In [1]:
```
import os
import glob
import json
import rtree

import numpy as np
import pandas as pd

import geopandas
import contextily as cx
from shapely.geometry import LineString, Point
from shapely.ops import split, unary_union
import warnings
from shapely.errors import ShapelyDeprecationWarning
warnings.filterwarnings("ignore", category=ShapelyDeprecationWarning)


from datetime import datetime
from datetime import timedelta

import matplotlib.pyplot as plt
import pytz
```

define load bike data functions

In [2]:
```
def loadBikeDataFileNames(folder):
    bikeFilesName = glob.glob(f'{folder}/*.json')
    bikeFilesName.sort()
    return bikeFilesName

def loadBikeData(bikeDataFileName):
    with open(bikeDataFileName, 'r') as file:
        content = json.load(file)
        bikes = content['countries'][0]['cities'][0]['places']
        bikes = pd.DataFrame(bikes)
        bikes = preprocessBikeData(bikes)

        timestamp = bikeDataFileName[-15:-5]
        return bikes, timestamp

def preprocessBikeData(df):
    df = df[df['bikes'].apply(lambda x: x>0)] # if not, explode will yiel
    df = df[['lat', 'lng', 'bike_numbers']]
    df = df.explode('bike_numbers')

    return df

bikeDataFileNames = loadBikeDataFileNames("data/2022.08.28")
bikeData, timestamp = loadBikeData(bikeDataFileNames[0])
bikeData
```

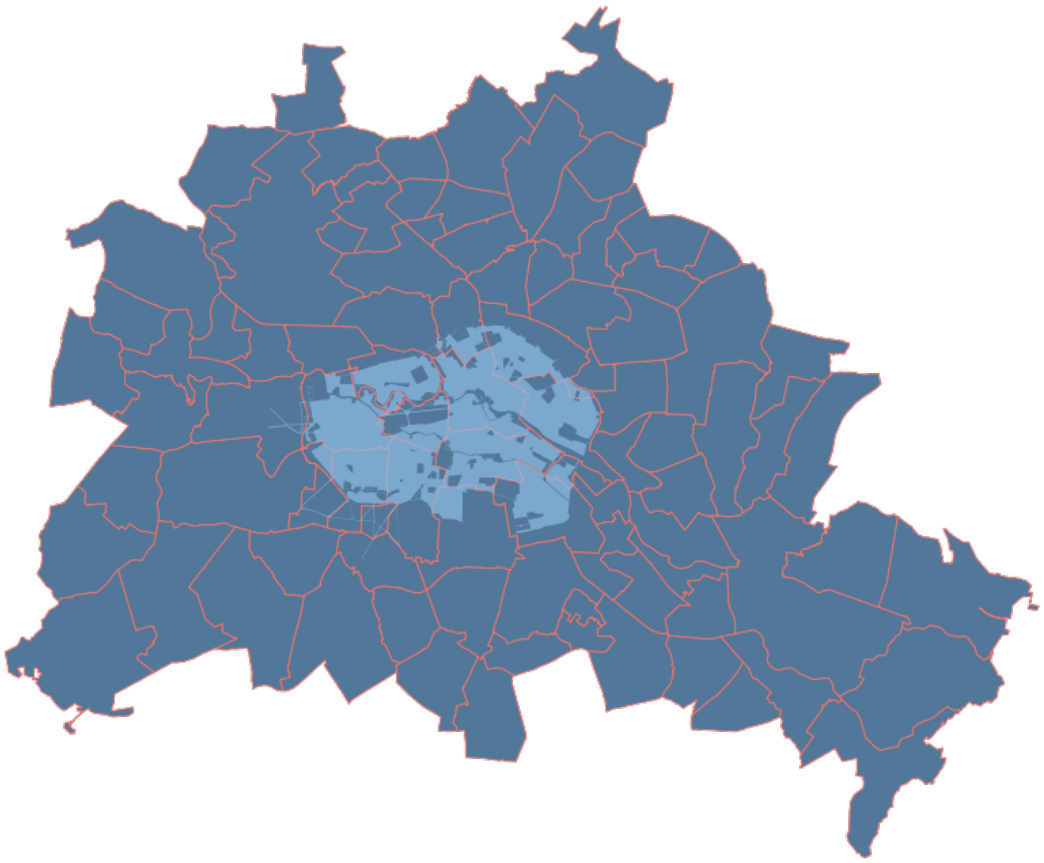
Out[2]:

	lat	lng	bike_numbers
0	52.504157	13.335328	16480
0	52.504157	13.335328	14257
1	52.496986	13.291210	16199
2	52.498323	13.296157	15195
2	52.498323	13.296157	13964
...	...	...	...
2596	52.484333	13.439081	16811
2597	52.518644	13.450926	18122
2598	52.485853	13.360917	16265
2599	52.485782	13.360962	19452
2600	52.480787	13.424276	15598

