open a file for write only (overwrite)
'a'	Append	Open a file for write only (append)
'r+'	Read+Write	open a file for both reading and writing
'x'	Create	Create a new file

Python File Modes

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 02_employee_birthday_writing.py
2
3 import csv
4
5 with open('02_employee_birthday.csv', mode='w') as employee_file:
6     employee_writer = csv.writer(employee_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
7
8     employee_writer.writerow(['John Smith', 'Accounting', 'November'])
9     employee_writer.writerow(['Erica Meyers', 'IT', 'March'])
```

Nun wollen wir auch noch in das File schreiben. Hier ist es wichtig, dass ein schreibfähiger mode verwendet wird - siehe letzte Slide.

CSV - Aufgabe



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```

Wir wollen uns nun mit 03-exercises/00_own_birthday_writing beschäftigen.

Kopieren Sie bitte das .csv und .py in ihren userfolder.

Fragen:

- Was passiert, wenn die Datei ausgeführt wird (mit den Daten)?
- Mit welchem **mode**, hätte man das verhindern können? (Datei einfach nochmals kopieren)
- Bitte fügen Sie nochmals 3 Personen hinzu

CSV



```
1 name,department,birthday month
2 John Smith,Accounting,November
3 Erica Meyers,IT,March
```



```
1 # 03_employee_birthday_writing_dict.py
2
3 import csv
4
5 with open('employee_file2.csv', mode='w') as csv_file:
6     fieldnames = ['emp_name', 'dept', 'birth_month']
7     writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
8
9     writer.writeheader()
10    writer.writerow({'emp_name': 'John Smith', 'dept': 'Accounting', 'birth_month': 'November'})
11    writer.writerow({'emp_name': 'Erica Meyers', 'dept': 'IT', 'birth_month': 'March'})
```

Natürlich geht dies auch mit DictWriter (dem Gegenüber von DictReader) und das schöne ist, nun können Dictionaries übergeben werden.

CSV



```
1 with open('test.csv', 'w') as csvfile:  
2     fieldnames = test_dict.keys()  
3     fieldvalues = zip(*test_dict.values())  
4  
5     writer = csv.writer(csvfile)  
6     writer.writerow(fieldnames)  
7     writer.writerows(fieldvalues)
```

Mehrere Zeilen sind hier nur mit Umwegen möglich, hierfür werden wir später Pandas verwenden.

JSON

JSON heisst zwar **JavaScript Object Notation**, ist aber ein komplett unabhängiges Datenformat ohne Bindung an die JavaScript-Sprache.

Gleichzeitig ist JSON mit seiner sehr einfachen Struktur und der Kodierung im Unicode Zeichensatz das ideale Format, um Daten zwischen Systemen auszutauschen.

JSON

```
1. {
2.     "customers": [
3.         {
4.             "firstName": "Max",
5.             "middleInitial": "M",
6.             "lastName": "Mustermann",
7.             "gender": "M",
8.             "age": 24,
9.             "address": {
10.                 "streetAddress": "Nordring 98",
11.                 "city": "Nürnberg",
12.                 "state": "Bayern",
13.                 "postalCode": 90409
14.             },
15.             "phoneNumbers": [
16.                 { "type": "home", "number": "0911/161803399" }
17.             ]
18.         },
19.         ,
20.         {
21.             "firstName": "Erika",
22.             "lastName": "Gabler",
23.             "gender": "F",
24.             "age": 57,
25.             "address": {
26.                 "streetAddress": "Neuer Wall 72",
27.                 "city": "Hamburg",
28.                 "state": "Hamburg",
29.                 "postalCode": 20354
30.             },
31.             "phoneNumbers": [
32.                 { "type": "home", "number": "040/27182818" }
33.                 , { "type": "mobil", "number": "0176/31415926" }
34.             ]
35.         }
36.     ]
37. }
```

JSON



```
1  {
2      "firstName": "Jane",
3      "lastName": "Doe",
4      "hobbies": ["running", "sky diving", "singing"],
5      "age": 35,
6      "children": [
7          {
8              "firstName": "Alice",
9              "age": 6
10         },
11         {
12             "firstName": "Bob",
13             "age": 8
14         }
15     ]
16 }
```

Wie wir sehen können, unterstützt JSON primitive Typen wie Zeichenketten und Zahlen sowie verschachtelte Listen und Objekte.

JSON

```
1  {
2      "firstName": "Jane",
3      "lastName": "Doe",
4      "hobbies": [ "running", "sky diving", "singing"],
5      "age": 35,
6      "children": [
7          {
8              "firstName": "Alice",
9              "age": 6
10         },
11         {
12             "firstName": "Bob",
13             "age": 8
14         }
15     ]
16 }
```

Moment, das sieht aus wie ein Python-Dictionary! Ich weiss, nicht wahr?

Das liegt daran, dass es sich um eine Form der UON (Universal Object Notation) handelt, sind also sehr ähnlich zueinander.

JSON

Python

dict

list, tuple

str

int, long, float

True

False

None

JSON

object

array

string

number

true

false

null

JSON

Wenn wir nun ein Python-Objekt haben:

```
● ● ●  
1 # 04_json_write.py  
2  
3 data = {  
4     "president": {  
5         "name": "Zaphod Beeblebrox",  
6         "species": "Betelgeusian"  
7     }  
8 }
```

Können wir dieses einfach serialisieren

```
● ● ●  
1 with open("json_write.json", "w") as write_file:  
2     json.dump(data, write_file)
```

JSON

Wenn wir nun ein Python-Objekt haben:

```
● ● ●  
1 data = {  
2     "president": {  
3         "name": "Zaphod Beeblebrox",  
4         "species": "Betelgeusian"  
5     }  
6 }
```

und dieses soll direkt weiterverwendet werden

```
● ● ●  
1 json_string = json.dumps(data)
```

JSON

Natürlich können wir die JSON-Datei auch wieder deserialisieren:

```
1 # 05_json_read.py
2
3 import json
4
5 with open("04_json_write.json", "r") as read_file:
6     data = json.load(read_file)
7
8 print(data)
9 print(data['president']['species'])
```

JSON - Aufgabe

Nun ist es Zeit für ein echtes Beispiel, bitte erstellt eine neue Datei im userfolder (Name Beispielsweise: json_example.py)



```
1 import json
2 import requests
```



```
1 response = requests.get("https://jsonplaceholder.typicode.com/todos")
2 todos = json.loads(response.text)
```



```
1 type(todos)
2 todos[:10]
```

JSON - Aufgabe Teil 2

Erweitere json_example.py nun so das die ersten 20 Todos in eine Datei geschrieben werden (json_example.json).

Pandas

Sobald wir mit grösseren Datenmengen zu tun haben, oder sehr flexibel von csv, auf json auf excel wechsel müssen, kommt das Paket Pandas ins Spiel.

Hinweis: Pandas wird sehr grosszügig in APPD erklärt, ich verwende es hier nur als Mittel zum Daten einlesen.

Pandas

Diesmal geht es um ein HR-Datenset:



```
1 Name,HireDate,Salary,SickDaysremaining
2 Graham Chapman,03/15/14,50000.00,10
3 John Cleese,06/01/15,65000.00,8
4 Eric Idle,05/12/14,45000.00,10
5 Terry Jones,11/01/13,70000.00,3
6 Terry Gilliam,08/12/14,48000.00,7
7 Michael Palin,05/23/13,66000.00,8
```



```
1 # 06_read_pandas.py
2
3 import pandas
4
5
6 df = pandas.read_csv('06_hrdata.csv')
7 print(df)
```

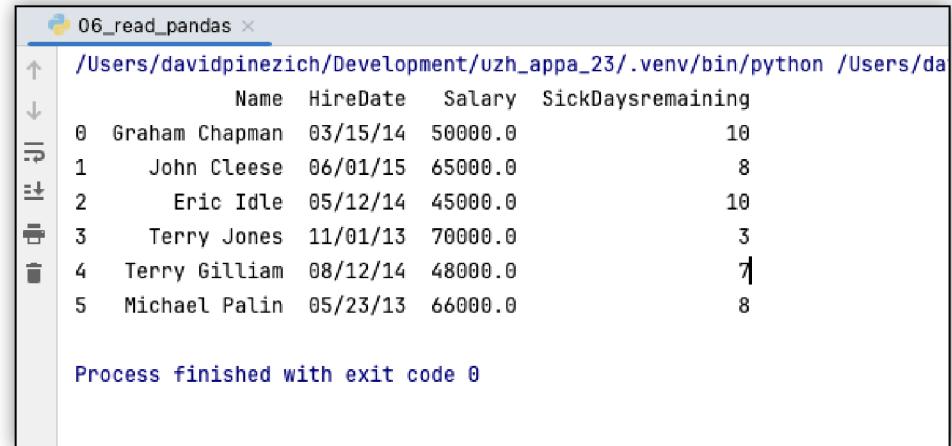
Pandas

Nun erhalten wir ein sogenanntes DataFrame

```
06_read_pandas ×  
/Users/davidpinezich/Development/uZH_APPA_23/.venv/bin/python /Users/davidpinezich/Development/uZH_APPA_23/06_read_pandas.py  
          Name    HireDate   Salary  SickDaysremaining  
0  Graham Chapman  03/15/14  50000.0                  10  
1      John Cleese  06/01/15  65000.0                  8  
2       Eric Idle  05/12/14  45000.0                 10  
3     Terry Jones  11/01/13  70000.0                  3  
4  Terry Gilliam  08/12/14  48000.0                  1  
5  Michael Palin  05/23/13  66000.0                  8  
  
Process finished with exit code 0
```

Das ist zum einen **hocheffizient**, zum anderen **leicht bedienbar**.

Pandas



The screenshot shows a Jupyter Notebook cell with the title "06_read_pandas". The code executed is a Python command to read a CSV file. The output displays a DataFrame with six rows and five columns: Name, HireDate, Salary, SickDays, and remaining. The data is as follows:

	Name	HireDate	Salary	SickDays	remaining
0	Graham Chapman	03/15/14	50000.0	10	
1	John Cleese	06/01/15	65000.0	8	
2	Eric Idle	05/12/14	45000.0	10	
3	Terry Jones	11/01/13	70000.0	3	
4	Terry Gilliam	08/12/14	48000.0	7	
5	Michael Palin	05/23/13	66000.0	8	

At the bottom of the output, it says "Process finished with exit code 0".

- Zunächst erkannte Pandas, dass die erste Zeile der CSV-Datei Spaltennamen enthält, und verwendete diese automatisch.
- Allerdings verwendet Pandas auch null-basierte Integer-Indizes im DataFrame. Das liegt daran, dass wir ihm nicht gesagt haben, **wie unser Index heißen** soll.
- Wenn man sich die Datentypen unserer Spalten ansieht, wird man feststellen, dass Pandas die Spalten Gehalt und verbleibende Krankheitstage ordnungsgemäss in Zahlen umgewandelt hat, die Spalte Einstellungsdatum ist jedoch immer noch ein String.

Pandas

Nun wollen wir etwas abfragen:



```
1 print(type(df['Hire Date'][0]))
```

Oder mit einem anderen Index strukturieren:



```
1 # 07_hrdata_name.csv
2
3 import pandas
4
5
6 df = pandas.read_csv('06_hrdata.csv', index_col='Name')
7 print(df)
```

Pandas

Und zusätzlich noch Daten parsen:

