Curious About New Places?

Explore Them Via OpenStreetMaps API

Vojta Filipec

PyConCZ 2019

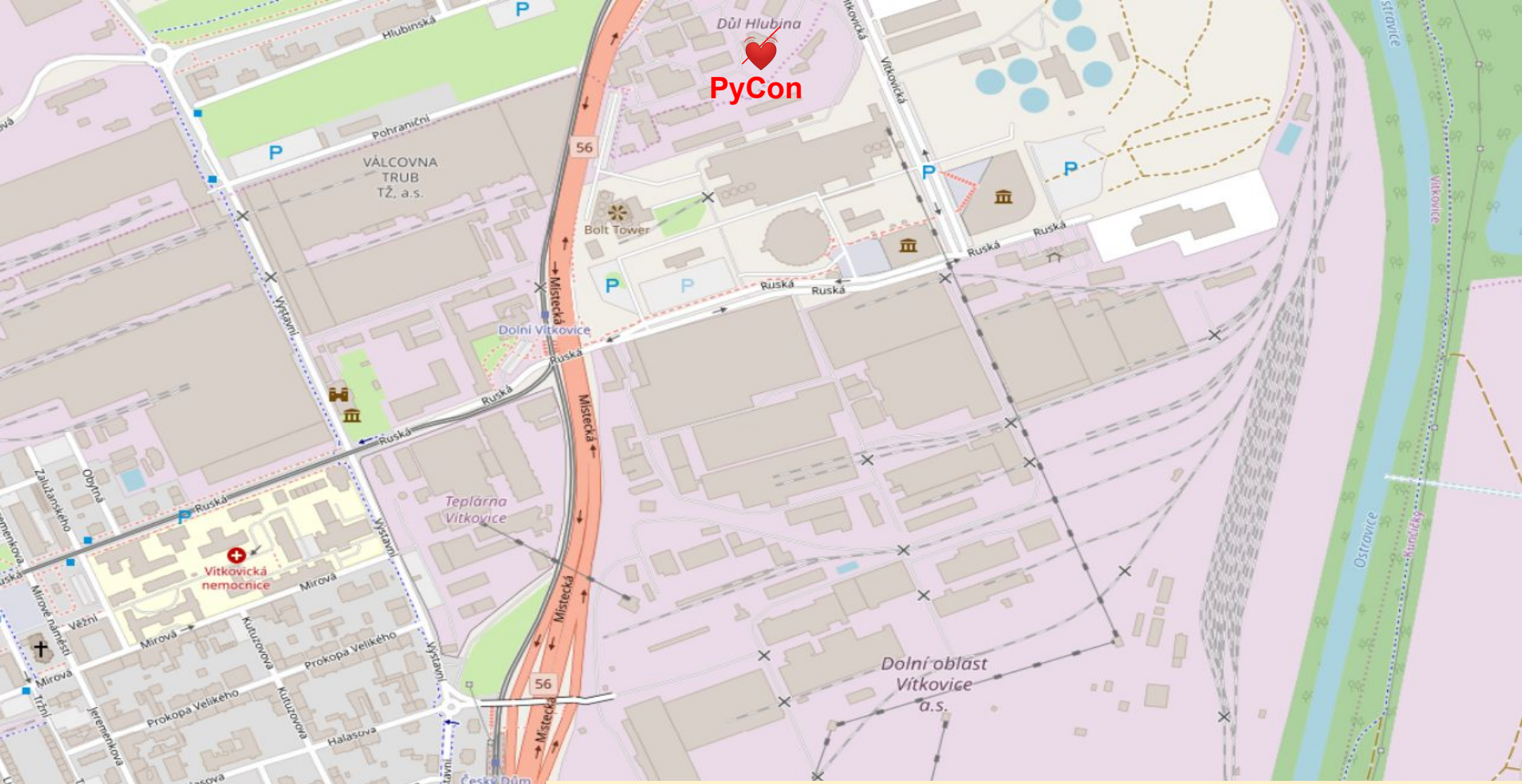
Whoami

☢ nuclear physicist

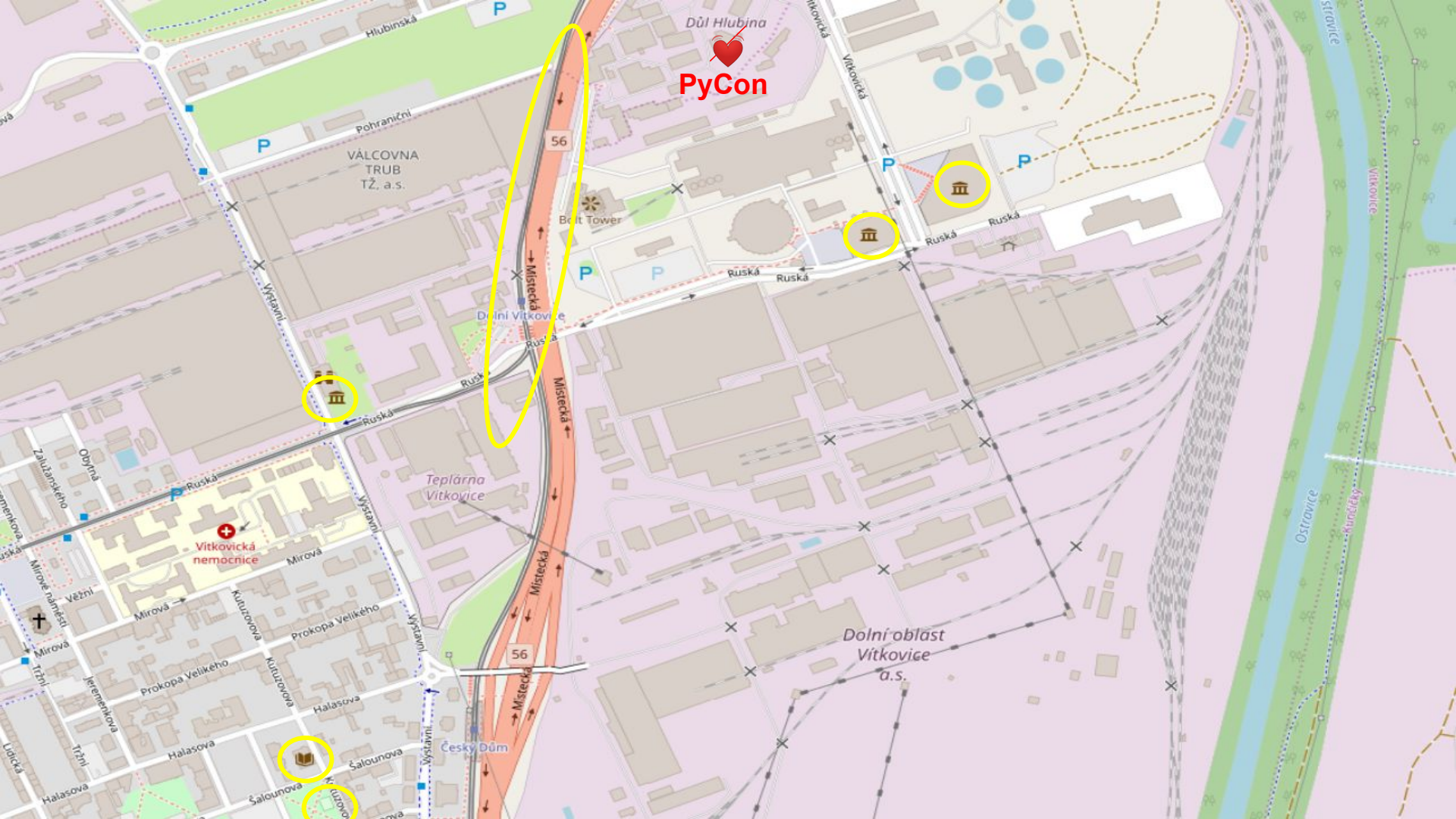
data scientist at Twisto.cz

✉ linkedin.com/in/vojtech-filipec/

PR github.com/vojtech-filipec/

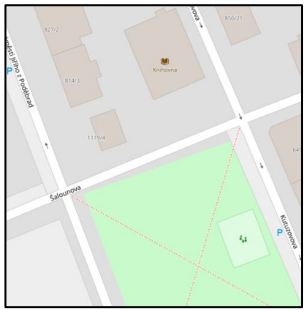


Look into the Map To Understand a Neighbourhood

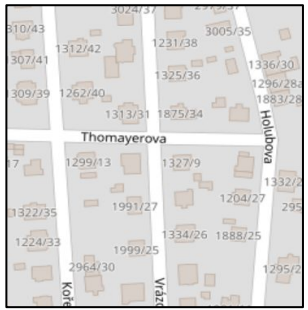

PyCon
Důl Hlubina
Hlubinská
Pohraniční
VÁLCOVNA
TRUB
TŽ, a.s.
Bolt Tower
Vítkovická
Ruská
Místecká
Dolní Vítkovice
Výstavní
Teplárna
Vítkovice
Vítkovická
nemocnice
Mírová
Zakuzanského
Obyrná
Kutuzovova
Prokopa Velikého
Jeremenkova
Tržní
Halasova
Šalounova
Lidická
Český Dům
Dolní oblast
Vítkovice
a.s.
Ostravice
Stravice
Vítkovice
56

