

OSM-Daten verarbeiten mit Python und Pyosmium

Sarah Hoffmann
lonvia@denofr.de

12. März 2020

FOSSGIS 2020
Freiburg

Tools für OSM-Daten-Verarbeitung

Rendering	<ul style="list-style-type: none">• osm2pgsql• imposm
Routing	<ul style="list-style-type: none">• OSRM• Graphhopper
Geocoding	<ul style="list-style-type: none">• Nominatim• Pelias
Visualisierung	<ul style="list-style-type: none">• QGIS

Tools für OSM-Daten-Verarbeitung

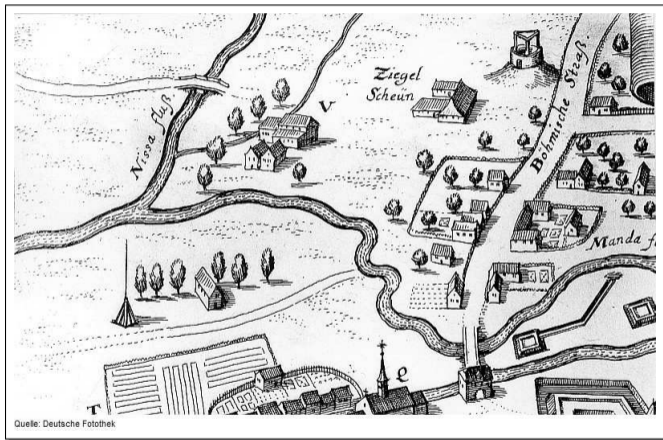
Rendering	<ul style="list-style-type: none">• osm2pgsql• imposm
Routing	<ul style="list-style-type: none">• OSRM• Graphhopper
Geocoding	<ul style="list-style-type: none">• Nominatim• Pelias
Visualisierung	<ul style="list-style-type: none">• QGIS
maßgefertigt	<ul style="list-style-type: none">• Pyosmium

Klassisches GIS

vs.