```
1 # 08_hrdata_parsed.py
2
3 import pandas
4
5
6 df = pandas.read_csv('06_hrdata.csv',
7                     index_col='Employee',
8                     parse_dates=['Hired'],
9                     header=0,
10                    names=['Employee', 'Hired', 'Salary', 'Sick Days'])
11 print(df)
```

Pandas

Und zum Schluss erstellen wir ganz einfach ein JSON daraus:

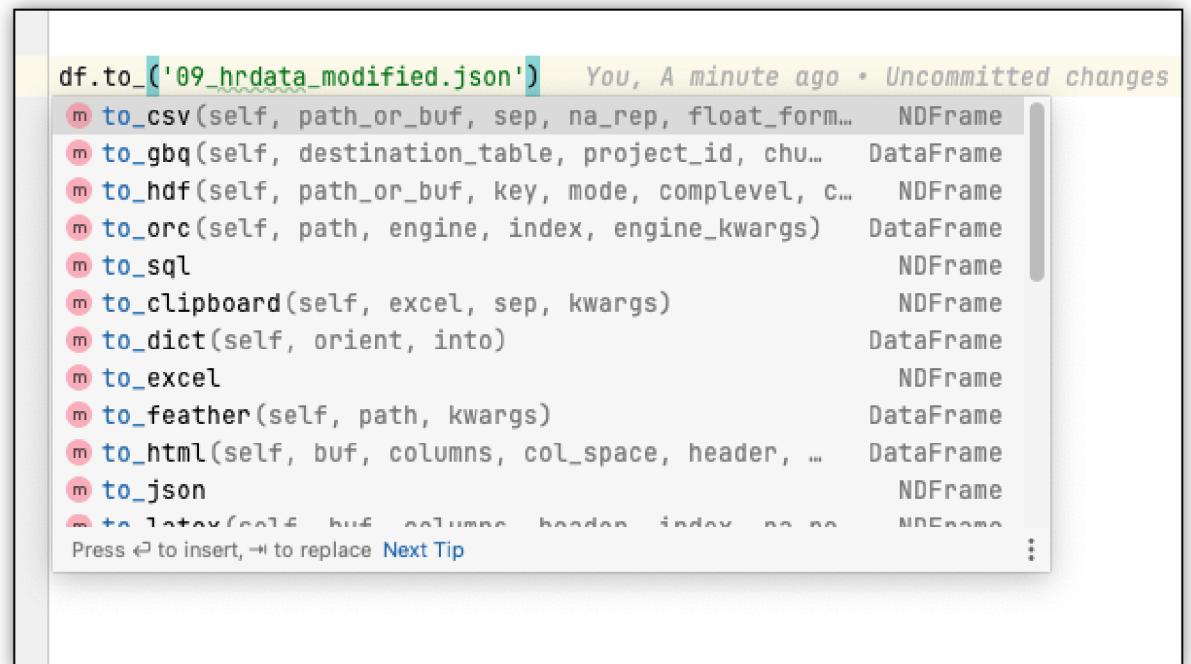
```
1 # 09_hrdata_to_json.py
2
3 import pandas
4
5 df = pandas.read_csv('06_hrdata.csv',
6                      index_col='Employee',
7                      parse_dates=['Hired'],
8                      header=0,
9                      names=['Employee', 'Hired', 'Salary', 'Sick Days'])
10
11 df.to_json('09_hrdata_modified.json')
```

Pandas

Doch damit nicht genug, kopiert bitte die Datei 09_hrdata_to_json in euren userfolder.

Versucht ein Excel-File und ein HTML-File zu erstellen.

Für Excel braucht es noch das Paket openpyxl.



```
df.to_('09_hrdata_modified.json')  You, A minute ago • Uncommitted changes
m to_csv(self, path_or_buf, sep, na_rep, float_form... NDFrame
m to_gbq(self, destination_table, project_id, chu... DataFrame
m to_hdf(self, path_or_buf, key, mode, complevel, c... NDFrame
m to_orc(self, path, engine, index, engine_kwarg... DataFrame
m to_sql
m to_clipboard(self, excel, sep, kwargs)
m to_dict(self, orient, into)
m to_excel
m to_feather(self, path, kwargs)
m to_html(self, buf, columns, col_space, header, ...
m to_json
m to_lxml(buf, columns, header, index, na_na, NDFrame
Press ⇧ to insert, ⇩ to replace Next Tip
```

Flask

Aus der letzten Aufgabe haben wir ein wunderbares HTML erhalten.

```
<table border="1" class="dataframe">
  <thead>
    <tr style="...">
      <th></th>
      <th>Hired</th>
      <th>Salary</th>
      <th>Sick Days</th>
    </tr>
    <tr>
      <th>Employee</th>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Graham Chapman</th>
      <td>2014-03-15</td>
      <td>50000.0</td>
      <td>10</td>
    </tr>
    <tr> You, 3 minutes ago • Uncommitted changes
  </tbody>
</table>
```

Flask

Aus der letzten Aufgabe haben wir ein wunderbares HTML erhalten.

Dies können wir auch einfach im Browser ansehen:



Employee	Hired	Salary	Sick Days
Graham Chapman	2014-03-15	50000.0	10
John Cleese	2015-06-01	65000.0	8
Eric Idle	2014-05-12	45000.0	10
Terry Jones	2013-11-01	70000.0	3
Terry Gilliam	2014-08-12	48000.0	7
Michael Palin	2013-05-23	66000.0	8

```
<table border="1" class="dataframe">
  <thead>
    <tr style="...">
      <th></th>
      <th>Hired</th>
      <th>Salary</th>
      <th>Sick Days</th>
    </tr>
    <tr>
      <th>Employee</th>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Graham Chapman</td>
      <td>2014-03-15</td>
      <td>50000.0</td>
      <td>10</td>
    </tr>
    <tr> You, 3 minutes ago • Uncommitted changes
```

Flask

Wir verwenden dieses Beispiel gleich um uns mit Flask vertraut zu machen.

Flask ist ein Web-Framework. Das bedeutet, dass Flask uns Werkzeuge, Bibliotheken und Technologien zur Verfügung stellt, mit denen man eine Webanwendung erstellen kann.

Diese Webanwendung kann aus ein paar Webseiten, einem Blog oder einem Wiki bestehen oder sogar so gross sein wie eine webbasierte Kalenderanwendung oder eine kommerzielle Website.

Flask



```
1 # 10_flask_starter.py
2
3 import flask
4
5 # Create the application.
6 APP = flask.Flask(__name__)
7
8 @APP.route('/')
9 def index():
10     """ Displays the index page accessible at '/' found in templates
11     """
12     return flask.render_template('10_hrdata_modified.html')
13
14
15 if __name__ == '__main__':
16     APP.debug=True
17     APP.run()
```

Flask

```
P
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 342-442-388
127.0.0.1 - - [18/Jan/2024 00:36:27] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Jan/2024 00:36:27] "GET /favicon.ico HTTP/1.1" 404 -
 * Detected change in '/Users/davidpinezich/Development/uzh_appa_23/data-automation/02-examples/10_flask_starter.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 342-442-388
 * Detected change in '/Users/davidpinezich/Development/uzh_appa_23/data-automation/02-examples/10_flask_starter.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 342-442-388
```

Ausprobieren ist angesagt, auf unserem lokalen Server

Flask

Diese Webseite könnte man nun mittels DigitalOcean oder PythonAnywhere für ca. 5 USD pro Monat problemlos ins Internet stellen.

<https://www.pythonanywhere.com/>

<https://www.digitalocean.com/>

Flask

So können wir ganz einfach die Resultate unsere Automation ins Internet stellen, wenn gewünscht.

Excel

Als Nächstes wollen wir uns auch noch mit Excel beschäftigen, denn viele Daten sind (und werden auch in Zukunft) in Excel sein.

Excel - Aufgabe

Mit dem Paket openpyxl ist der Umgang mit Excel überraschend einfach.

```
1 # 11_data_from_excel.py
2
3 import openpyxl
4
5 wb = openpyxl.load_workbook('ch-bevoelkerung.xlsx')
6 wb.sheetnames # The workbook's sheets' names.
7 ['VZ RFP 1850', 'VZ RFP 1860', 'VZ RFP 1870']
8
9 sheet = wb['VZ RFP 1850'] # Get a sheet from the workbook.
10
11 type(sheet)
12 # <class 'openpyxl.worksheet.worksheet.Worksheet'>
13 print(sheet.title) # Get the sheet's title as a string.
14
15 print(sheet['C15'].value)
16 print(sheet['D15'].value)
```

Excel - Aufgabe

Versuchen Sie mal alle Gemeindenamen auszugeben:

Excel - Aufgabe

Versuchen Sie mal alle Gemeindenamen auszugeben:

```
1 # 12_data_from_excel_iterate.py
2
3 import openpyxl
4
5 wb = openpyxl.load_workbook('ch-bevoelkerung.xlsx')
6 wb.sheetnames # The workbook's sheets' names.
7 ['VZ RFP 1850', 'VZ RFP 1860', 'VZ RFP 1870']
8
9 sheet = wb['VZ RFP 1850'] # Get a sheet from the workbook.
10
11 type(sheet)
12 # <class 'openpyxl.worksheet.worksheet.Worksheet'>
13 print(sheet.title) # Get the sheet's title as a string.
14
15 print(sheet['C15'].value)
16 print(sheet['D15'].value)
17
18 for cell in sheet['C']:
19     print(cell.value)
```

Excel - Aufgabe

Versuchen Sie nun das Gleiche aber mit Pandas.