3684 rows × 3 columns

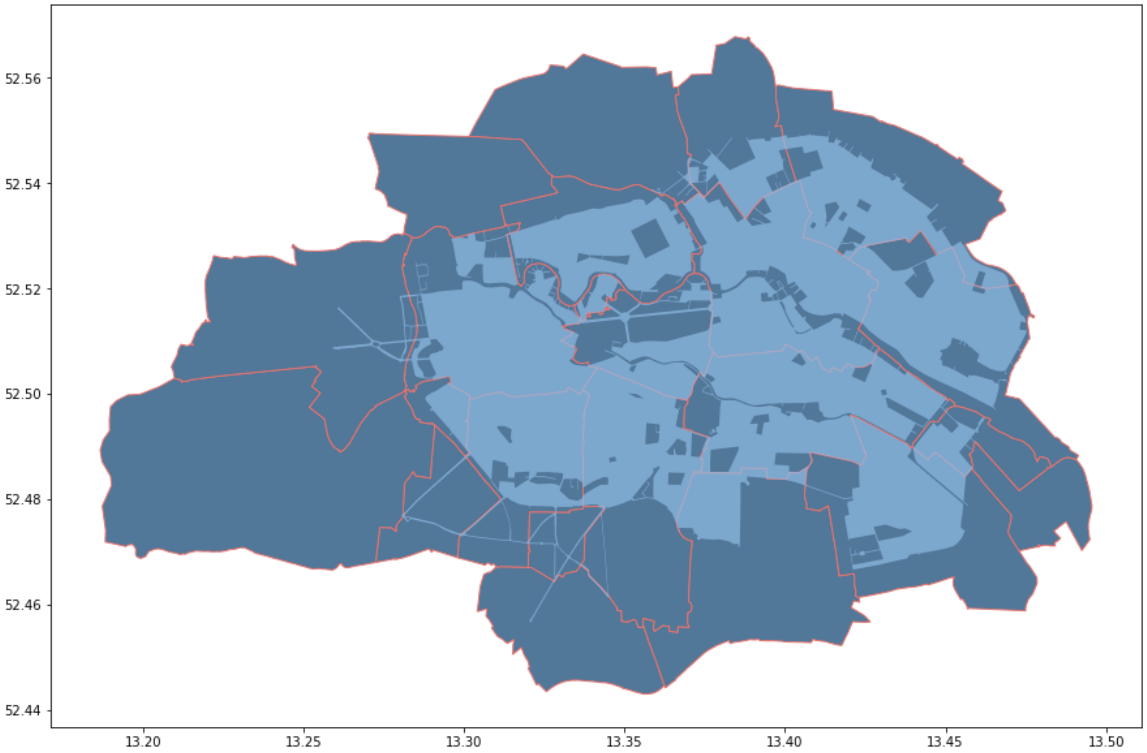
Load polygons

In [3]:
```
colors = ['#517799', '#E6746F', '#91BDE6', '#E68D2C']

def initKiezePlot(kieze):
    kiezeAx = kieze.plot(color=colors[0], edgecolor=colors[1], figsize=(1
    kiezeAx.set_axis_off();
    return kiezeAx

def initFlexZonePlot(flexZone, ax=None):
    return flexZone.plot(ax=ax, color=colors[2], alpha=0.7)

kieze = geopandas.read_file("lor_ortsteile.geojson")
kiezeAx = initKiezePlot(kieze)
flexZone = geopandas.read_file("flexzone_bn.json")
flexZoneAx = initFlexZonePlot(flexZone, kiezeAx)
```



show only kieze that insersects flexzone

In [4]:
```
kiezeIntersectingFlexzone = kieze[kieze['gml_id'].isin(kieze.overlay(flex
kiezeIntersectingFlexzoneAx = kiezeIntersectingFlexzone.plot(color=colors
flexZoneAx = initFlexZonePlot(flexZone, kiezeIntersectingFlexzoneAx)
```



Bike locations

In [5]:
```
def getPointGdfFromLatAndLngDf(df, latName, lngName):
#     df['geometry'] = df.apply(lambda x: Point(x[lngName], x[latName]),
    df = df.assign(geometry = df.apply(lambda x: Point(x[lngName], x[latN
    gdf = geopandas.GeoDataFrame(df, geometry='geometry', crs="EPSG:4326"

    return gdf

bikeData, timestamp = loadBikeData(bikeDataFileNames[0])
bikeDataGdf = getPointGdfFromLatAndLngDf(bikeData, 'lat', 'lng')
```

In [6]:
```
kiezeAx = initKiezePlot(kieze)
flexZoneAx = initFlexZonePlot(flexZone, kiezeAx)
bikeDataGdf.plot(ax=flexZoneAx, color=colors[3], markersize=5)
```