OSM Datenmodell

Klassisches GIS-Modell

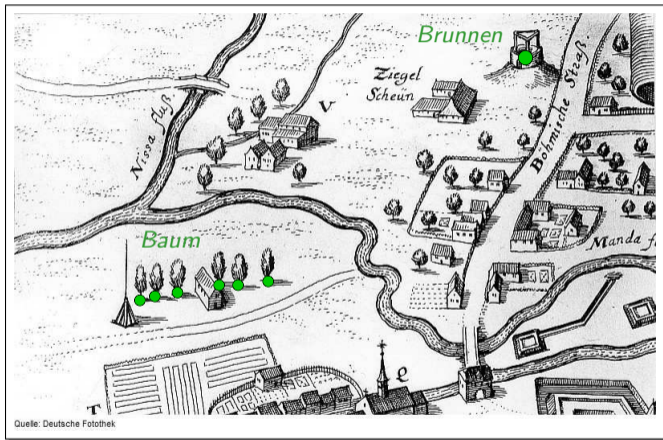


Punkt

Linie

Polygon

Klassisches GIS-Modell

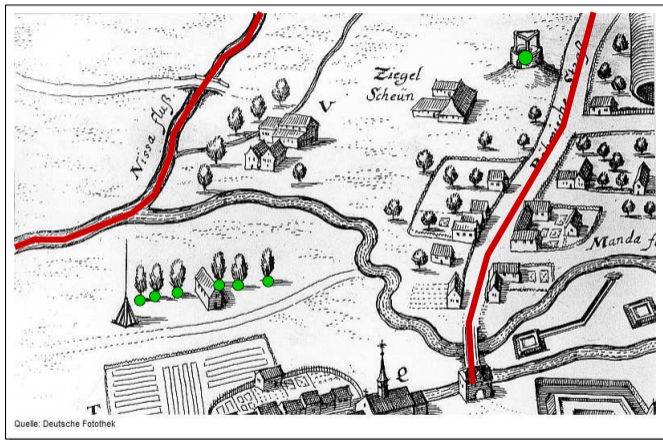


Punkt

Linie

Polygon

Klassisches GIS-Modell

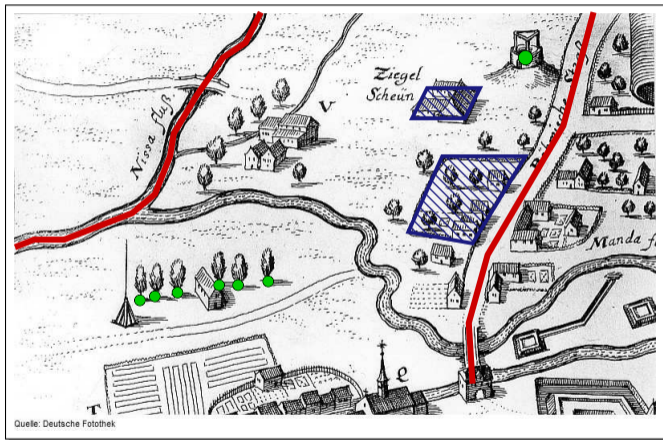


Punkt

Linie

Polygon

Klassisches GIS-Modell

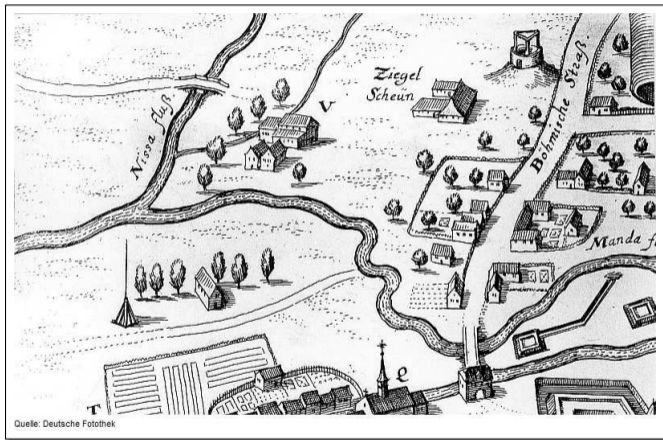


Punkt

Linie

Polygon

OSM Datenmodell

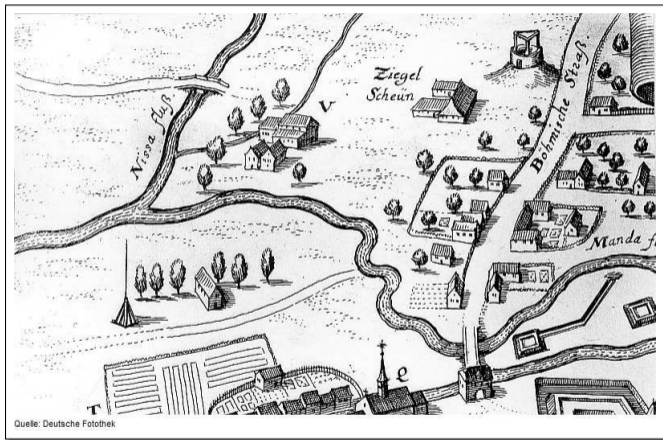


Node = Punkt

Way = Linie

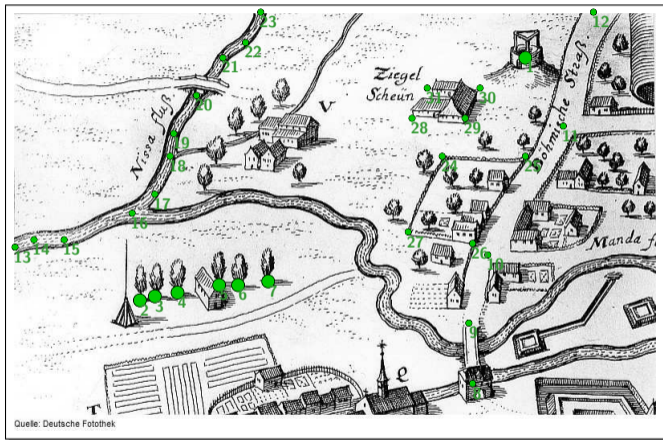
Relation = Polygon

OSM Datenmodell



~~Node = Punkt
Way = Linie
Relation = Polygon~~

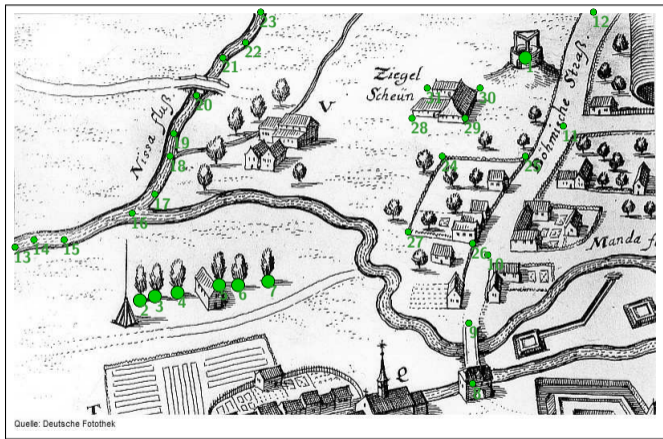
OSM Datenmodell



Node: Koordinate
+ Attribute (Tags)

Nur Nodes haben
eine Geometrie.