“Can you do it for all locations in our DB?”

place #1

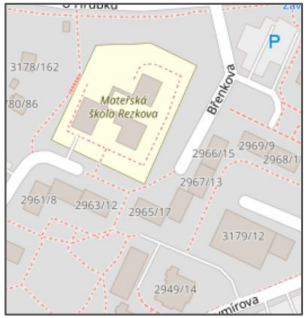


place #2



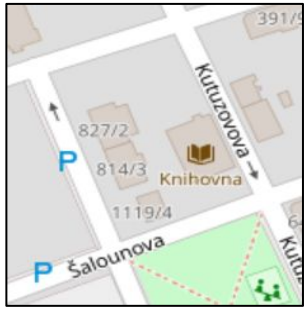
place #3

place #4



. . .

place #”big-data”



API: OpenStreetMaps

How To Explore Many Places via OSM

1. Decide what you're after …
2. search manually …
3. replicate via a query …
4. run in python, parametrize and serialize …
5. download all relevant points of interest.

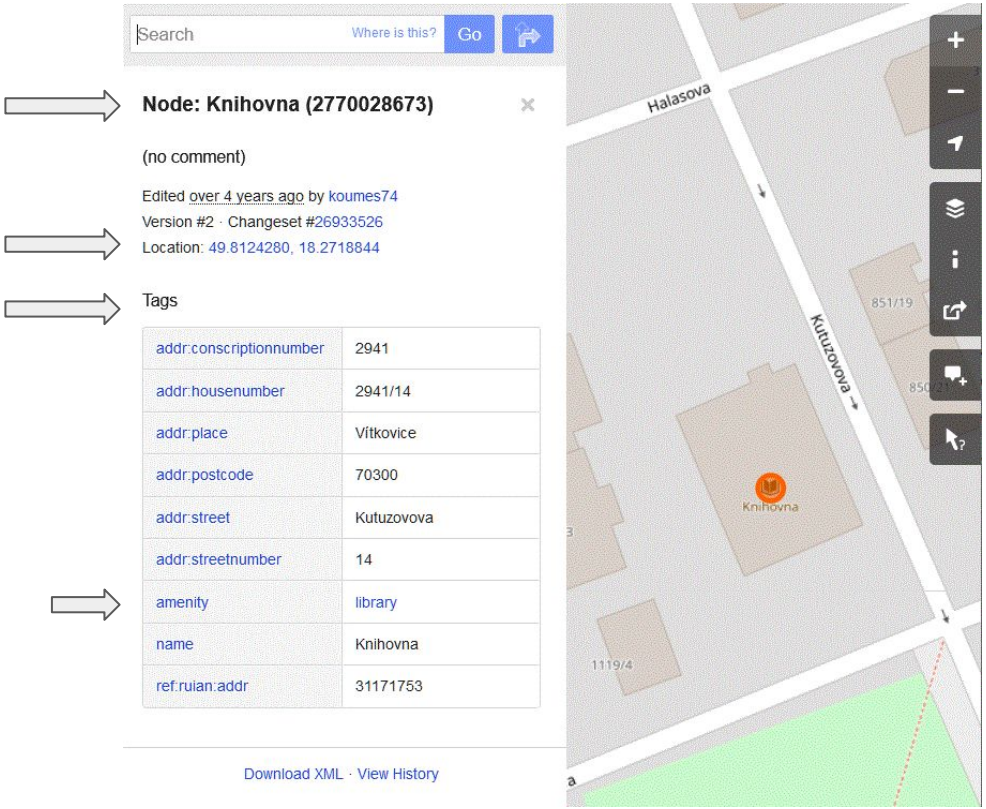
1. Decide What You Are After

“Are there any libraries around here?”