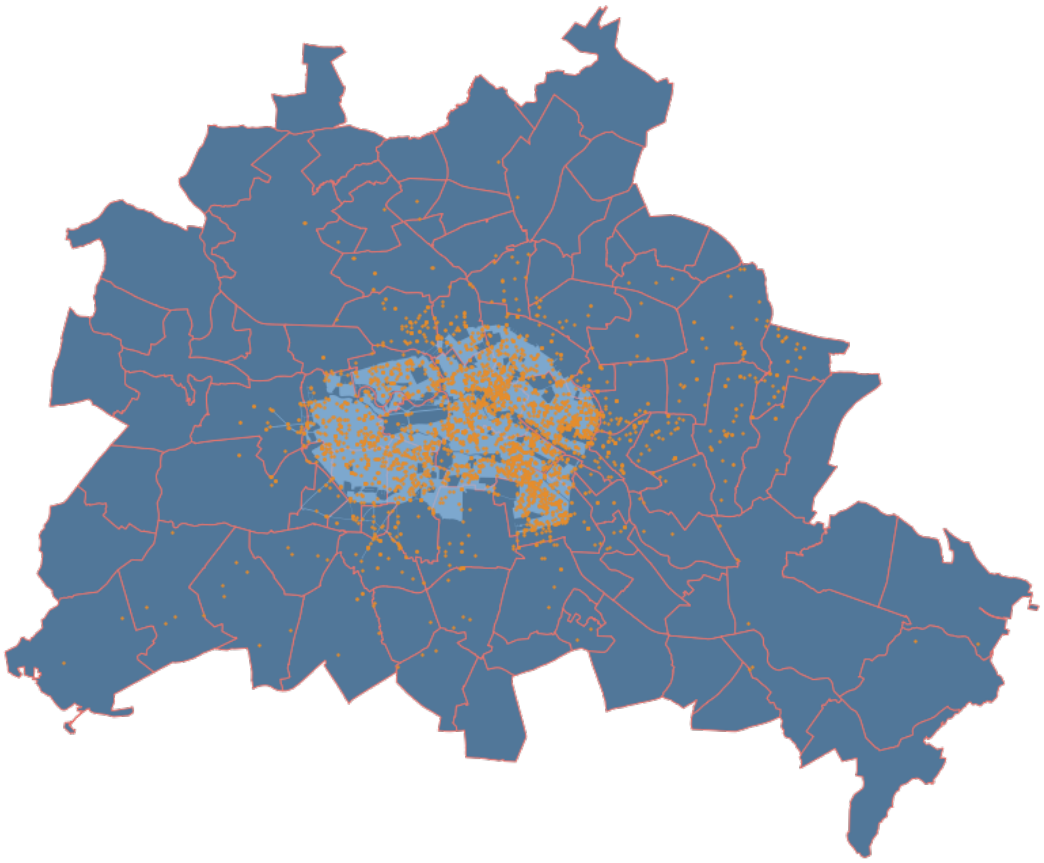
Out[6]:
```
<AxesSubplot:>
```

need to restrict the locations

Method 1: box filter around kiezes

In [7]:
```
kiezeAx = initKiezePlot(kieze)
flexZoneAx = initFlexZonePlot(flexZone, kiezeAx)
bikeDataGdf.overlay(kieze).plot(ax=flexZoneAx, alpha=1, color=colors[3],
```

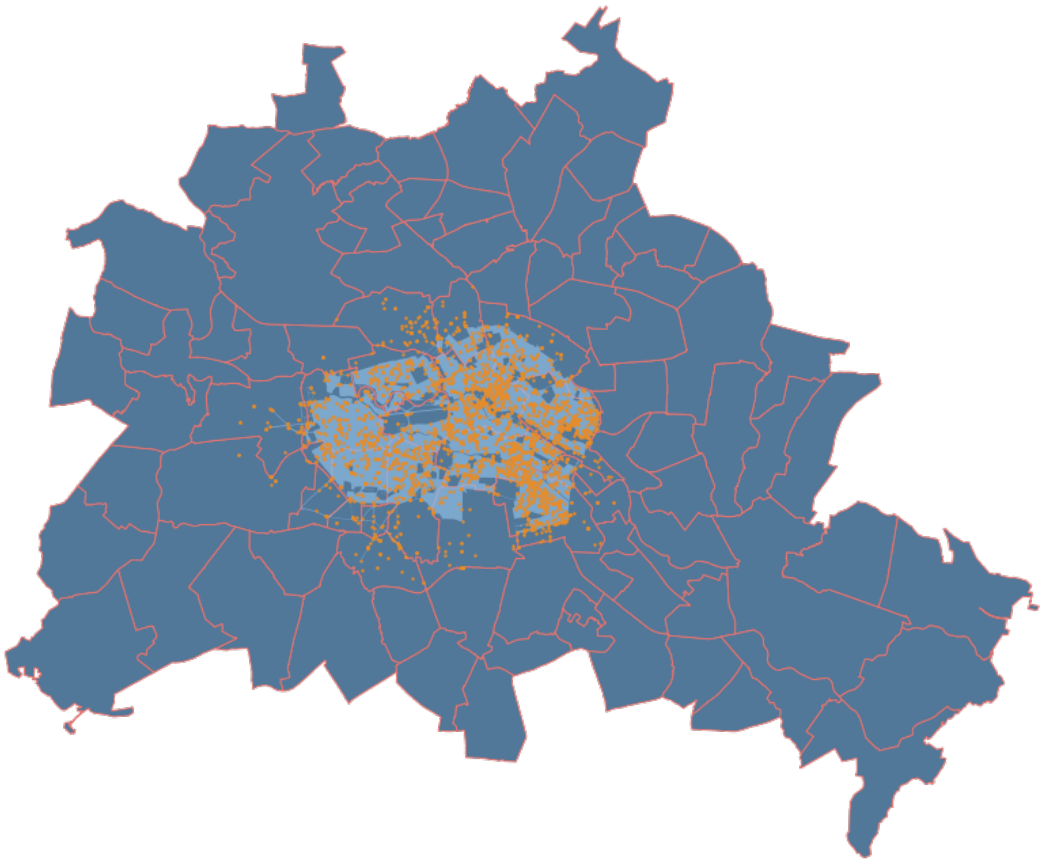
Out[7]:
```
<AxesSubplot:>
```



Method 2: kieze that intersect the flexzone

In [8]:
```
kiezeAx = initKiezePlot(kieze)
flexZoneAx = initFlexZonePlot(flexZone, kiezeAx)
bikeDataGdf.overlay(kiezeIntersectingFlexzone).plot(ax=flexZoneAx, alpha=
```