OSM Datenmodell



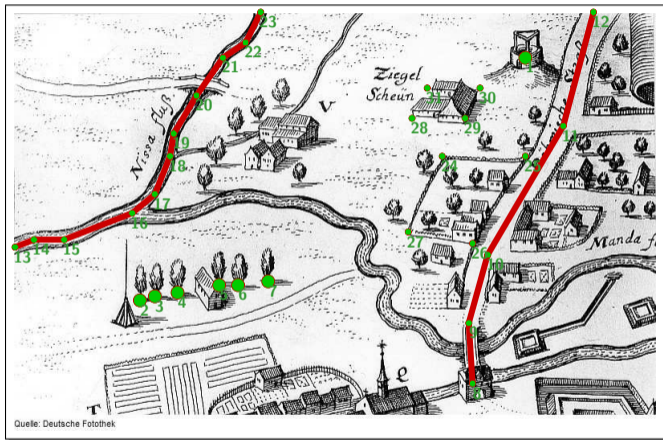
Node: Koordinate

Way: Liste von Nodes

Relation: Set von Objekten

+ Attribute (Tags)

OSM Datenmodell



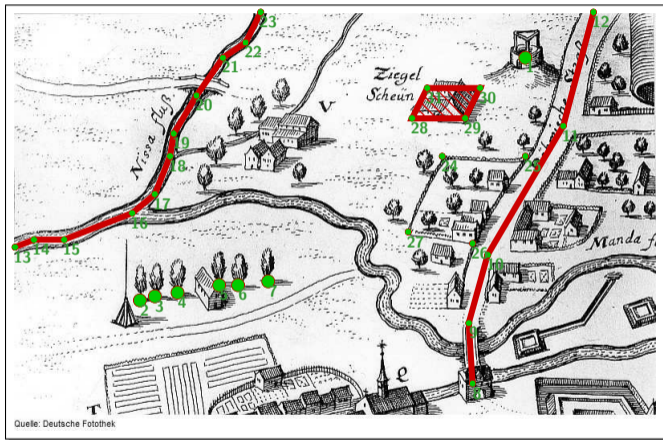
Node: Koordinate

Way: Liste von Nodes

Relation: Set von Objekten

+ Attribute (Tags)

OSM Datenmodell



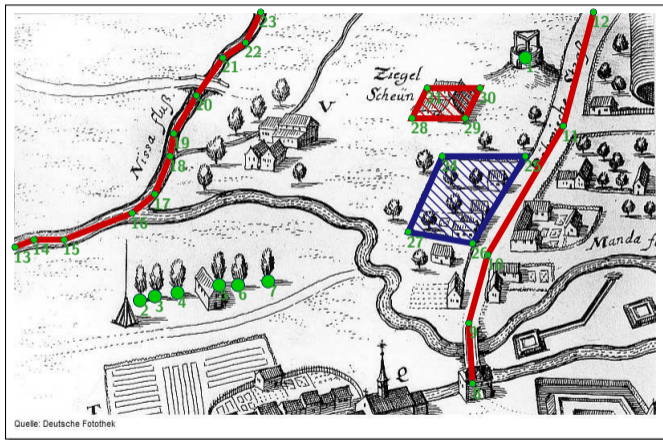
Node: Koordinate

Way: Liste von Nodes

Relation: Set von Objekten

+ Attribute (Tags)

OSM Datenmodell



Node: Koordinate

Way: Liste von Nodes

Relation: Set von Objekten

+ Attribute (Tags)

OSM Datenmodell: Verwendung von Relationen

Busrouten



Ways: Strassen entlang der Route
Nodes: Haltestellen

Abbiegebeschränkung

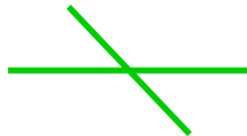
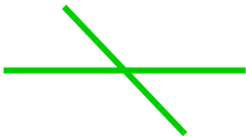


Ways: Beschränkung von/nach
Node: Kreuzung

OSM-Daten beschreiben
Topologie.