```
1 import pandas  
2  
3 df = pandas.read_excel(open('ch-bevoelkerung.xlsx', 'rb'),  
4                         sheet_name='VZ RFP 1850', index_col=2, header=2)
```

Daten visualisieren

Daten zu visualisieren, ist oft etwas, was aus unseren vorherigen Arbeiten folgt.

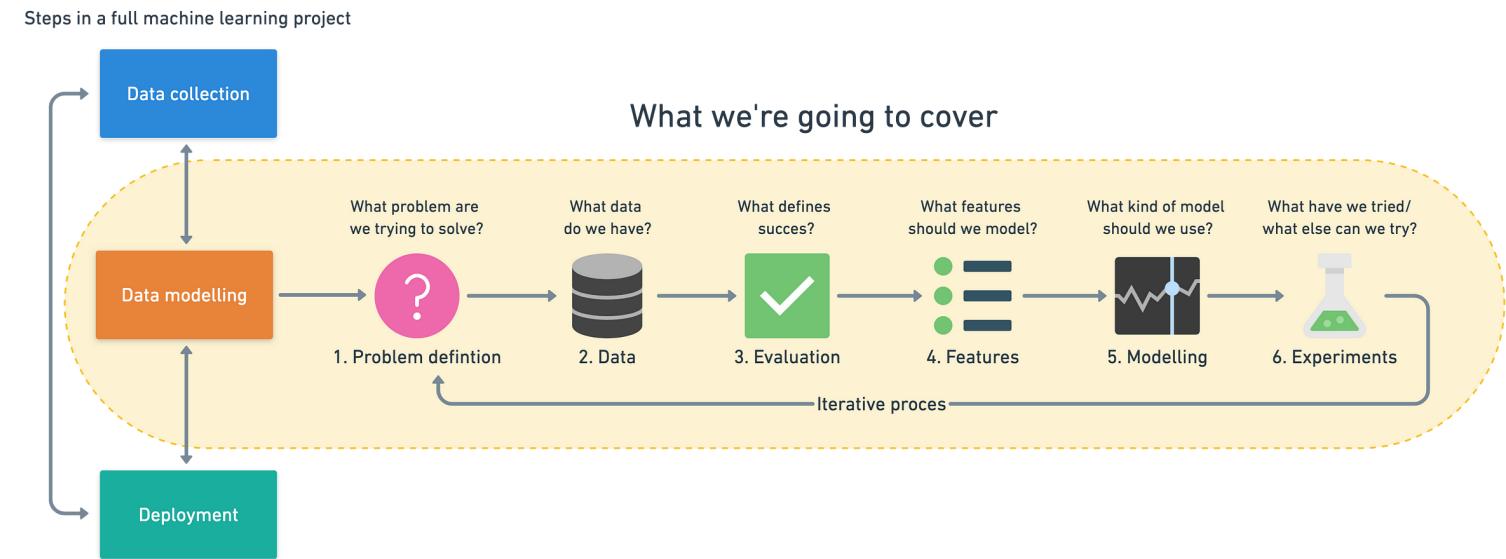
Das Thema ist sehr breit, und es folgt lediglich eine Übersicht.

Visualisieren sollte ein iterativer Prozess sein

1. Problemdefinition
2. Daten
3. Evaluation (wie definiere ich Erfolg)
4. Features (was brauche ich genau)
5. Visualisieren
6. Testen (Experimentieren)

Ein Schritt zurück ist jederzeit aufgrund neuer Erkenntnisse möglich (und nicht unbedingt schlecht)!

(Grober) Ablauf



Für eine Visualisierung kann man dies ebenfalls so angehen:

1. Problemdefinition
2. Daten
3. Evaluation (wie definiere ich Erfolg)
4. Features (was brauche ich genau)
5. Visualisieren
6. Testen (Experimentieren)

Dog-Science

Die Trendmarke für Ihren Hund



Problemstellung (Problemdefinition)

- Dog-Science ist eine Trendmarke aus dem asiatischen Raum und möchte gerne nach Zürich expandieren.
- Im Heimatland von Dog-Science ist genau bekannt, wo die meisten Hundehalter wohnen und welche Produkte bevorzugt werden. Die Schweiz und insbesondere Zürich ist aber unbekannt.
- Das Budget von Dog-Science ist limitiert, die Stadt Zürich hat aber (fiktiv) Daten und Unterstützung für einen Pop-up-Store zugesichert
- Können wir die Ausgangslage von Dog-Science verbessern?

Ein Erster «Blick» in die Daten

The screenshot shows the homepage of the Stadt Zürich Open Data portal. The top navigation bar includes the logo 'Stadt Zürich Open Data', links for 'Startseite', 'Datensätze', and 'Kategorien', and a search bar. Below the header, a breadcrumb trail indicates the current page: 'Hundebestand der Stadt Zürich'. On the left, there's a sidebar with a 'Lizenz' section showing 'Creative Commons CCZero' and a 'OPEN DATA' button. The main content area features a large title 'Hundebestand der Stadt Zürich'. A descriptive text explains that the dataset contains information about dog owners and their dogs from the city's dog register. Below this, a section titled 'Daten und Ressourcen' lists a CSV file named '20200306_hundehalter.csv' and a 'Entdecke +' button.

Hundebestand der Stadt Zürich

In diesem Datensatz finden Sie Angaben zu Hunden und deren Besitzerinnen und Besitzern aus dem aktuellen Bestand des städtischen Hunderegisters. Bei den hundehaltenden Personen sind Informationen zur Altersgruppe, dem Geschlecht und dem statistischen Quartier des Wohnorts angegeben. Zu jedem Hund ist die Rasse, der Rassetyp, das Geschlecht, das Geburtsjahr und die Farbe erfasst. Das Hunderegister wird von der Abteilung Hundekontrolle der Stadtpolizei Zürich geführt.

Daten und Ressourcen

20200306_hundehalter.csv
Comma-Separated Values. Weitere Informationen zu CSV finden Sie in unserer...

Entdecke +

Ein Erster «Blick» in die Daten

HALTER_ID	ALTER	GESCHLECHT	STADTKREIS	STADTQUARTIER	RASSE1	GEBURTSJAHR_HUND	GESCHLECHT_HUND	HUNDEFARBE
574	61-70	w		2	23 Mischling gross	2013	w	schwarz
695	41-50	m		6	63 Labrador Retriever	2012	w	braun
893	71-80	w		7	71 Mittelschnauzer	2010	w	schwarz
916	41-50	m		3	34 Mischling klein	2015	w	hellbraun
1177	51-60	m		10	102 Shih Tzu	2011	m	schwarz/weiss
