WISSEN
TECHNIK
LEIDENSCHAFT

Lab 1: Routing [Tutorial]

Navigation Systems

WS 2023/24

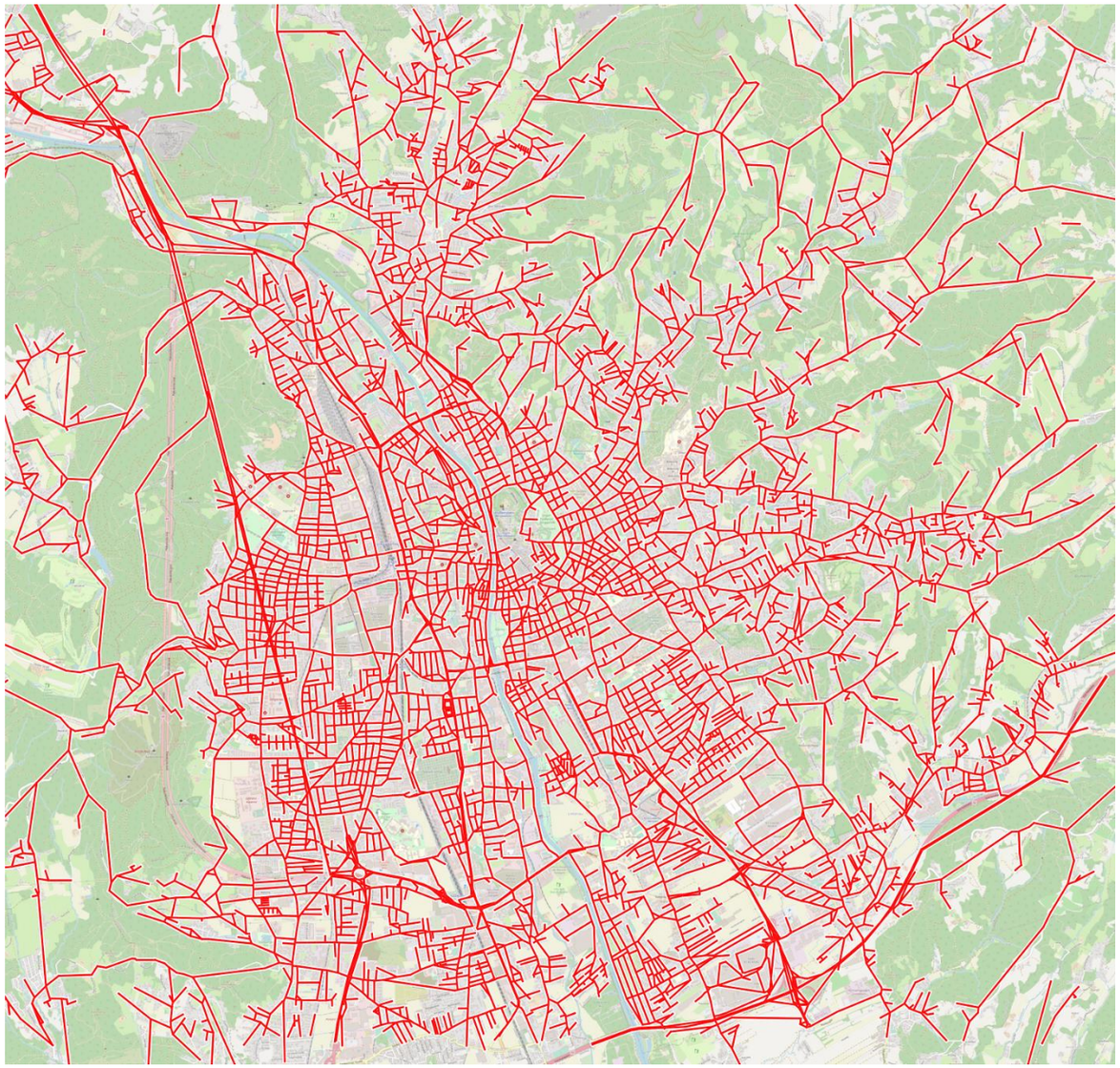
Robin Härtl
haertl@student.tugraz.at

Aim

- Compute optimal routes
- Dijkstra algorithm
- Start node: place where you live
- End nodes:
  - Basilika Mariatrost
  - Schloss Eggenberg
  - Shopping-Center Murpark