2. Search Manually

www.openstreetmap.org

guess: search “knihovna” (library)



more in Appendix

3. Replicate via a Query (part 1/2)

How are objects mapped in OSM:

- Every object consists of points
  - **node**: e.g. a statue
  - **way**: road, or park
  - **relation**: “collections” (e.g. country of Indonesia)

- Every object can be tagged:
  - `key:value` pairs (e.g. `surface:asphalt`)
  - extensive list of features

more in Appendix

3. Replicate via a Query (part 1/2)

How are objects mapped in OSM:

- Every object consists of points
  - **node**: e.g. a statue
  - **way**: road, or park
  - **relation**: “collections” (e.g. country of Indonesia)

- Every object can be tagged:
  - `key:value` pairs (e.g. `surface:asphalt`)
  - extensive list of features

more in Appendix

3. Replicate via a Query (part 1/2)

How are objects mapped in OSM:

- Every object consists of points
  - **node**: e.g. a statue
  - **way**: road, or park
  - **relation**: “collections” (e.g. country of Indonesia)

- Every object can be tagged:
  - `key:value` pairs (e.g. `surface:asphalt`)
  - extensive [list of features]

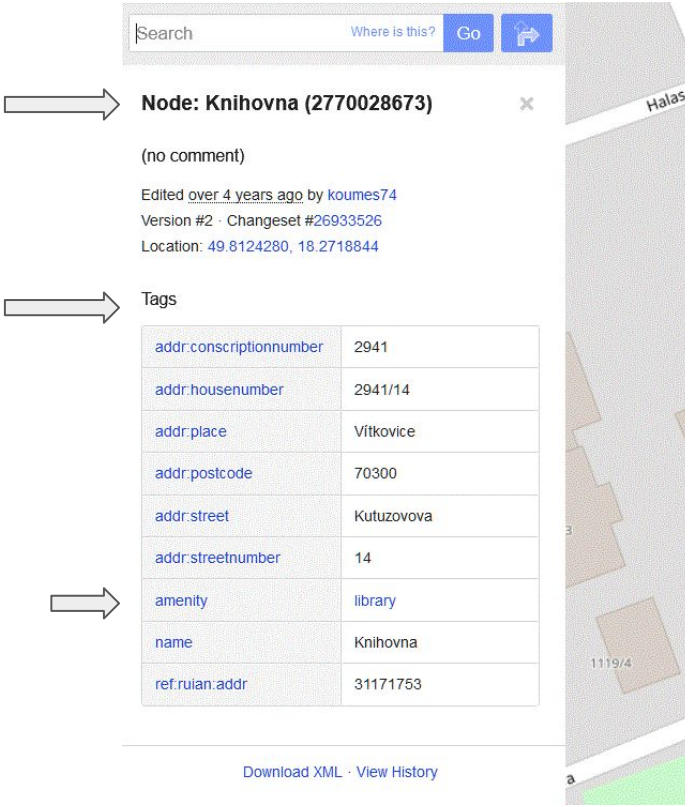
more in Appendix

3. Replicate via a Query (part 2/2)

query language of OSM:

```
type of object (node, way, rel?)
  [tags as filter]
  (where to search);
out;
```

3. Replicate via a Query (parts 1 + 2)




```
node
  [amenity=library]
  (around: 50, 49.812428, 18.2718844);
out;
```

[overpass-turbo]