4054	51-60	w		11	111 Lagotto Romagnolo	2016	w	weiss/beige
4135	41-50	w		9	91 Mischling klein	2016	w	schwarz
4206	71-80	w		8	82 Havaneser	2016	w	hellbraun/weiss
4281	61-70	w		9	91 Chihuahua	2011	w	hellbraun
4388	61-70	w		11	115 Mops	2006	m	beige
4726	51-60	m		5	52 Mischling gross	2007	m	schwarz
4726	51-60	m		5	52 Golden Retriever	2013	w	creme
4747	61-70	m		2	24 Chihuahua	2013	m	weiss/braun
4850	51-60	m		4	42 Chihuahua	2013	w	beige
4862	51-60	m		4	42 Mops	2006	m	braun
5040	61-70	m		10	102 Labrador Retriever	2016	w	gelb
5088	61-70	m		7	72 Labrador Retriever	2014	w	schwarz
5113	61-70	m		11	119 Beagle	2010	w	tricolor
5113	61-70	m		11	119 Chihuahua	2017	w	beige
5225	71-80	m		3	34 Lagotto Romagnolo	2007	m	braun
5227	71-80	m		10	101 Border Terrier	2011	w	tricolor

Ein Erster «Blick» in die Daten

- Die Daten haben eine gute, aber nicht perfekte Qualität
- Die Rassen wurden leider sehr ungenau definiert
 - Oft als «Mischling gross» oder «Mischling klein» definiert, aber nicht weiter ausdefiniert
 - Oft vertreten
 - Chihuahua 573
 - Labrador Retriever 426
 - Selten vertreten
 - Oesterreichischer Pinscher 1
 - Daisy-Dog 1

Ein Erster «Blick» in die Daten

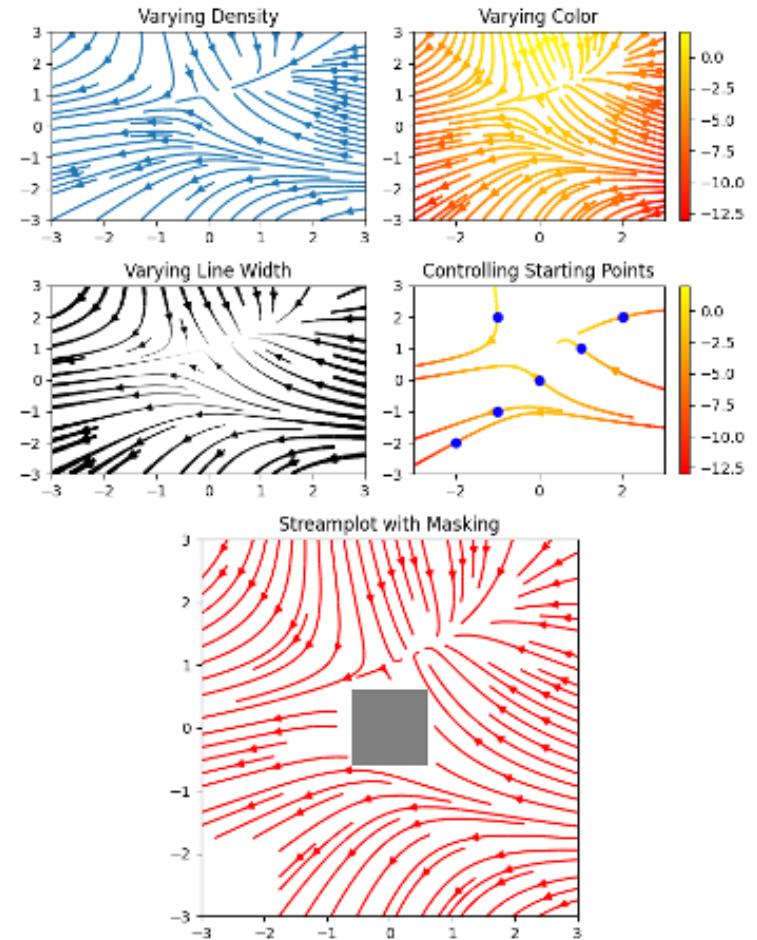
- Deutlich mehr weibliche als männliche Hunde (5402 zu 2439)
- Bei den Fell-Farben ist schwarz am meisten vertreten (800), danach tricolor (725) und weiss 634), seltener «schwarz melliert» oder «gelb / schwarz) je ein mal
- Das Hundalter schauen wir uns in den Visualisierungen an (bitte Ausreisser beachten)
- Wie ist die Hundehalter / Hund Ratio?
 - 6562 Hundehalter haben einen gemeldeten Hund
 - 581 Hundehalter haben 2 oder mehrere gemeldete Hunde
 - Der Top-Hundehalter besitzt 14 gemeldete Hunde

Visualisierung

- Viele Libraries mit unterschiedlichen Spezialitäten
 - Matplotlib: Für statische, animierte und interaktive Visualisierungen (eine der ältesten Bibliotheken)
 - Pygal: Dynamische SVG-Charting Bibliothek
 - Seaborn: Basiert auf Matplotlib und bietet ein high-level Interface für statistische Grafiken
 - Altair: Basiert auf Vega/Vega Lite und ist eine deklarative statistische Visualisierungs-Bibliothek
 - Ggplot2: System zur Erstellung von deklarativen Grafiken
 - Plotly: Interaktiv und vom User analysierbar
 - Bokeh: Bibliothek für interaktive Visualisierungen
 - Geoplotlib: Hauptsächlich für Karten

Matplotlib

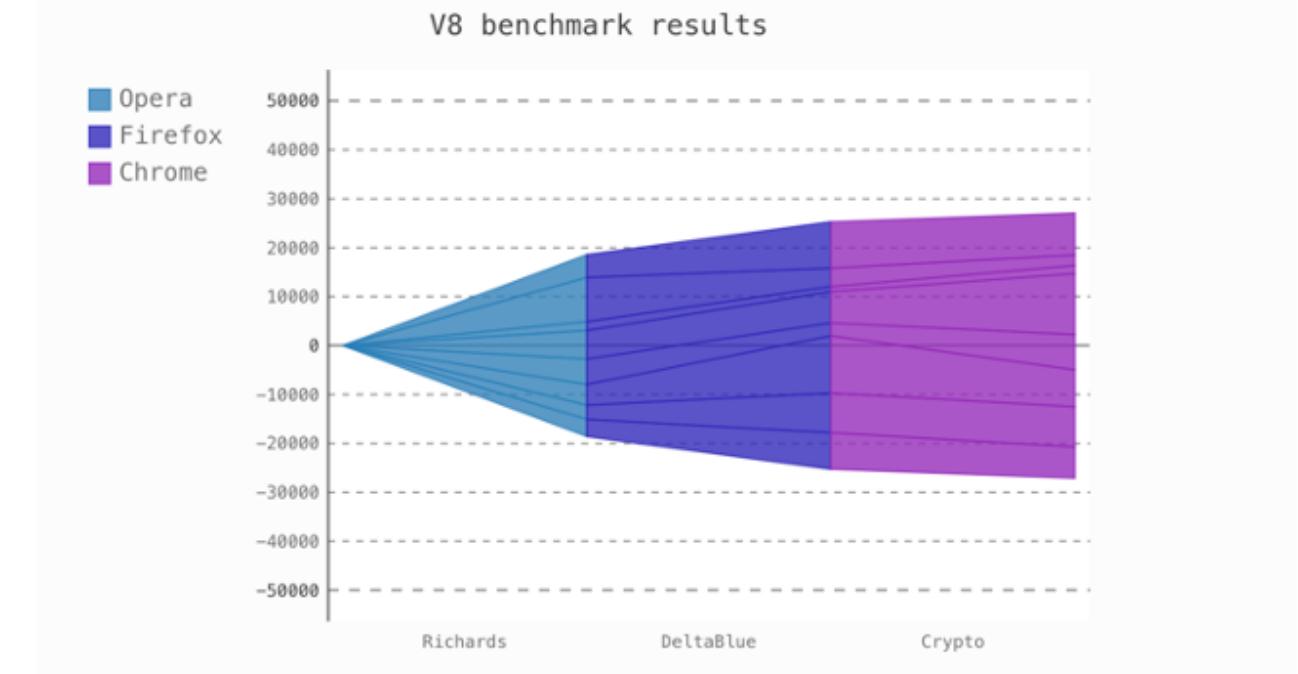
- Geeignet für einfache und komplexe Darstellungen
- Mehrere Darstellungen via Subplots möglich
- Bietet das grösste und allgemeinste Spektrum



Streamplot with various plotting options.

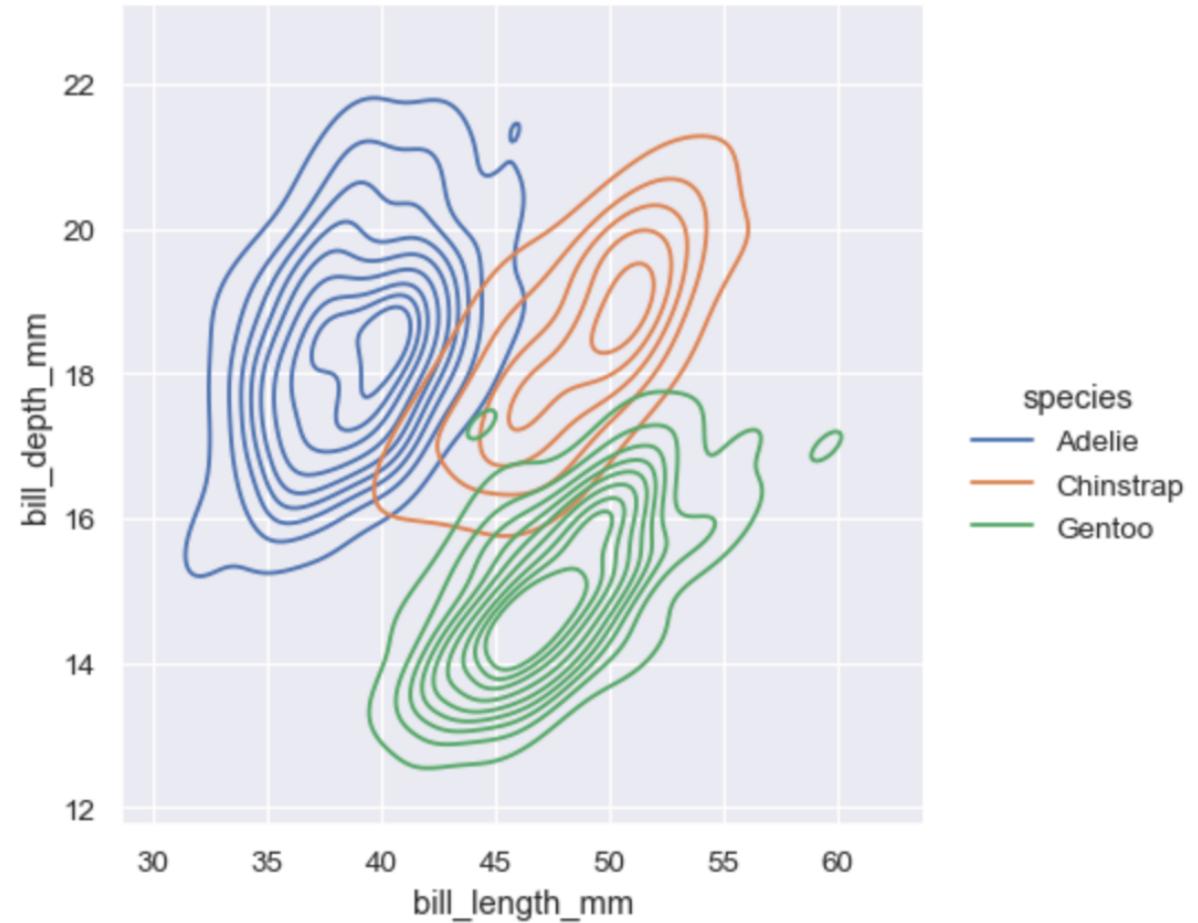
Pygal

- Wird leider nicht mehr (aktiv) weiterentwickelt
- Guter Funktionsumfange und dabei relativ einfach gelernt
Einfach als interaktives HTML auszugeben



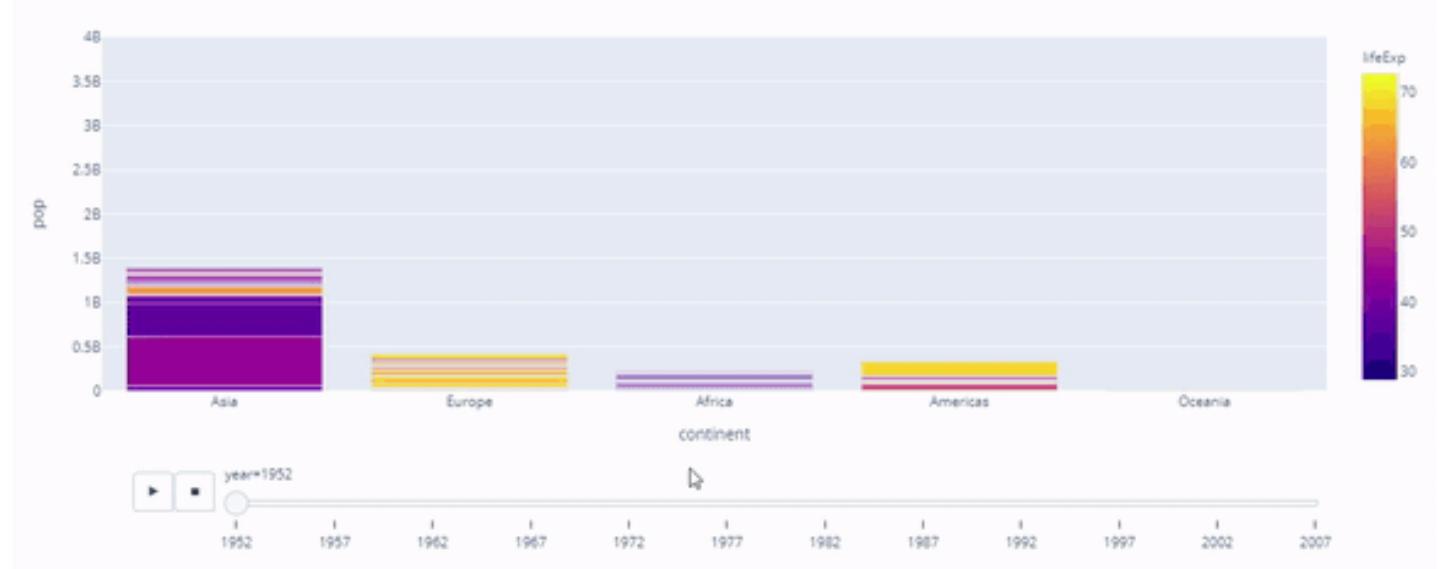
Seaborn

- Basiert auf Matplotlib
- Vor allem für statistische Visualisierungen geeignet
- Wird unser erstes Beispiel sein



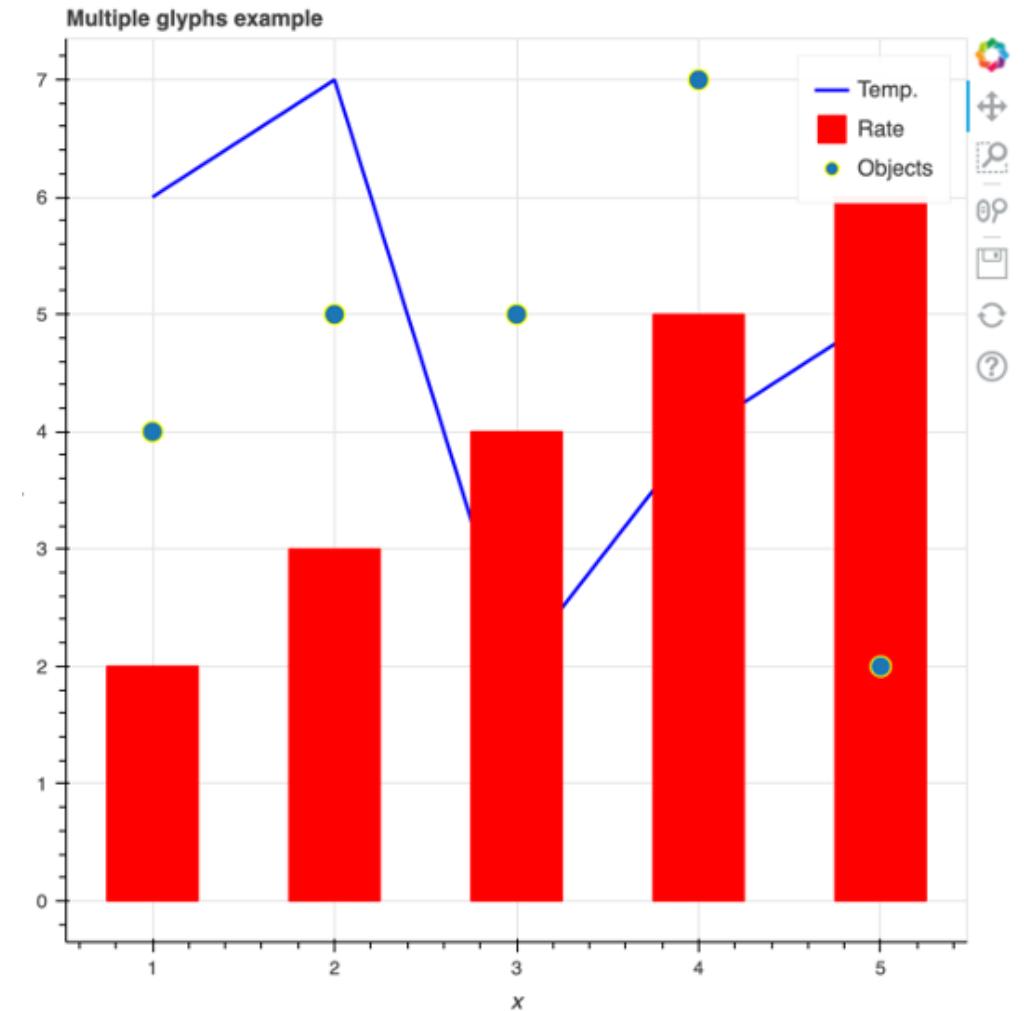
Plotly

- Bietet eine grosse Bandbreite an Charts
- Lässt sich animieren
- Besitzt «out of the box» Manipulationstools

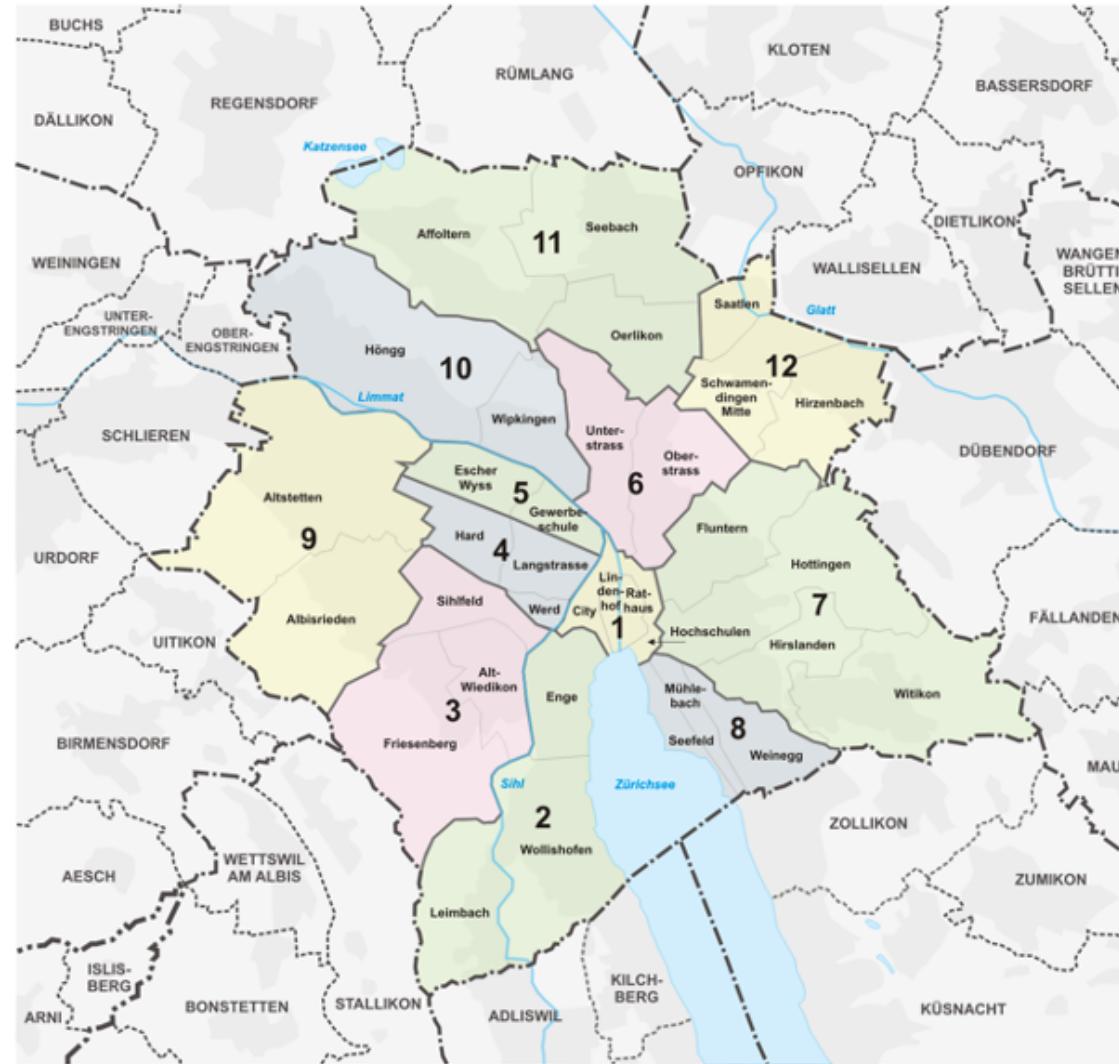


Bokeh

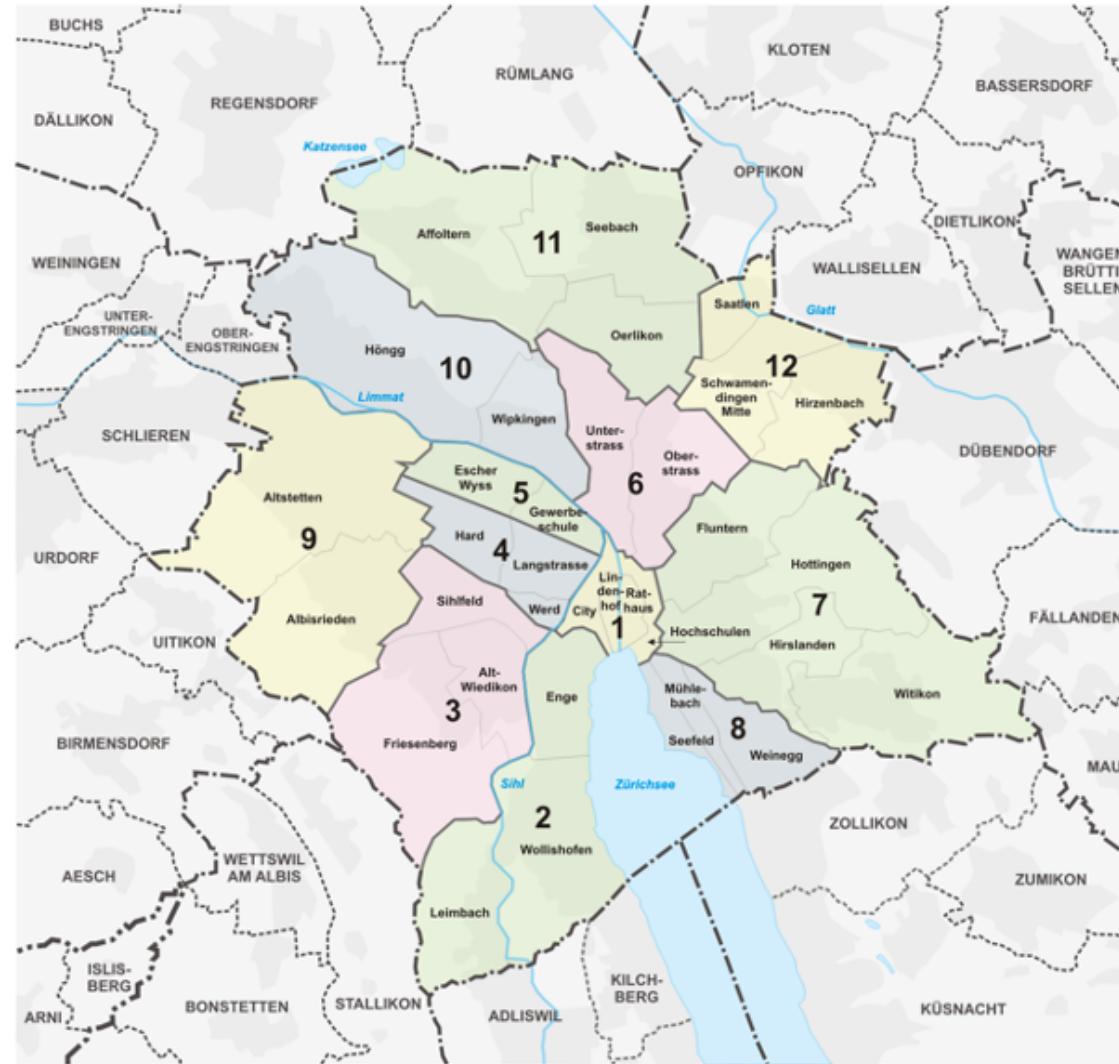
- Bietet eine grosse Bandbreite an interaktiven Charts
- Besitzt «out of the box» Manipulationstools