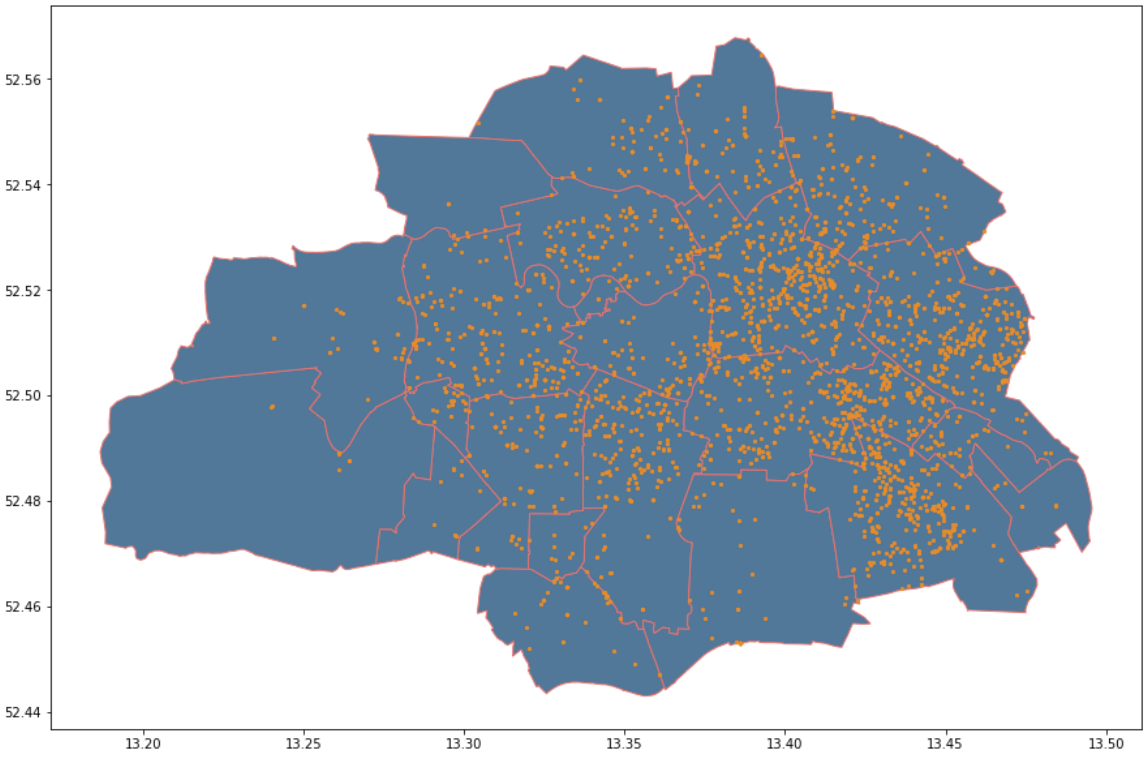
Out[8]:
```
<AxesSubplot:>
```



only show the filtered kieze and bikes

In [9]:
```
kiezeIntersectingFlexzoneAx = kiezeIntersectingFlexzone.plot(color=colors
bikeDataGdf.overlay(kiezeIntersectingFlexzone).plot(ax=kiezeIntersectingF
```