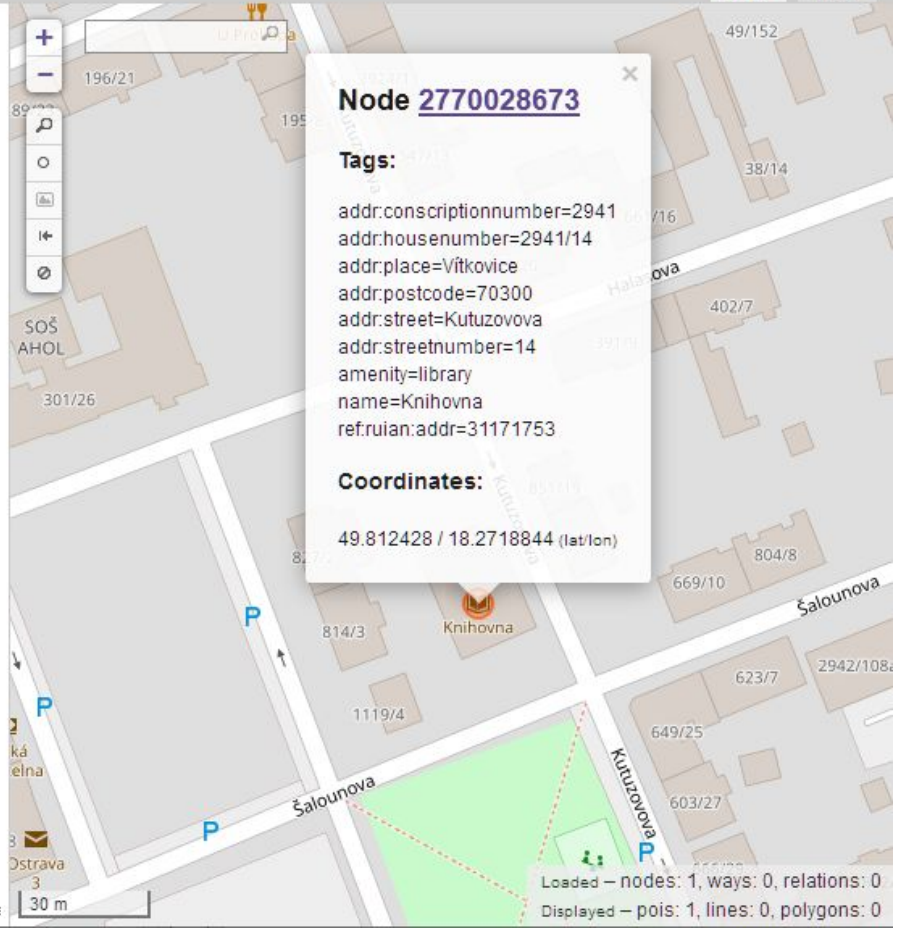
overpass-turbo.eu

Run | Share | Export | Wizard | Save | Load | Logout | Settings | Help

overpass turbo

Map | Data

```
node
  [amenity=library]
  (around: 50, 49.812428, 18.2718844);
out;
```



4. Run the Query in Python

```
import overpy
api = overpy.Overpass()

resp = api.query("""
node
  [amenity=library]
  (around: 50, 49.812428, 18.2718844);
out;""")

print("# nodes:", len(resp.nodes))
```

4. Run the Query in Python

```
import overpy
api = overpy.Overpass()

resp = api.query("""
node
  [amenity=library]
  (around: 50, 49.812428, 18.2718844);
out;""")

print("# nodes:", len(resp.nodes))
```

more in Appendix

4. Parametrize & Serialize

```
lat = 49.812428
lon = 18.2718844


resp = api.query("""
node
  [amenity=library]
  (around: 50, {}, {});
out;""".format(lat, lon)
)
```

demo in notebook

4. Parametrize & Serialize

```
lat = 49.812428
lon = 18.2718844


resp = api.query("""
node
  [amenity=library]
  (around: 50, {}, {});
out;""".format(lat, lon)
)
```

```
lats = [50.000, 49.000, 48.000]
lons = [18.272, 18.272, 18.272]

for lat, lon in zip(lats, lons):
      resp = api.query(

       … code from left …



      )
```

demo in notebook

5. Download All Relevant Points of Interest

- **relevant**: use tags - billions of elements in OSM (stats)
- **all**:

5. Download All Relevant Points of Interest

- **relevant**: use tags - billions of elements in OSM (stats)
- **all**: (node + way + rel)

```
<node id="2958302725" lat="49.8297783" lon="18.2861635">
    <tag k="leisure" v="playground"/>
    <tag k="source" v="survey"/>
</node>
<way id="181087418">
    <center lat="49.8278720" lon="18.2623439"/>
    <nd ref="1915344509"/>
    <nd ref="1915344489"/>
    <nd ref="1915344507"/>
    <nd ref="1915344517"/>
    <nd ref="1915344509"/>
    <tag k="leisure" v="playground"/>
</way>
```

Thank you … Questions?

Extra info follows

Contribute to OSM

- Easy to start contributing: web GUI

API's and Offline Access:

- on-line via Python: summary of options
  - library `overpass`: too simple, link
  - library `OverPy`: my choice, link

- off-line: tool `Osmosis`: can import OSM data to a DB
  - howto: https://wiki.openstreetmap.org/wiki/Osmosis

Finding The Right tag

a guess often works

proper way:

- https://taginfo.openstreetmap.org … statistics of key-value pairs
- https://wiki.openstreetmap.org/wiki/Map_Features … list of tags

OSM Query Language

- Dedicated query language:
  - reference
  - node / way / relation at a GPS position
  - node / way / relation around a GPS position
  - keyword filtering
  - recursion and union

- Offline: Local installation, updating mechanism available
- Online: Overpass API & few others (details)

Everything can be tagged:

Dalejský potok and *Vltava*

- Nodes along the stream
- Brook as a way: connected nodes
- (actually multiple ways)
-
-
- Properties mapped to each way

- Nodes along each river-bank
- Bank as a way: connected nodes
- River as a surface between banks with "holes":
  - Islands as surfaces
- Properties mapped as:
  - Relation of surfaces