Der Ort ist wichtig...



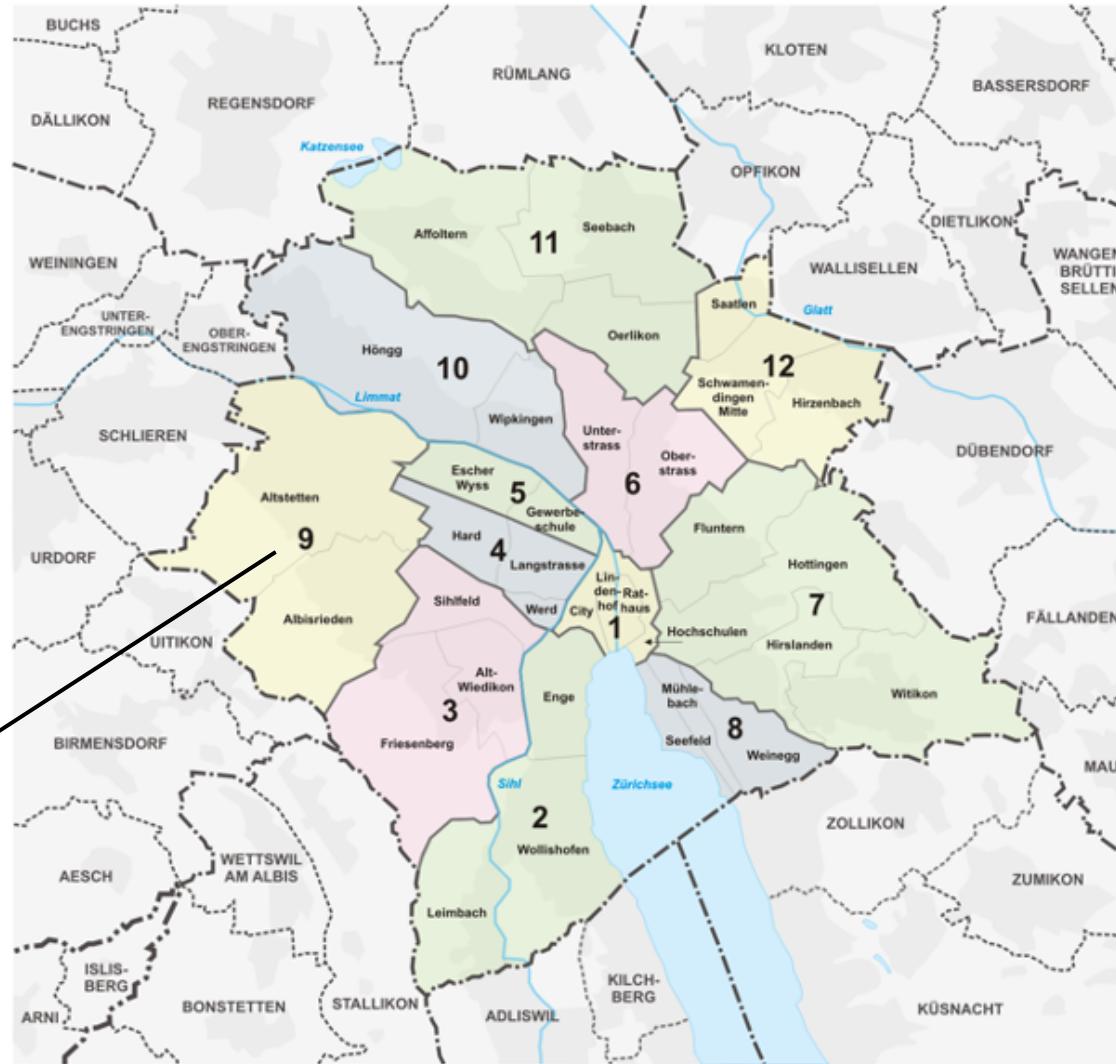
Der Ort ist wichtig...



Platz 3: 984

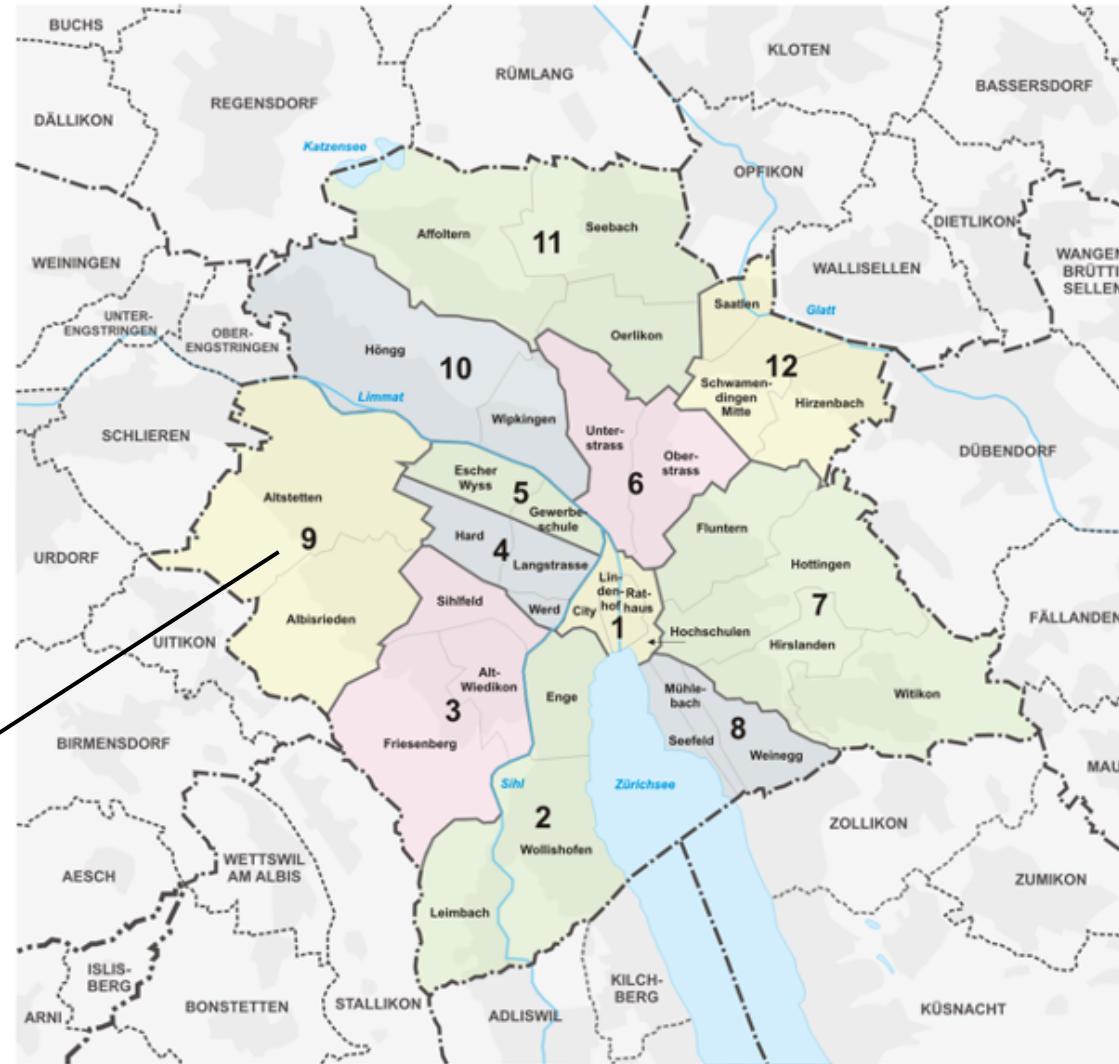
Der Ort ist wichtig...

Platz 3: 984



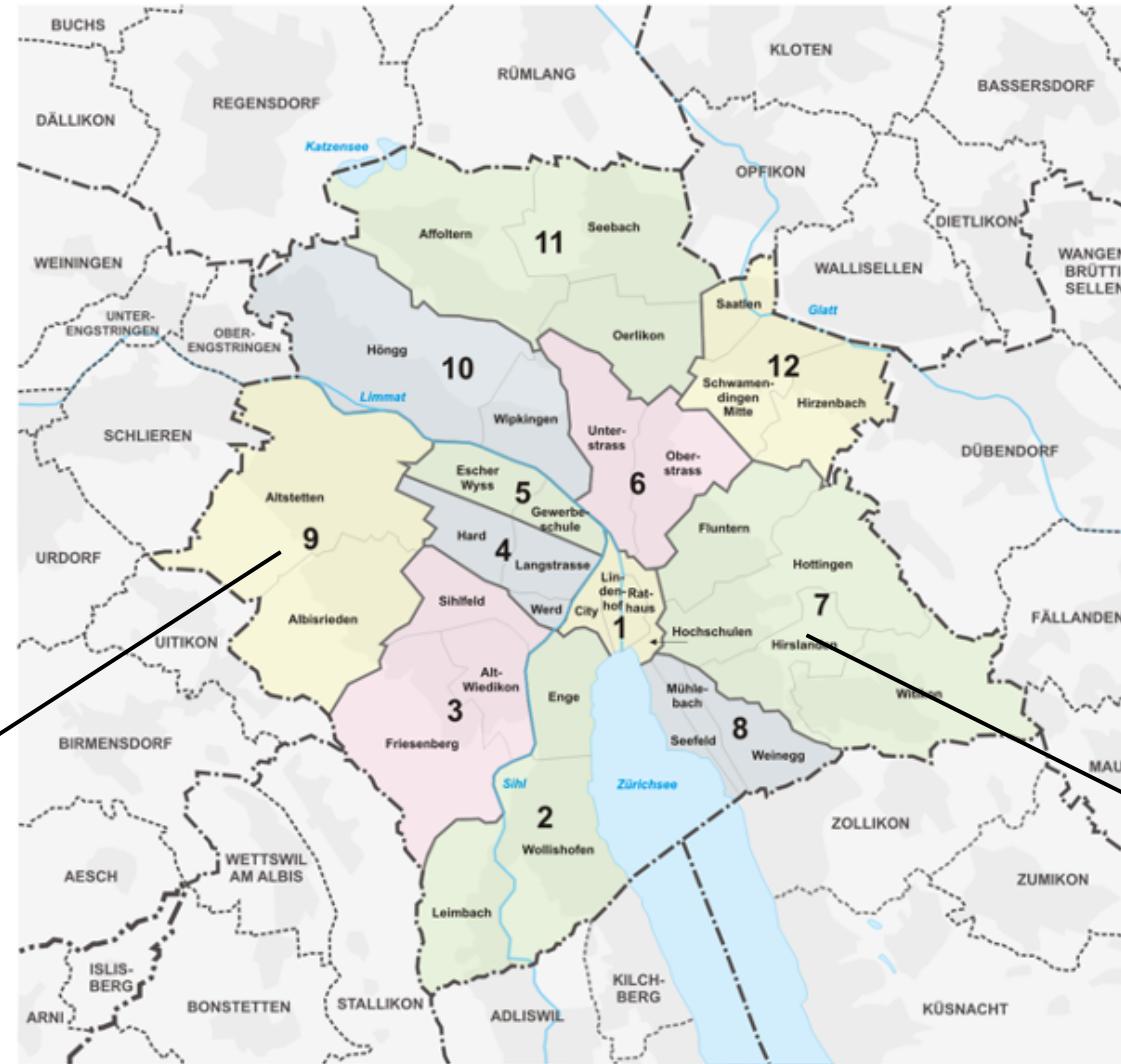
Der Ort ist wichtig...

Platz 3: 984



Platz 2: 1087

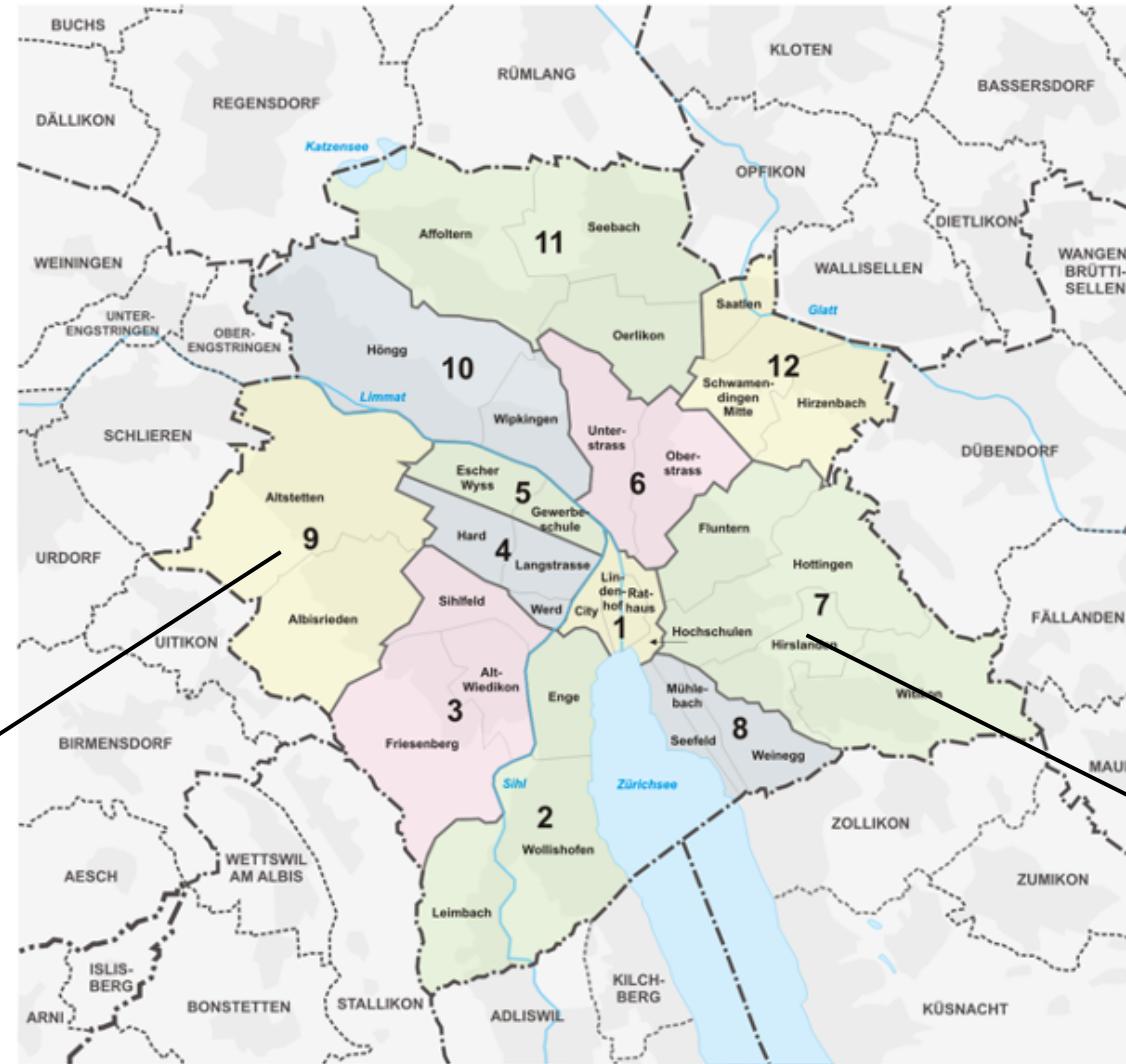
Der Ort ist wichtig...



Platz 3: 984

Platz 2: 1087

Der Ort ist wichtig...

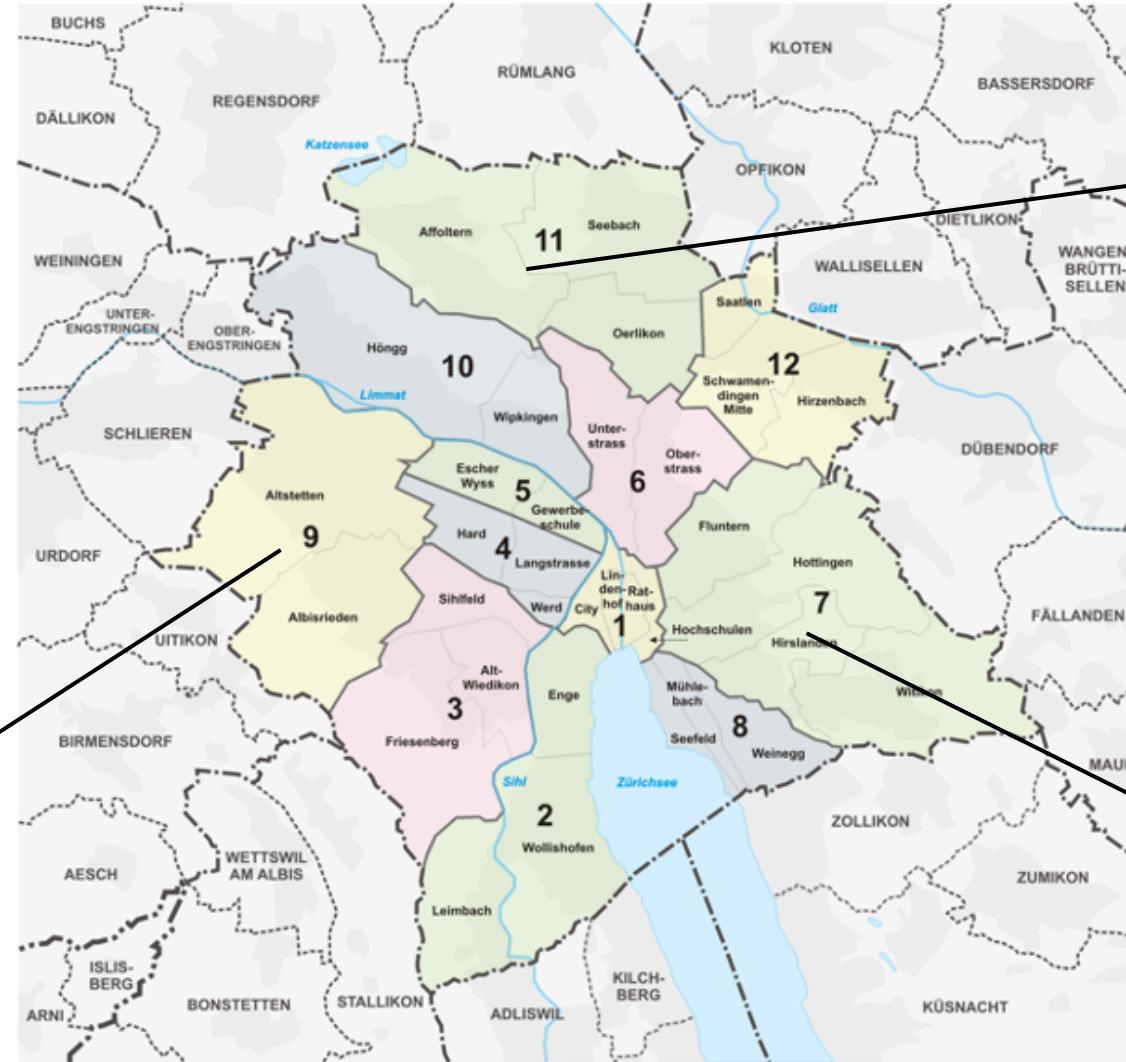


Platz 3: 984

Platz 1: 1352

Platz 2: 1087

Der Ort ist wichtig...



Platz 3: 984

Platz 1: 1352

Platz 2: 1087

Der Ort ist wichtig...

 Stadt Zürich
Open Data

Startseite Datensätze Kategorien



[/ Datensätze / Hundebestand der Stadt Zürich](#)

 [Lizenz](#)
 Creative Commons CCZero
[OPEN DATS](#)

 [Datensatz](#)  [Kategorien](#)  [Showcases](#)

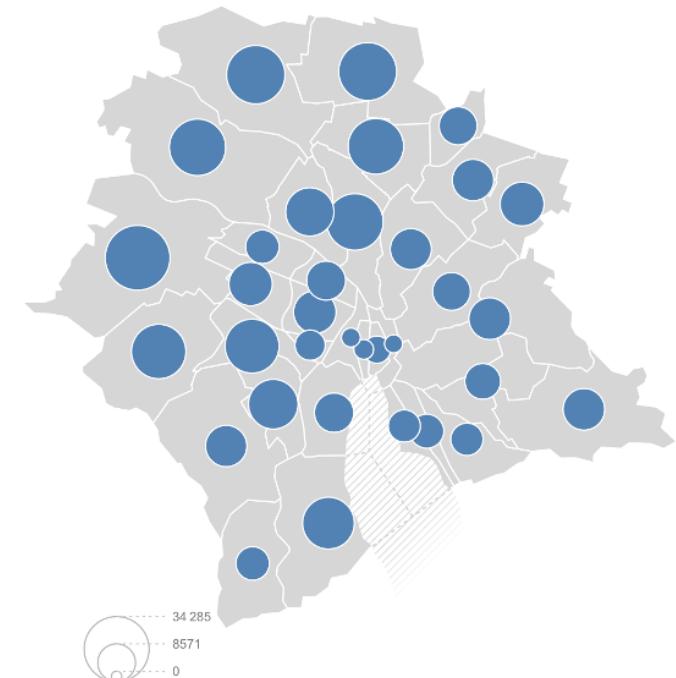
Hundebestand der Stadt Zürich

In diesem Datensatz finden Sie Angaben zu Hunden und deren Besitzerinnen und Besitzern aus dem aktuellen Bestand des städtischen Hunderegisters. Bei den hundehaltenden Personen sind Informationen zur Altersgruppe, dem Geschlecht und dem statistischen Quartier des Wohnorts angegeben. Zu jedem Hund ist die Rasse, der Rassetyp, das Geschlecht, das Geburtsjahr und die Farbe erfasst. Das Hunderegister wird von der Abteilung Hundekontrolle der Stadtpolizei Zürich geführt.

 [20200306_hundehalter.csv](#)
Comma-Separated Values. Weitere Informationen zu CSV finden Sie in unserer...

 [Entdecke](#) ▾

Bevölkerung nach Stadtquartier



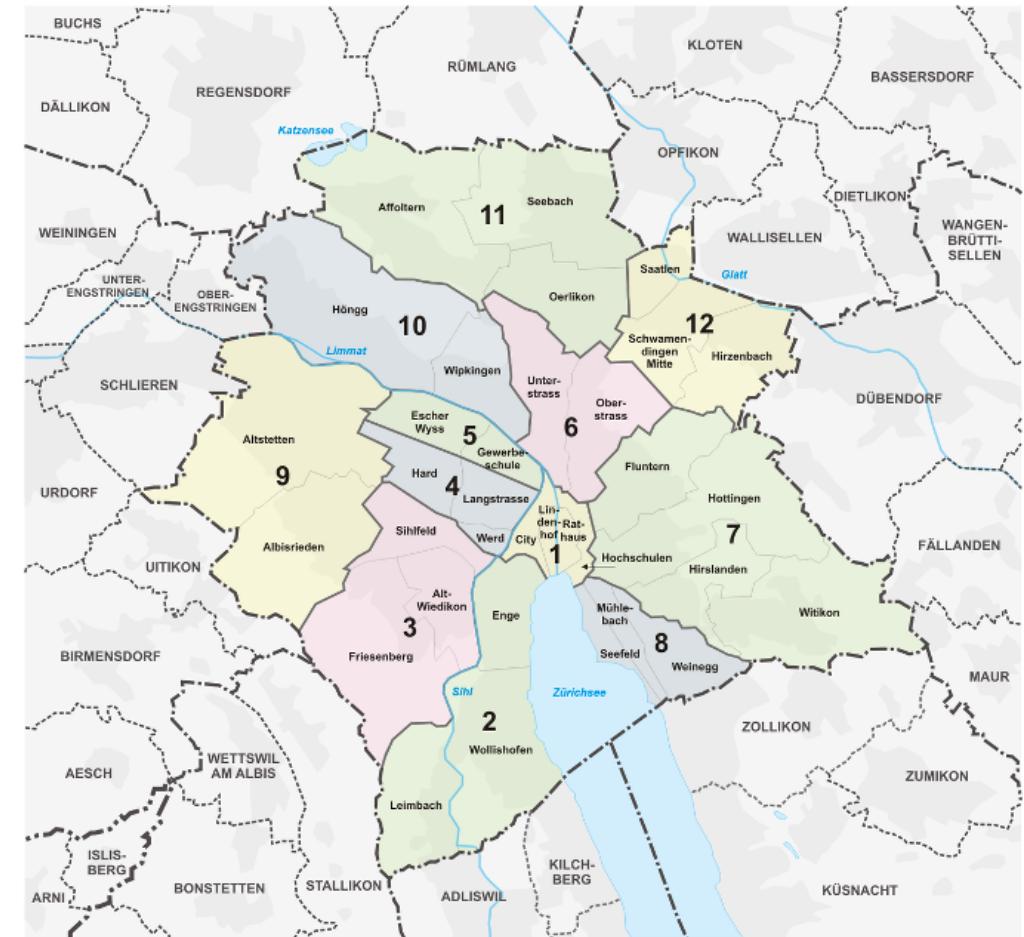
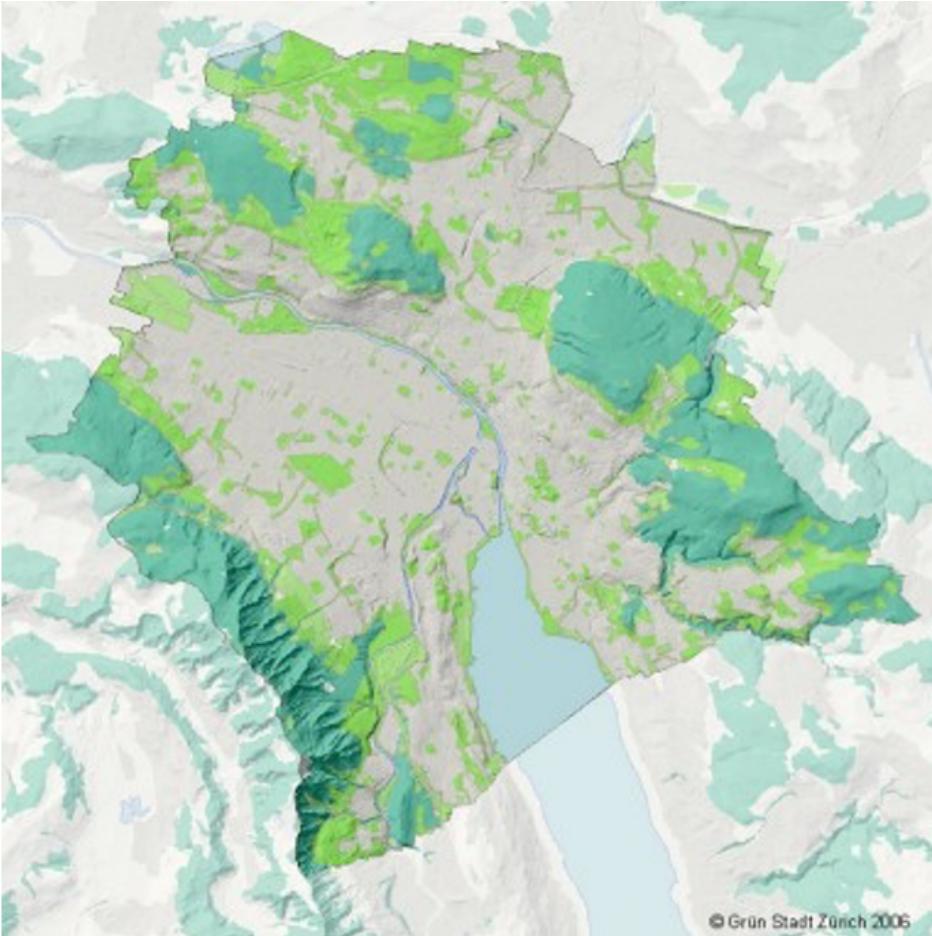
Hier besonders auf zeitliche Unterschiede achten:

Hundedaten: März 2020 / Bevölkerungsdaten: 1993-2020

Ist die Anzahl Personen wichtig?

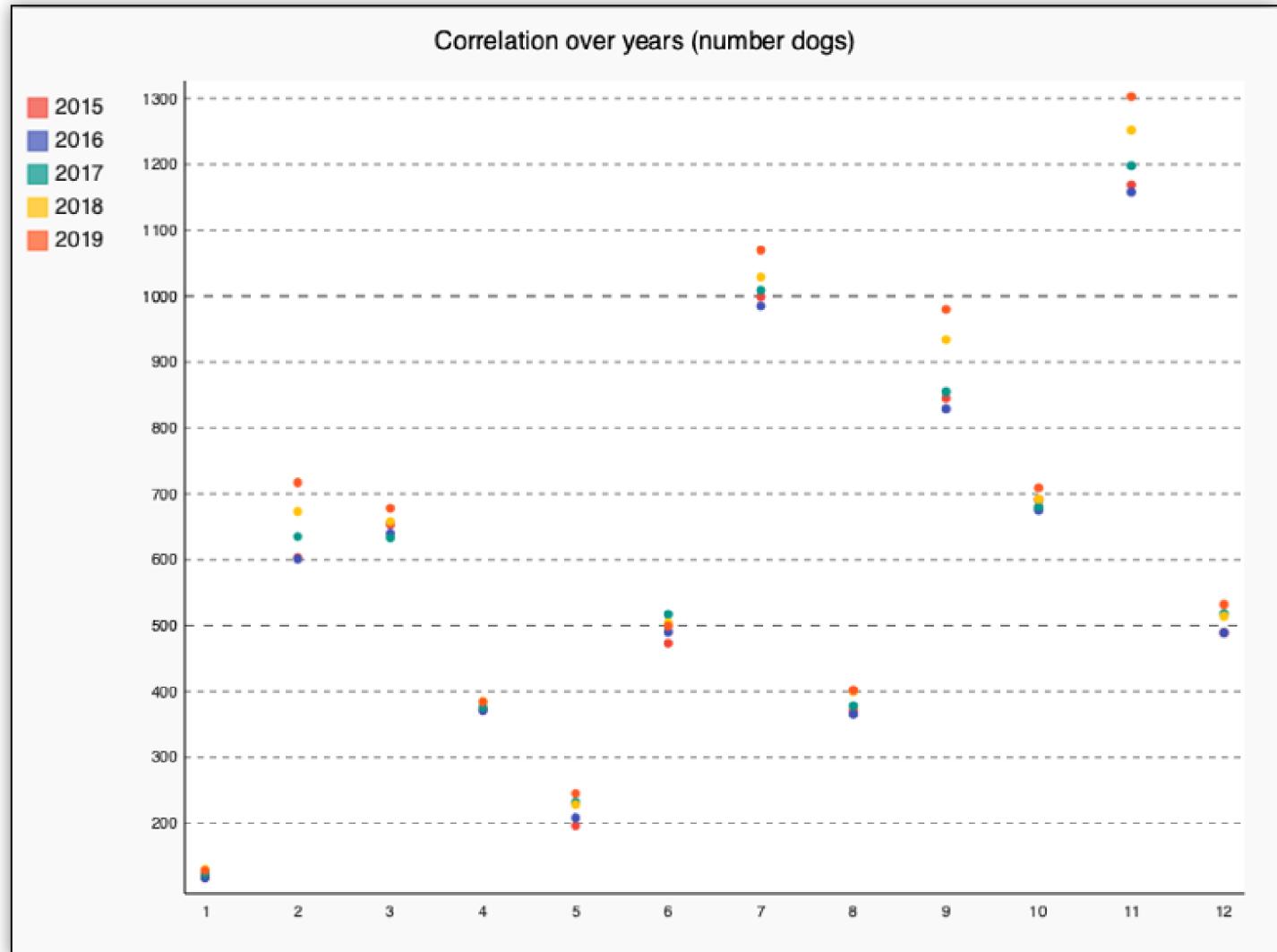
- Kreis 1; 5 831
- Kreis 2; 35 552
- Kreis 3; 50 756 => viele Menschen, weniger Hunde (697)
- Kreis 4; 29 034
- Kreis 5; 15 622
- Kreis 6; 35 317
- Kreis 7; 38 629
- Kreis 8; 17 456
- Kreis 9; 56 462
- Kreis 10; 41 044
- Kreis 11; 76 188
- Kreis 12; 32 845

Sind Grünflächen wichtig?



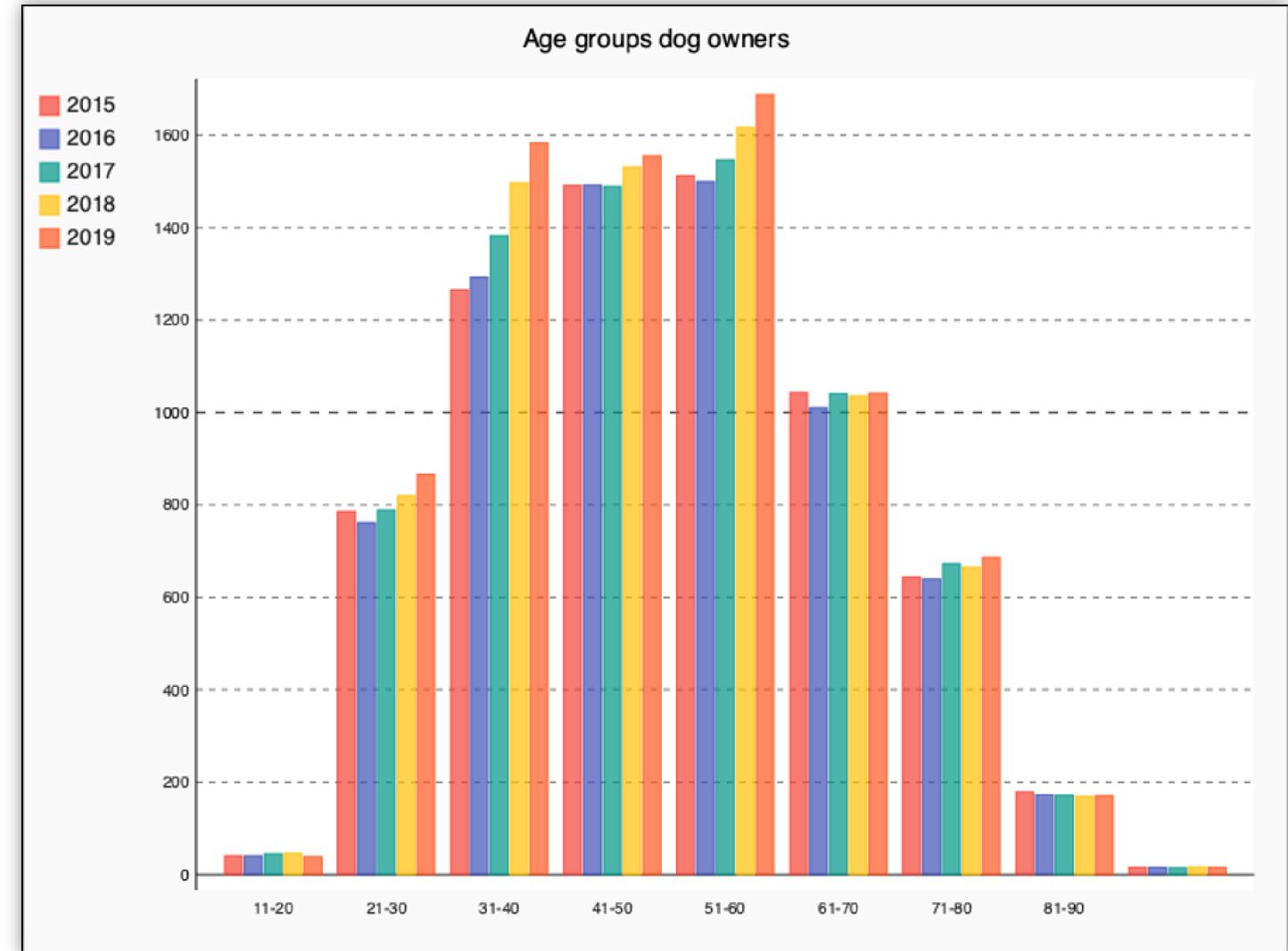
Korrelation berechnen

Siehe Step 2 in den Unterlagen



Altersgruppen

Am stärksten Ausgeprägt
ist die Gruppe von 31-60



Wie weiter?

- Selbstverständlich könnte man hier nun noch weite mehr Verfahren anwenden:
 - K-Nearest Neighbour
 - Naive Bayes
 - Lineare Regression
 - Multiple Regression
 - Logistische Regression
 - Decision-Trees
 - Neuronal-Networks / Deep-Learning
 - Clustering
 - usw.

Abschluss: Was empfehlen wir Dog-Science?

- Wir empfehlen Dog-Science einen Platz **im Kreis 11** in der **Nähe einer Grünfläche zu beantragen.**
- Wir empfehlen das Produkt für 31-61-Jährige Personen zu gestalten.
- Es darf etwas kosten (siehe Durchschnittsgehalt Zürich).
- Es darf nicht zu verspielt sein und sollte eher einfach verständlich sein (Interpretation der Altersgruppe).

Capstone Project

Wir bauen zusammen ein Dashboard mit Dash.

Capstone Project

Schaut euch als erstes folgendes an:

<https://tracking-dashboard-app.herokuapp.com/dashboard>

Capstone Project

Unser Beispiel wird simpler, verbindet aber dash (plotly) mit unseren Daten-Skills und Ideen zur Visualisierung.

Capstone Project

Dash Hello World

Capstone Project

Dash Hello World