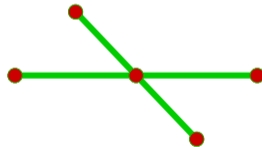
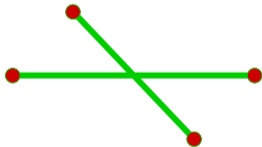
Geometrie ist Sache der Interpretation.

Beispiel: Brücke vs. Kreuzung



GIS-Welt

Beispiel: Brücke vs. Kreuzung



OSM-Welt

OSM-Daten mit Pyosmium verarbeiten

```
node    id=456374 x=7.7170000  
        [highway=motorway_junct  
node    id=452982 x=7.7697955  
        []  
node    id=452983 x=7.7676013  
        []  
way     id=4099116 nodes=[6688  
        [highway=unclassified,  
relation id=6866 members=[w2322
```



Einstiegsbeispiel: Inhalt der OSM-Datei auflisten

```
import osmium
```

```
def print_info(obj):  
    print(str(obj))
```

```
handler = osmium.make_simple_handler(  
    node=print_info,  
    way=print_info,  
    relation=print_info)
```

```
handler.apply_file("freiburg.osm.pbf")
```

Das ist ein pyosmium-Programm.

Callback-Funktion(en):
was tun mit einem Objekt

Handler:
welche OSM-Typen,
welche Funktion

Auf 'Freiburg' anwenden.

Einstiegsbeispiel: Inhalt der OSM-Datei auflisten

```
import osmium
```

```
def print_info(obj):  
    print(str(obj))
```

```
handler = osmium.make_simple_handler(  
    node=print_info,  
    way=print_info,  
    relation=print_info)
```

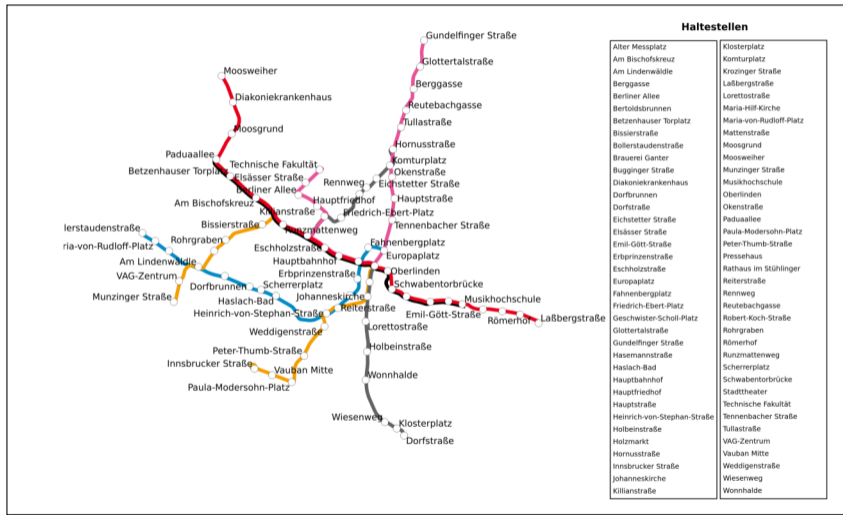
```
handler.apply_file("freiburg.osm.pb
```

```
me@home:~$ python3 basic.py  
n452955: location=7.810977/48.068316 tags={}  
n452957: location=7.812677/48.059660 tags={traff  
...  
n7235830609: location=7.802707/47.990415 tags={}  
w4018407: nodes=[293294331,456442,967115842,4564  
w4018427: nodes=[5021738137,456375,29337749,5021  
...  
w775622909: nodes=[7235830609,7235830606,7235830  
r882: members=[w481211268@inner,w376310956@outer  
r3344: members=[n446307533@TMC:RoadStart,n147786  
...  
r10742268: members=[w775565429@inner,w775565428  
me@home:~$
```

OSM-Dateien sind sortiert:
Node, Way, Relation.

Wenn sie es doch nicht sind: vorverarbeiten!

Beispiel: alphabetische Liste der Tram-Haltestellen

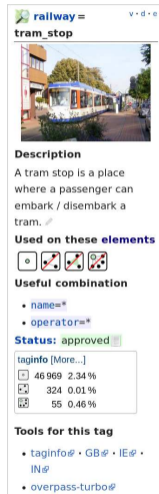


Was ist eine Straßenbahn-Haltestelle?


siehe OSM Wiki, Taginfo, etc.