Given data

- nodepl.txt
  - Coordinates of nodes (φ and λ in degrees)
- nodelist.txt
  - Adjacency list of nodes
- arclist.txt
  - Adjacency list of arcs
  - Columns: time, distance, speed limit, clazz, flags



Tasks

- Dijkstra with both cost functions
  - Time
  - Distance

- Additionally, choose between:
- 2.1 Self-developed cost-function for Dijkstra’s algorithm
- 2.2 Heuristic A* algorithm

Deliverables

- ZIP folder
  - Source code, named "*lab01_ws2023_24_SURNAME.py*"
  - README.txt (contains add. information about runtime and possible errors/exceptions, version of used programming language etc.)
  - Additional files (if necessary)
- Presentation
  - Visualization of optimal routes (time & distance)
  - Evaluation of routes
  - Self-made cost function *or* A*-algorithm
  - 5 minutes video with audio commentary
- By **Dec 15th, 8.00 a.m.**

Peer-evaluation of presentations

- Watch and evaluate 3 presentations of your peers by **Dec 20th**
- Assessment criteria
  - Presentation style
  - Completeness
  - Visualization of results and interpretation
  - Own cost function *or* comparison to A*

Best presentation award

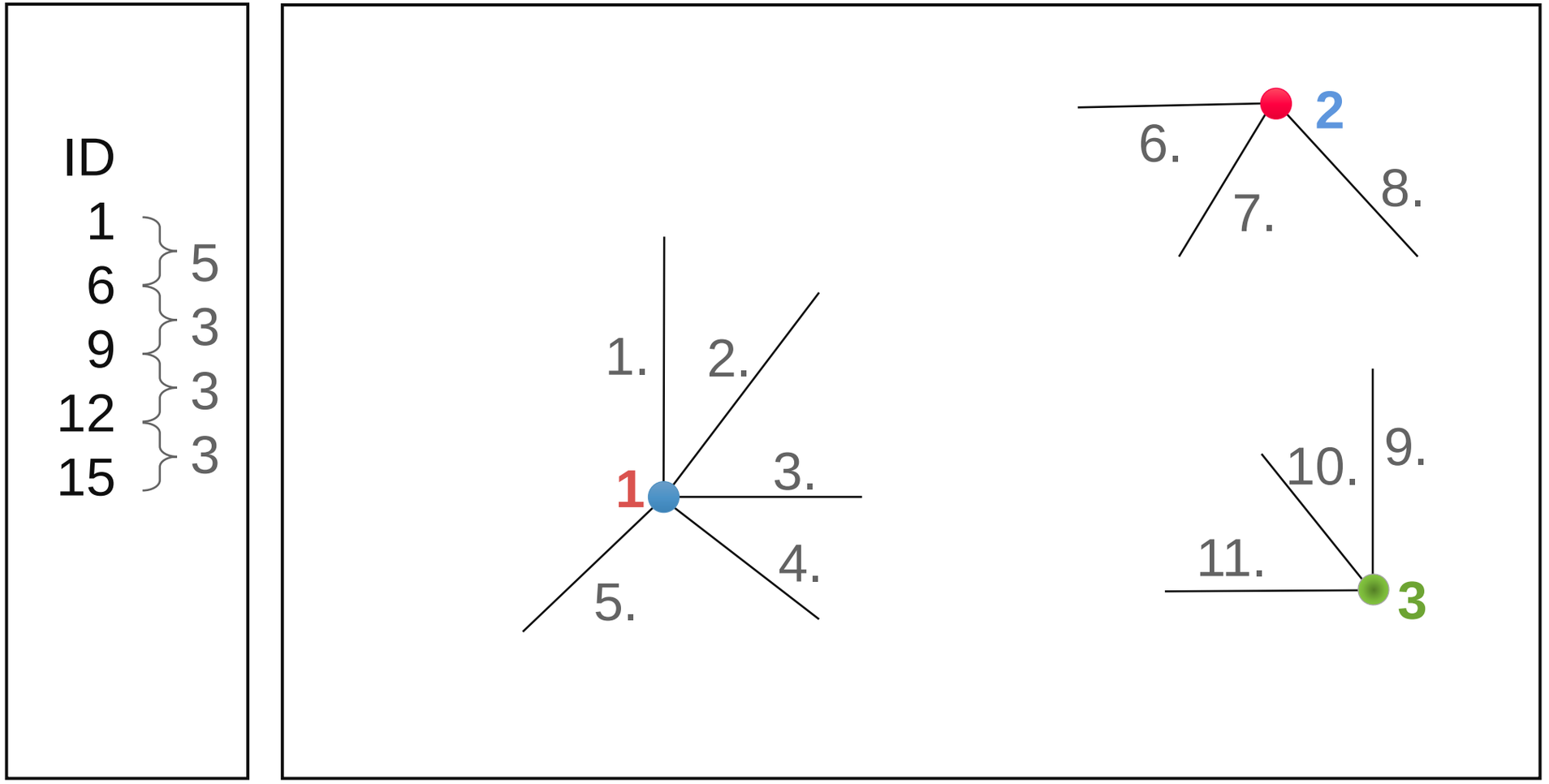
- 3 best presentations will be rewarded with a small prize and will be put on TUBE