```
1 # 13_hello_world_dash.py
2
3 import openpyxl
4 from dash import Dash, html
5
6 app = Dash(__name__)
7
8 app.layout = html.Div([
9     html.Div(children='Hello World')
10])
11
12 if __name__ == '__main__':
13     app.run(debug=True)
```

Capstone Project

Mit Daten verbinden

Capstone Project

Mit Daten verbinden



```
1 # 14_dash_with_data.py
2
3 # Import packages
4 from dash import Dash, html, dash_table
5 import pandas as pd
6
7 # Incorporate data
8 df = pd.read_csv('gapminder2007.csv')
9
10 # Initialize the app
11 app = Dash(__name__)
12
13 # App layout
14 app.layout = html.Div([
15     html.Div(children='My First App with Data'),
16     dash_table.DataTable(data=df.to_dict('records'), page_size=10)
17 ])
18
19 # Run the app
20 if __name__ == '__main__':
21     app.run(debug=True)
```

Capstone Project

Und einer Visualisierung

Capstone Project

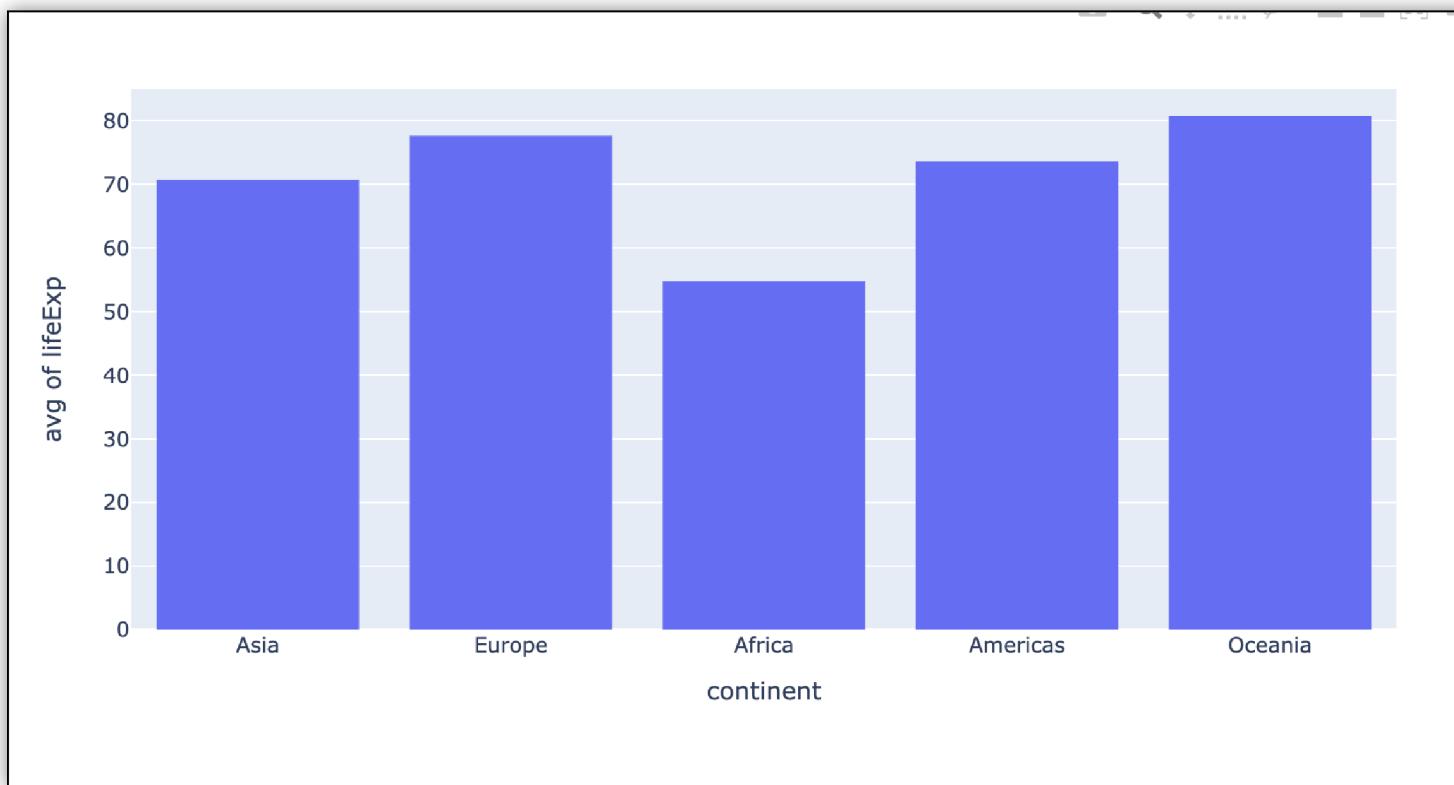
Und einer Visualisierung