Out[9]:
```
<AxesSubplot:>
```



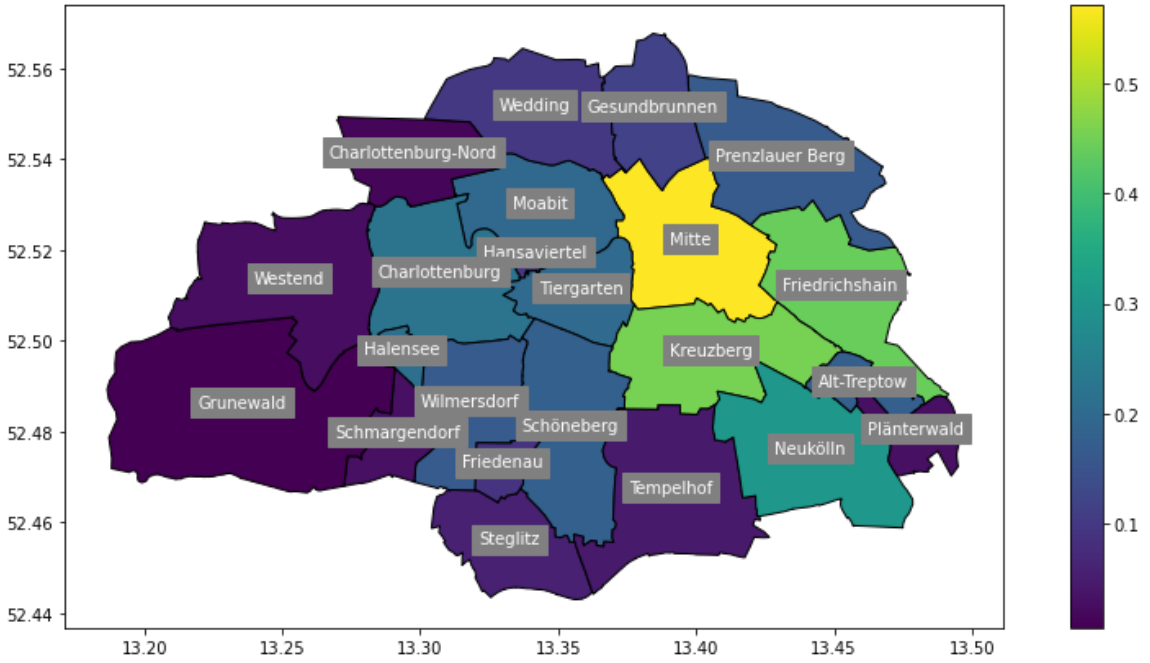
plot heatmap of bikes per kiez density

In [10]:
```
bikeAmount = kiezeIntersectingFlexzone.T.apply(lambda x: bikeDataGdf.with
bikeDensity = bikeAmount/kiezeIntersectingFlexzone['FLAECHE_HA']

kiezeIntersectingFlexzone = kiezeIntersectingFlexzone.assign(bikeAmount=b
kiezeIntersectingFlexzone = kiezeIntersectingFlexzone.assign(bikeDensity=
kiezeIntersectingFlexzone.head()
```

Out[10]:

	gml_id	spatial_name	spatial_alias	spatial_type	OTEIL	BEZIRK	FLAECHE_H
0	re_ortsteil.0101	0101	Mitte	Polygon	Mitte	Mitte	1063.87
1	re_ortsteil.0102	0102	Moabit	Polygon	Moabit	Mitte	768.79
2	re_ortsteil.0103	0103	Hansaviertel	Polygon	Hansaviertel	Mitte	52.53
3	re_ortsteil.0104	0104	Tiergarten	Polygon	Tiergarten	Mitte	516.06
4	re_ortsteil.0105	0105	Wedding	Polygon	Wedding	Mitte	919.91

In [11]:
```
ax = kiezeIntersectingFlexzone.plot(column='bikeDensity', cmap='viridis',
_ = kiezeIntersectingFlexzone.apply(lambda x: ax.annotate(text=x['OTEIL']
```



Bike Tours

In [12]:
```
def buildBikeTourPairs(bikeData, skip):
    return list(zip(bikeData[::skip][:-1:],bikeData[::skip][1::]))

bikeTours = buildBikeTourPairs(bikeDataFileNames, 6)
bikeTours[:2]
```

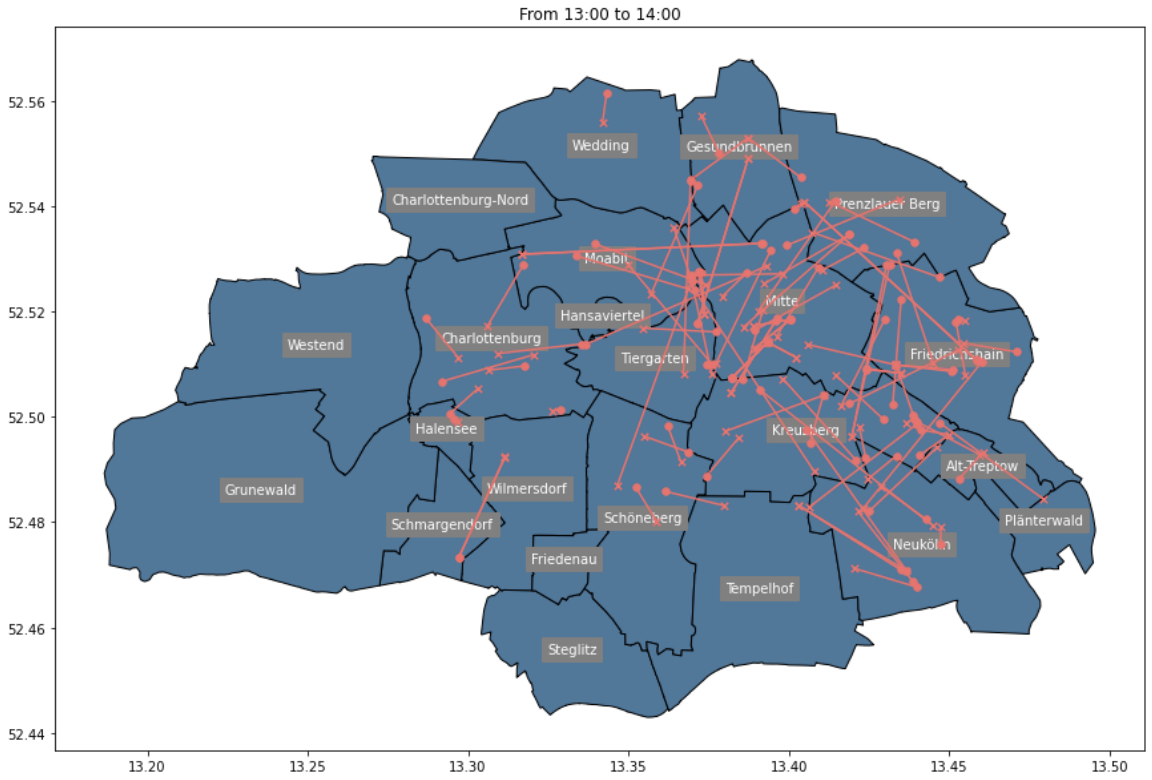
Out[12]:
```
[('data/2022.08.28/1661637604.json', 'data/2022.08.28/1661641201.json'),
 ('data/2022.08.28/1661641201.json', 'data/2022.08.28/1661644803.json')]
```