Dijkstra Algorithm

Dijkstra algorithm

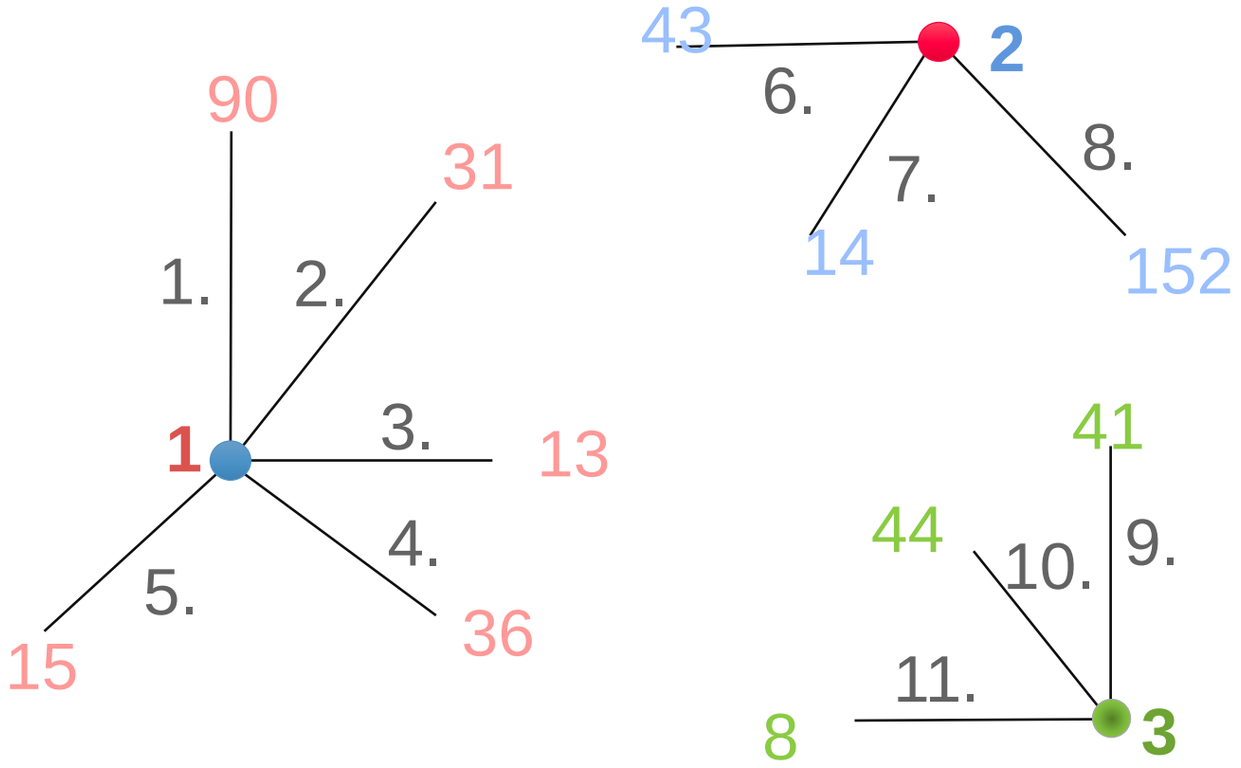
- Shortest path from one node to all other nodes
- Assign temporary or permanent labels during search process
- Predecessor list