```
1 # 15_dash_with_data_and_vis.py
2
3 # Import packages
4 from dash import Dash, html, dash_table, dcc
5 import pandas as pd
6 import plotly.express as px
7
8 # Incorporate data
9 df = pd.read_csv('gapminder2007.csv')
10
11 # Initialize the app
12 app = Dash(__name__)
13
14 # App layout
15 app.layout = html.Div([
16     html.Div(children='My First App with Data and a Graph'),
17     dash_table.DataTable(data=df.to_dict('records'), page_size=10),
18     dcc.Graph(figure=px.histogram(df, x='continent', y='lifeExp', histfunc='avg'))
19 ])
20
21 # Run the app
22 if __name__ == '__main__':
23     app.run(host='0.0.0.0', port=5005)
```

Capstone Project

Und einer Visualisierung



Capstone Project

Und Controls (Steuerung)

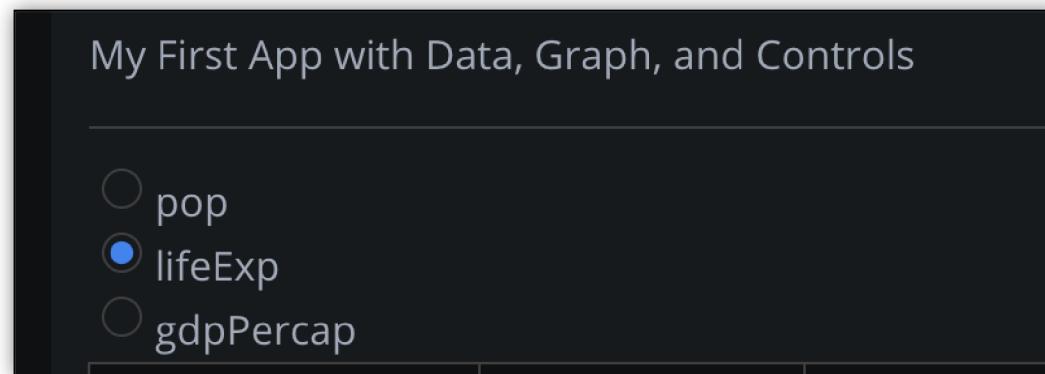
Capstone Project

Und Controls (Steuerung)