- OSM Node
- Tagging: railway=tram_stop
- hat name Tag

```
def is_tram_stop(node):  
    return node.tags.get('railway') == 'tram_stop' \  
           and 'name' in node.tags
```



railway = tram_stop v · d · e



Description
A tram stop is a place where a passenger can embark / disembark a tram. ↗

Used on these elements



Useful combination

- name=*
- operator=*

Status: approved

taginfo [More...]

📍	46 969	2.34 %
🗺️	324	0.01 %
📄	55	0.46 %

Tools for this tag

- taginfo ⓘ · GB ⓘ · IE ⓘ · IN ⓘ
- overpass-turbo ⓘ

Alphabetische Haltestellenliste: 1.Versuch

```
stops = []
```

```
def filter_node(node):  
    if is_tram_stop(node):  
        stops.append(node)
```

```
h = osmium.make_simple_handler(node=filter_node)  
h.apply_file("freiburg.osm.pbf")
```

```
names = [stops.tags['name'] for stop in stops]
```

```
for name in sorted(names):  
    print(name)
```

Alle Tram-Haltestellen sammeln.

Auf 'Freiburg' anwenden.

Namen finden, sortieren, ausgeben.

Alphabetische Haltestellenliste: 1.Versuch

```
me@home:~$ python3 tram-stop-list.py
stops =
Traceback (most recent call last):
  File "stop-list-wrong.py", line 8, in <module>
    h.apply_file("freiburg.osm.pbf")
def filter
  if i
RuntimeError: Node callback keeps reference to OSM object. This is not allowed.
me@home:~$
```

```
h = osmium
h.apply_file("freiburg.osm.pbf")
```

```
names = [stop.tags['name'] for stop in stops]
```

```
for name in sorted(names):
    print(name)
```

anwenden.

Namen finden, sortieren,
ausgeben.

Alle relevanten Informationen
(und nur die!) kopieren.

Alphabetische Haltestellenliste: 2.Versuch

```
stops = set()
```

```
def filter_node(node):  
    if is_tram_stop(node):  
        stops.add(node.tags['name'])
```

```
h = osmium.make_simple_handler(node=filter_node)  
h.apply_file("freiburg.osm.pbf")
```

```
for name in sorted(stops):  
    print(name)
```

Set statt Liste vermeidet
Duplikate.

Namen der Haltestellen
sammeln.

Auf 'Freiburg'
anwenden.

Namen sortieren,
ausgeben.

Alphabetische Haltestellenliste: 2.Versuch

```
stops = set()
```

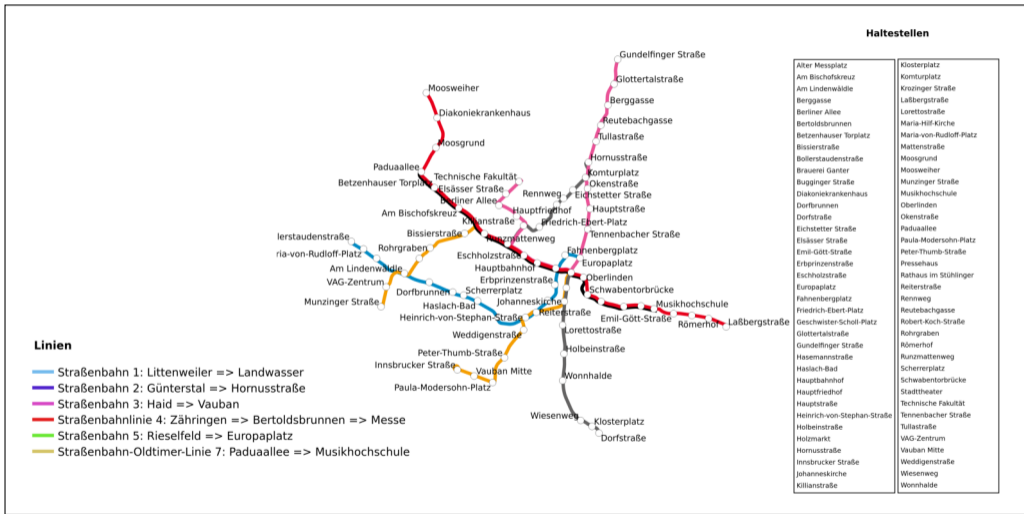
```
def filter_node(node):  
    if is_tram_stop(node):  
        stops.add(node.tags['name'])
```

```
h = osmium.make_simple_handler(node)  
h.apply_file("freiburg.osm.pbf")
```

```
for name in sorted(stops):  
    print(name)
```

```
me@home:~$ python3 tram-stop-list.py  
( Schwabentorschleife )  
Alter Messplatz  
Am Bischofskreuz  
Am Lindenwäldle  
Berggasse  
Berliner Allee  
Bertoldsbrunnen  
Betzenhauser Torplatz  
...  
Tullastraße  
VAG-Zentrum  
Vauban-Mitte  
Weddigenstraße  
Wiesenweg  
Wonnhalde  
me@home:~$
```