In [13]:
```
def getLineString(x):
    return LineString([(x['lng_x'], x['lat_x']), (x['lng_y'], x['lat_y'])

def createBikeTour(bikeData1, bikeData2):
    dfMerged = bikeData1.merge(bikeData2, on='bike_numbers')
    dfLines = dfMerged.apply(getLineString, axis=1)
    dfMerged['lines'] = dfLines

    gdf = geopandas.GeoDataFrame(dfMerged, geometry='lines', crs="EPSG:43
    gdf['length'] = gdf['lines'].to_crs(crs="EPSG:25832").length # first

    gdf = gdf[gdf['length'] > 100] # remove bikes that moved under 100 me

    return gdf

def getTimeFromTimestamp(ts):
    return datetime.fromtimestamp(int(ts)).astimezone(pytz.timezone('Euro

bikes1, ts1 = loadBikeData(bikeTours[13][0])
bikes2, ts2 = loadBikeData(bikeTours[13][1])
bikeTour = createBikeTour(bikes1, bikes2)
bikeTourFiltered = bikeTour[bikeTour.within(unary_union(kiezeIntersecting

startPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_x', 'lng_
endPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_y', 'lng_y'
```

Plot 1: bike tours lines

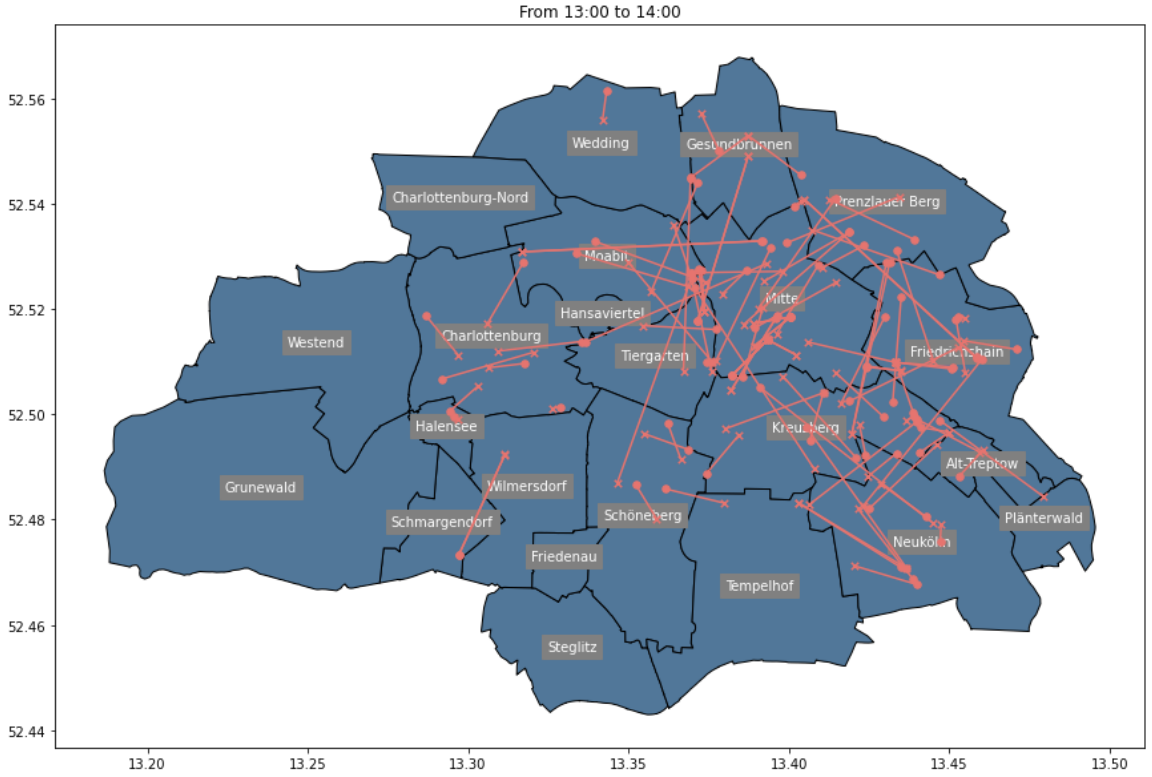
In [14]:
```
kiezeIntersectingFlexzoneAx = kiezeIntersectingFlexzone.plot(zorder=0,col
_ = kiezeIntersectingFlexzone.apply(lambda x: kiezeIntersectingFlexzoneAx

bikeTourFilteredAx = bikeTourFiltered.plot(zorder=2, ax=kiezeIntersecting
bikeTourFilteredAx.set_title(f"From {getTimeFromTimestamp(ts1)} to {getTi

startPointsAx = startPoints.plot(zorder=3, ax=bikeTourFilteredAx, color=c
endPoints.plot(zorder=4, ax=startPointsAx, color=colors[1], marker='x', m
```

Out[14]:
```
<AxesSubplot:title={'center':'From 13:00 to 14:00'}>
```



In [15]:
```
kiezeIntersectingFlexzoneAx = kiezeIntersectingFlexzone.plot(zorder=0,col
_ = kiezeIntersectingFlexzone.apply(lambda x: kiezeIntersectingFlexzoneAx

bikeTourFilteredAx = bikeTourFiltered.plot(zorder=2, ax=kiezeIntersecting
bikeTourFilteredAx.set_title(f"From {getTimeFromTimestamp(ts1)} to {getTi

startPointsAx = startPoints.plot(zorder=3, ax=bikeTourFilteredAx, color=c
endPoints.plot(zorder=4, ax=startPointsAx, color=colors[1], marker='x', m
```