Adjacency list of nodes



Adjacency list of arcs

v_j	c_{ij} [m]
90	63.20
31	78.77
13	40.25
36	78.26
15	81.44
43	34.78
14	81.44
152	81.44
41	77.18
44	11.90
8	45.89



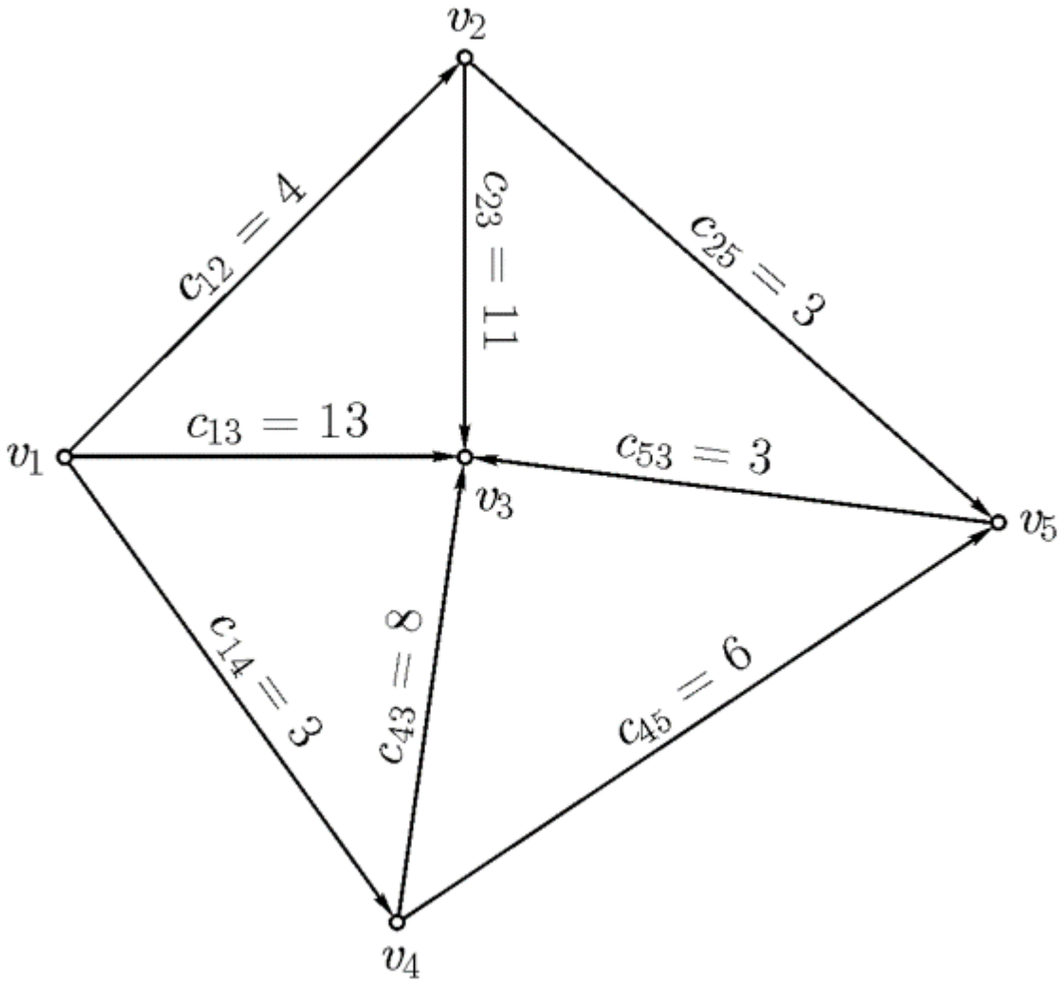
Tutorial example

nodelist_example.txt

1	1
2	4
3	6
4	6
5	8
6	9
7	

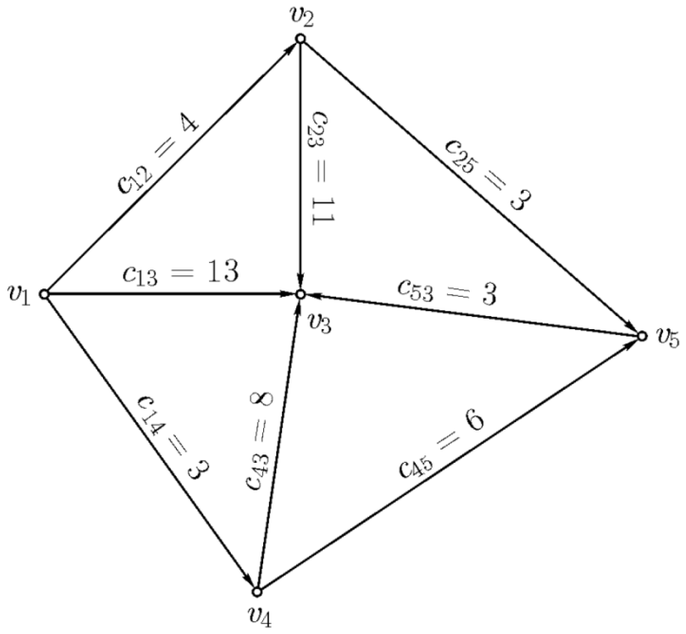
arclist_example.txt

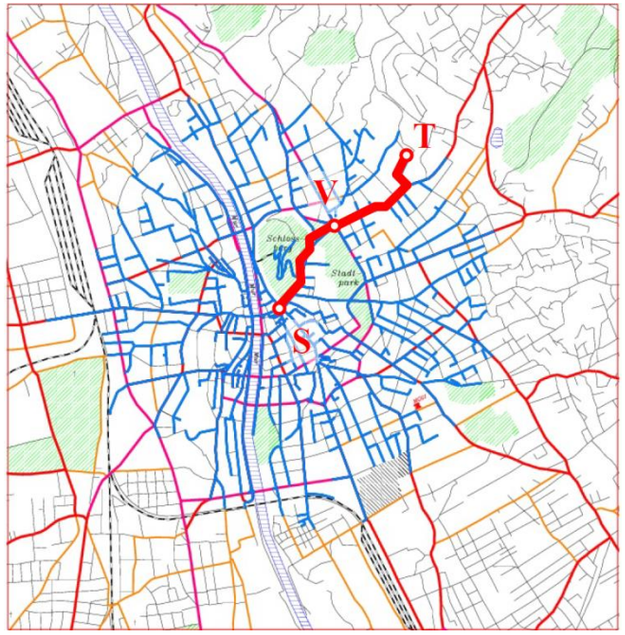
1	2	4
2	3	13
3	4	3
4	3	11
5	5	3
6	3	8
7	5	6
8	3	3



Dijkstra algorithm

- Check with small amount of data
  - Example from lecture





[python] Einlesen von Daten

Einlesen mit Numpy


```
import numpy as np
np.loadtxt()
```

- **`fname`** Name des Files, z.B. “`nodelist.txt`”
- **`delimiter`** String, welcher zur Trennung von Werten verwendet wird.
- **`skiprows`** Anzahl an Zeilen, welche übersprungen werden sollen

- mehr auf: https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html

Zugriff auf Daten: Numpy

- Zugriff auf bestimmte “Spalte”, z.B. auf die erste Spalte:
  - **`data1[:, 0]`**