Beispiel: sortierte Liste der Tramlinien mit Länge



Was ist eine Tram-Route?

https://wiki.openstreetmap.org/wiki/Tag:route%3Dtram

English Create account Log in

Page Discussion Read View source View history Search OpenStreetMap Wiki

Tag:route=tram

cestina · Deutsch · English · español · français · português · русский · 日本語 · [Other languages](#) · [Translate](#)

This tag can be used to mark the route of a tram (a.k.a "streetcar") service (eg the "Red line").

Tagging a tram route relation

Use one relation for each direction (or variation), so you need at least 2 relations for a tram line, both connected with a route master relation.

Tags to add to a tram route

Key	Value	Comment
type	route	(mandatory)
route	tram	(mandatory)
ref	reference	The reference by which the route is known. e.g. 4, 4A, X13, etc. (recommended)
operator	operator	Name of the company that operates the route.
	individual name	
name	for PTv2@ please use: <vehicle type> <reference number>: <initial stop> => <terminal stop>	The name of the route or line.
network	local / regional network	Name (abbr.) of the network; e.g., BVG, RMV. (optional)
duration	time from start to end	The amount of time from the first to last stop of the tram route. Use HH:MM:SS, H:MM:SS, HH:MM, H:MM, MM, or M format.
interval	time between arrivals	The time between arrivals at any given station along the route, also known as the "service frequency". Use HH:MM:SS, H:MM:SS, HH:MM, H:MM, MM, or M format.
colour	ex: red / #FFEEDD	The "official" color for the tram route. Identifiers in some cities. (optional)
to	name	Destination station
from	name	Start station

route = tram

Description
The route of a tram or streetcar public transport service

Group: Route

Used on these elements

Requires

- type=route

Useful combination

- name=*
- operator=*
- network=*
- ref=*
- from=*

Was ist eine Tram-Route?

← → ↻ 🏠 <https://wiki.openstreetmap.org/wiki/Tag:route%3Dtram> ... 📄 🌟 🗺️ 📄 📄 📄 📄

🌍 Wiki

Page [Discussion](#) [Read](#) [View source](#) [View history](#)

Tag:route=tram

Tag:route=tram - Other languages [Purge](#) [Help](#)
čestina · **Deutsch** · English · español · français · português · русский · 日本語 · [Other languages](#) · [Translate](#)

This tag can be used to mark the route of a tram (a.k.a "streetcar") service (eg the "Red line").


Tagging a tram route relation

Use one relation for each direction (or variation), so you need at least 2 relations for a tram line, both connected with a route master relation.

```
def route_title(relation, length):  
    ref = relation.tags.get('ref', '?')  
    start = relation.tags.get('from', '??')  
    end = relation.tags.get('to', '??')  
  
    return "Linie {}: {} => {} ( {:.2f} km) \"\  
        .format(ref, start, end, length/1000.0)
```





colour	ex: red / #FFEEDD	The "official" color for the tram route. Identifiers in some cities. <i>(optional)</i>
to	name	Destination station
from	name	Start station

route = tram



Description
The route of a tram or streetcar public transport service

Group: Route

Used on these elements
   

Requires
• type=route

Useful combination
• name=+
• operator=+
• network=+
• ref=+
• from=+




Was ist eine Tram-Route?

← → ↻ 🏠 <https://wiki.openstreetmap.org/wiki/Tag:route%3Dtram> ... 📄 ☆

duration	<i>time from start to end</i>	The amount of time from the first to last stop of the tram route. Use HH:MM:SS, H:MM:SS, HH:MM, H:MM, MM, or M format.
interval	<i>time between arrivals</i>	The time between arrivals at any given station along the route, also known as the "service frequency". Use HH:MM:SS, H:MM:SS, HH:MM, H:MM, MM, or M format.
colour	<i>ex: red / #FFEEED</i>	The "official" color for the tram route. Identifiers in some cities. <i>(optional)</i>
to	<i>name</i>	Destination station
from	<i>name</i>	Start station

Members

Add all tracks [railway=tram](#) and stops to the relation as members.