Out[15]:
```
<AxesSubplot:title={'center':'From 13:00 to 14:00'}>
```

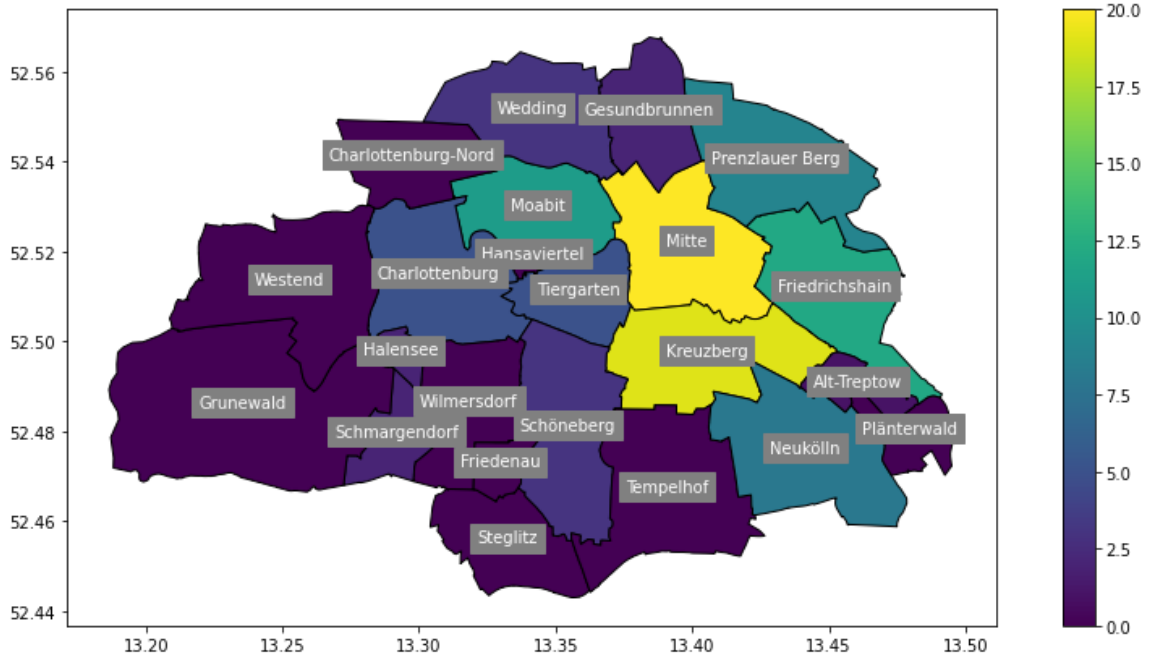


Plot 2: heatmap of touring

In [16]:
```
kiezeIntersectingFlexzone = kiezeIntersectingFlexzone.assign(tourStart=ki
kiezeIntersectingFlexzone = kiezeIntersectingFlexzone.assign(tourEnd=kiez
kiezeIntersectingFlexzone = kiezeIntersectingFlexzone.assign(tourSum=(kie
```

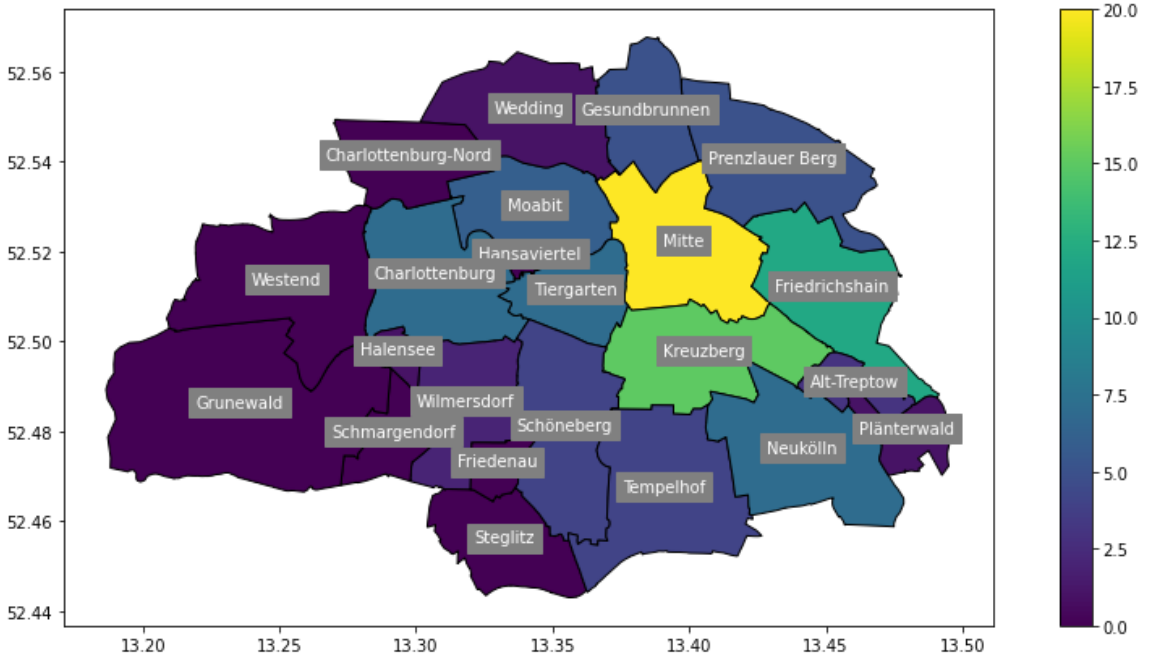
Tour start

In [17]:
```
ax = kiezeIntersectingFlexzone.plot(column='tourStart', cmap='viridis', v
_ = kiezeIntersectingFlexzone.apply(lambda x: ax.annotate(text=x['OTEIL']
```