- Zugriff auf bestimmte Zahl
  - **`data1[Zeile][Spalte]`**
  - z.B. Wert aus 1. Zeile und 6. Spalte: **`data1[0][5]`**

Einlesen mit Pandas


```
import pandas as pd
pd.read_csv()
```

- **`fname`**: Name des Files, z.B. `"nodelist.txt"`
- **`delim_whitespace`**: Gibt an, ob Leerzeichen (z. B. `'  '` oder `'\t'`) als Trennzeichen verwendet werden sollen (Wert `True` oder `False`).
- **`names=["spalte1", "spalte2"]`**: Spalten bekommen Bezeichnungen.

- mehr auf: pandas.read_csv — pandas 1.5.1 documentation (pydata.org)

Zugriff auf Daten: Pandas

- Zugriff auf ganze Spalte
  - **`data1["phi"]`**
- Zugriff auf einzelnen Wert:
  - **`data1["phi"][5]`**

Vorteil von Pandas: Spalten haben Namen = Zugriff auf Bezeichnungen

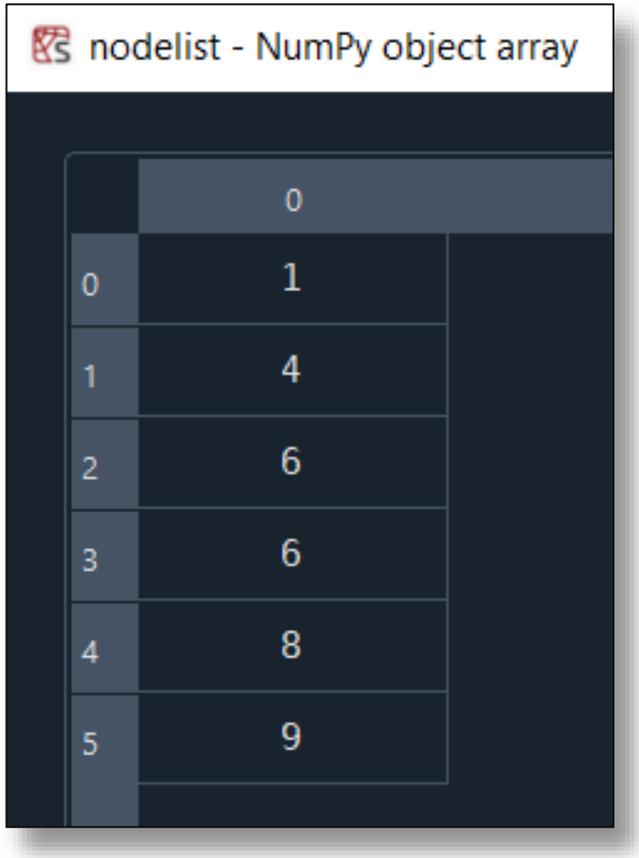
- Index soll mit 1 starten (und nicht mit 0)
  - **`data1.index = data1.index+1`**

Beispiel: Einlesen [Numpy]

```
nodelist =
np.loadtxt('nodelist_example.txt',
 skiprows=0, dtype='int')

print(nodelist[4])
    = 8

print(len(nodelist))
    = 6
```



Listen

```
templist=[]
templist.append(2)
templist.append(4)
templist.append(5)

print(templist)
    = [2, 4, 5]
```

```
templist =[2,5,7,8]
templist.remove(7)
print(templist)
    = [2, 5, 8]

del(templist[2])
print(templist)
    = [2, 5]
```