Way/node	Role	Recurrence?	Discussion
	(blank)	one or more	The tracks making up the route. The order of the member in the relation should be identical to the order where the tram runs through.
	stop	Zero or more	A tram stop on the route. The order of the members in the relation should be identical to the order in the timetable.
	platform	Zero or more	A tram platform belonging to the route. The order of the members in the relation should be identical to the order of the stops in the timetable.

In [PTv2](#), the roles `route`, `forward` and `backward` are invalid.

See also

- [route=train](#) - The route of a train service (e.g. London-Paris Eurostar). Not to be confused with a named stretch of railroad track (see `route=railway`).

Intermezzo: Weglänge berechnen (1. Versuch)

```
def print_way_length(way):  
    length = osmium.geom.haversine_distance(way.nodes)  
    print("{}: {:.2f}m".format(way.id, length))  
  
h = osmium.make_simple_handler(way=print_way_length)  
h.apply_file("freiburg.osm.pbf")
```

Intermezzo: Weglänge berechnen (1. Versuch)

```
def print_way_length(way):
    length = osmium.geom.haversine_distance(way.nodes)
    print("{}: {:.2f}m".format(way.id, length))

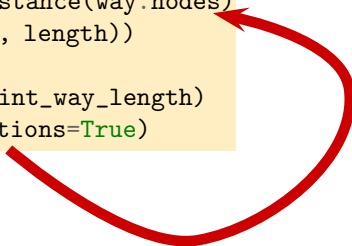
h = osmium.make_simple_handler(way=print_way_length)
h.apply_file("freiburg.osm.pbf")
```

```
me@home:~$ python route-length.py
Traceback (most recent call last):
  File "route-length-one-pass.py", line 21, in <module>
    h.apply_file("freiburg.osm.pbf")
  File "route-length-one-pass.py", line 7, in save_way_length
    ways[way.id] = osmium.geom.haversine_distance(way.nodes)
osmium._osmium.InvalidLocationError: invalid location
me@home:~$
```

Zum Berechnen von Geometrien muss
man Node-Koordinaten speichern.
Das kann teuer werden!

Intermezzo: Weglänge berechnen (2. Versuch)

```
def print_way_length(way):  
    length = osmium.geom.haversine_distance(way.nodes)  
    print("{}: {:.2f}m".format(way.id, length))  
  
h = osmium.make_simple_handler(way=print_way_length)  
h.apply_file("freiburg.osm.pbf", locations=True)
```



Intermezzo: Weglänge berechnen (2. Versuch)

```
def print_way_length(way):  
    length = osmium.geom.haversine_distance(way.nodes)  
    print("{}: {:.2f}m".format(way.id, length))  
  
h = osmium.make_simple_handler(way=print_way_length)  
h.apply_file("freiburg.osm.pbf", locations=True)
```

```
me@home: ~$ python3 way_length.py  
4018407: 2964.10m  
4018427: 195.48m  
4018441: 452.65m  
...  
775622908: 38.31m  
775622909: 18.39m  
me@home: ~$
```


Liste der Tramlinien mit Länge

```
ways = {}  
  
def save_way_length(way):  
    ways[way.id] = osmium.geom.haversine_distance(way.nodes)  
  
routes = []  
  
def process_route(relation):  
    if relation.tags.get('route') == 'tram':  
        total = 0  
        for m in relation.members:  
            if m.type == 'w' and m.role == '':  
                total += ways.get(m.ref, 0)  
  
    routes.append(route_title(relation, total))
```

Weglängen
speichern.

Längen nach
Members
aufsummieren.

Liste der Tramlinien mit Länge

```
ways = {}

def save_way_length(way): # 1. Schritt: Länge aller Wege merken
    ...

routes = []

def process_route(relation): # 2. Schritt: Routeninfo sammeln
    ...

h = osmium.make_simple_handler(way=save_way_length,
                               relation=process_route)
h.apply_file("freiburg.osm.pbf", locations=True)

for route in sorted(routes):
    print(route)
```