```
● ● ●  
1 # 16_dash_with_data_and_vis_cont.py  
2  
3 # Add controls to build the interaction  
4 @callback(  
5     Output(component_id='controls-and-graph', component_property='figure'),  
6     Input(component_id='controls-and-radio-item', component_property='value')  
7 )  
8 def update_graph(col_chosen):  
9     fig = px.histogram(df, x='continent', y=col_chosen, histfunc='avg')  
10    return fig
```

Capstone Project

Sowie die effektiven Radio-Buttons

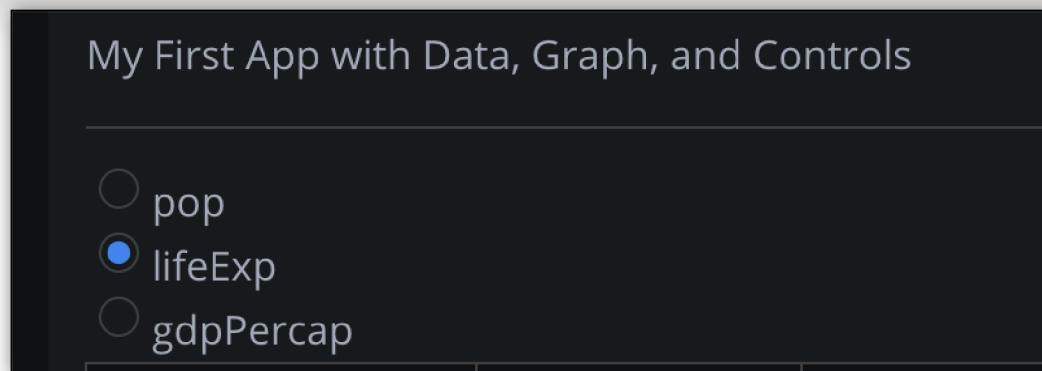


Capstone Project

Sowie die effektiven Radio-Buttons



```
1 # 16_dash_with_data_and_vis_cont.py
2
3 # App layout
4 app.layout = html.Div([
5     html.Div(children='My First App with Data, Graph, and Controls'),
6     html.Hr(),
7     dcc.RadioItems(options=['pop', 'lifeExp', 'gdpPercap'], value='lifeExp', id='controls-and-radio-item'),
8     dash_table.DataTable(data=df.to_dict('records'), page_size=6),
9     dcc.Graph(figure={}, id='controls-and-graph')
10 ])
```



Capstone Project

Die finale - extern gestylte Variante schauen wir uns im Code an.

06-capstone.

Vielen Dank

Das war alles für den Moment

Ich liebe es, über Vue, Python und mehr zu diskutieren

Sie finden mich unter @dpinezich