Ergebnisse und Visualisierung

Route aus Predecessor-Liste ermitteln

Geeignete Tools zur Visualisierung finden

- φ/λ-Achsverhältnis beachten
- Beschriftungen
- Angemessene Darstellung

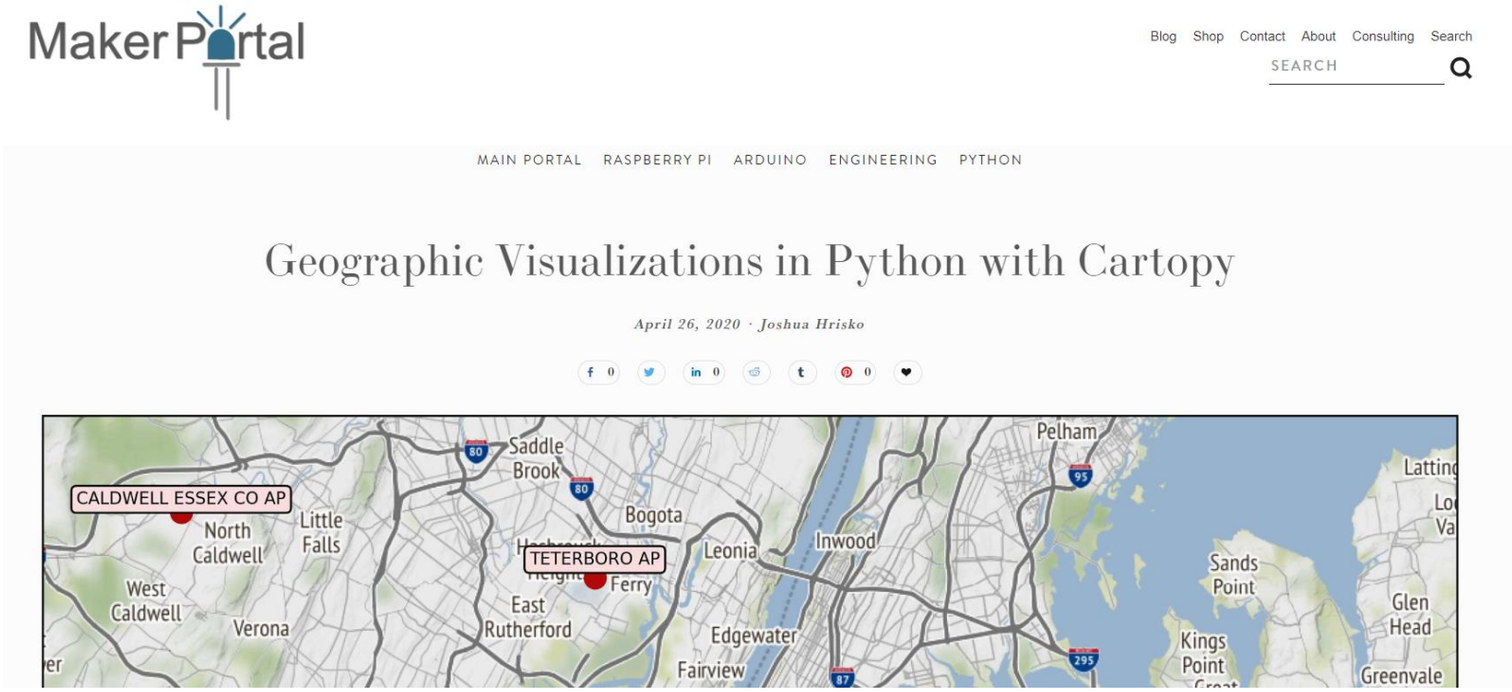
Visualisierung

- QGIS (o.ä.)
  - *Getrennte Textdatei als Layer hinzufügen*
  - *„Punkte zu Weg“*

- Python
  - Bibliothek wie z.B. Cartopy verwenden
  - Bild einlesen und Ecken der Bilder georeferenzieren

Visualisierung mit Cartopy

https://makersportal.com/blog/2020/4/24/geographic-visualizations-in-python-with-cartopy



- **Abgabe bis 15. Dezember, 8:00 Uhr** im TeachCenter
  - **Video** (*.mp4*), max. 5 Min!
  - **ZIP-Ordner**
    - Source-Code („*lab01_ws2023_24_SURNAME.py*“*)
    - ReadMe-File („*README.txt*“) mit allen Informationen, die zum Starten des Programms benötigt werden
    - ggf. zusätzlich benötigte Dateien

- Anschließend: Peer Review bis 20. Dezember
  - Jeweils Bewertung 3 anderer Präsentationen

**) Dateiendung je nach verwendeter Programmier- oder Skriptsprache*