Liste der Tramlinien mit Länge

```
ways = {}
```

```
def save_way_length
```

```
...
```

```
routes = []
```

```
def process_route(r
```

```
...
```

```
h = osmium.make_sim
```

```
h.apply_file("freik
```

```
for route in sorted(routes):
```

```
    print(route)
```

```
me@home: python3 route-list-length.py
```

```
Linie 1: Laßbergstraße => Moosweiher (9.52 km)
```

```
Linie 1: Moosweiher => Laßbergstraße (9.71 km)
```

```
Linie 2: Dorfstraße => Hornusstraße (7.98 km)
```

```
Linie 2: Hornusstraße => Dorfstraße (7.92 km)
```

```
Linie 3: Innsbrucker Straße => Munzinger Straße (9.20 km)
```

```
Linie 3: Munzinger Straße => Innsbrucker Straße (9.35 km)
```

```
Linie 4: Gundelfinger Straße => Technische Fakultät (8.11 km)
```

```
Linie 4: Technische Fakultät => Gundelfinger Straße (8.18 km)
```

```
Linie 5: Bollerstaudenstraße => Europaplatz (6.15 km)
```

```
Linie 5: Europaplatz => Bollerstaudenstraße (6.18 km)
```

```
Linie 7:   =>   (6.08 km)
```

```
Linie 7:   =>   (6.12 km)
```

```
me@home: 
```

Länge der Wege: 3. Versuch

```
# Vorverarbeitung: interessante Wege finden
ways = {}

def find_tram_ways(relation):
    if relation.tags.get('route') == 'tram':
        for m in relation.members:
            if m.type == 'w' and m.role == '':
                ways[m.ref] = 0

h = osmium.make_simple_handler(relation=find_tram_ways)
h.apply_file("freiburg.osm.pbf")

# weiter wie vorher...
```

Länge der Wege: 3. Versuch

```
# Vorverarbeitung:
```

```
ways = {}
```

```
def find_tram_ways()
```

```
    if relation.tag
```

```
        for m in re
```

```
            if m.ty
```

```
                way
```

```
h = osmium.make_sim
```

```
h.apply_file("freik
```

```
# weiter wie vorher
```

```
me@home: python3 route-list-length.py
```

```
Linie 1: Laßbergstraße => Moosweiher (9.52 km)
```

```
Linie 1: Moosweiher => Laßbergstraße (9.71 km)
```

```
Linie 2: Dorfstraße => Hornusstraße (7.98 km)
```

```
Linie 2: Hornusstraße => Dorfstraße (7.92 km)
```

```
Linie 3: Innsbrucker Straße => Munzinger Straße (9.20 km)
```

```
Linie 3: Munzinger Straße => Innsbrucker Straße (9.35 km)
```

```
Linie 4: Gundelfinger Straße => Technische Fakultät (8.11 km)
```


```
Linie 4: Technische Fakultät => Gundelfinger Straße (8.18 km)
```

```
Linie 5: Bollerstaudenstraße => Europaplatz (6.15 km)
```

```
Linie 5: Europaplatz => Bollerstaudenstraße (6.18 km)
```

```
Linie 7:   =>   (6.08 km)
```

```
Linie 7:   =>   (6.12 km)
```

```
me@home: 
```

Dateien mehrmals lesen.
Das spart Speicher.

Dateien verkleinern vor der Verarbeitung.

Geografische Ausschnitte (z.B. von der Geofabrik).

Nach Tags filtern (z.B. mit osmium-tool).

Danke

Source code: <https://github.com/osmcode/pyosmium>

Installation: `pip3 install osmium`

`lonvia@denofr.de`

Bilderquellen:

Karte Papiermühle Zittau, CC-by-SA 3.0, Deutsche Fotothek

[https://commons.wikimedia.org/wiki/File:Fotothek_df_rp-d_0330034_Zittau._Papiermühle_\(V\),_Ausschnitt_aus,_Die_Stadt_Zittau_1643_\(Sign.,_VIII_129\).jpg](https://commons.wikimedia.org/wiki/File:Fotothek_df_rp-d_0330034_Zittau._Papiermühle_(V),_Ausschnitt_aus,_Die_Stadt_Zittau_1643_(Sign.,_VIII_129).jpg)

Brücke, CC-by-SA 2.0, formulanone [https://commons.wikimedia.org/wiki/File:AB63nRoad-W00SHbus_\(28376977241\).jpg](https://commons.wikimedia.org/wiki/File:AB63nRoad-W00SHbus_(28376977241).jpg)

Flaschenabfüllung, CC-by-SA 3.0, Tomas er https://commons.wikimedia.org/wiki/File:Pol_Roger_disgorgement_line_8-finished_bottles-v2.jpg

Geschenkbbox, CC-by-SA 4.0, Emoji One <https://commons.wikimedia.org/wiki/File:Emojione-1F381.svg>