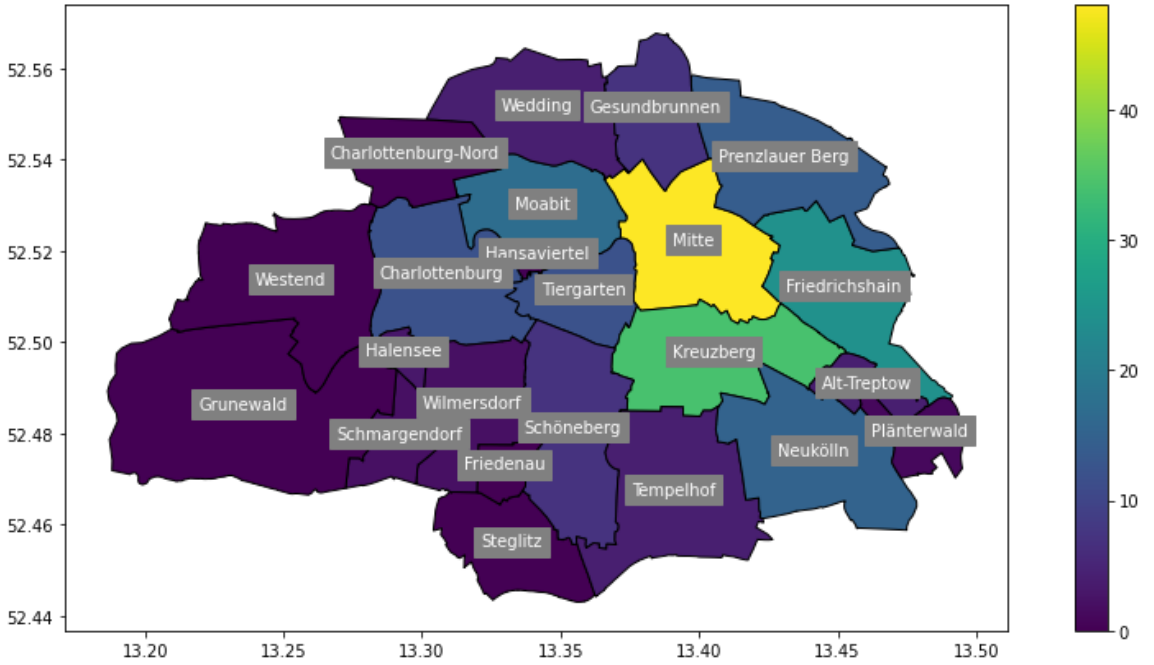
tour end

In [18]:
```
ax =kiezeIntersectingFlexzone.plot(column='tourEnd', cmap='viridis', vmax
_ = kiezeIntersectingFlexzone.apply(lambda x: ax.annotate(text=x['OTEIL']
```



tour sum

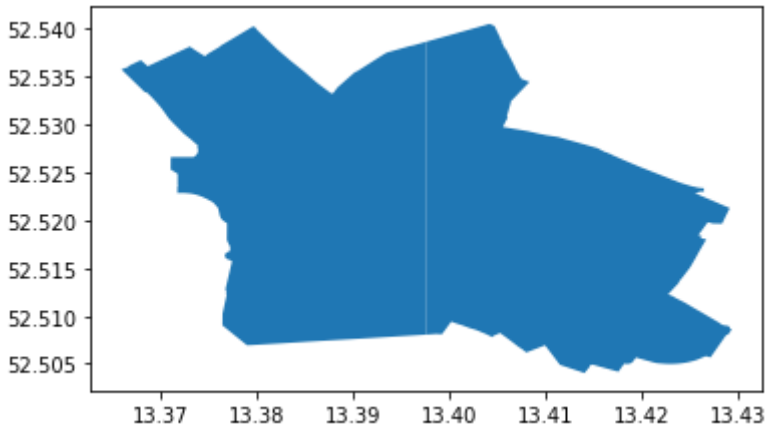
In [19]:
```
ax =kiezeIntersectingFlexzone.plot(column='tourSum', cmap='viridis', edge
_ = kiezeIntersectingFlexzone.apply(lambda x: ax.annotate(text=x['OTEIL']
```



Plot 3: heatmap by start and stop split

In [20]:
```
def isSplittedPolygonOnTheLeft(splittedPolygon, polygon):
    return splittedPolygon.bounds[0] == polygon.bounds[0]

def splitPolygonIntoHalf(p):
    minx, miny, maxx, maxy = p.bounds
    splitterx = minx + (maxx-minx)/2
    splitter = LineString([[splitterx, miny], [splitterx, maxy]])

    splittedPolygon = split(p, splitter)

    if(isSplittedPolygonOnTheLeft(splittedPolygon.geoms[0], splittedPolyg
        return geopandas.GeoSeries([splittedPolygon.geoms[0], splittedPol
    else:
        return geopandas.GeoSeries([splittedPolygon.geoms[1], splittedPol


splitPolygonIntoHalf(kiezeIntersectingFlexzone['geometry'].iloc[0]).plot(
```

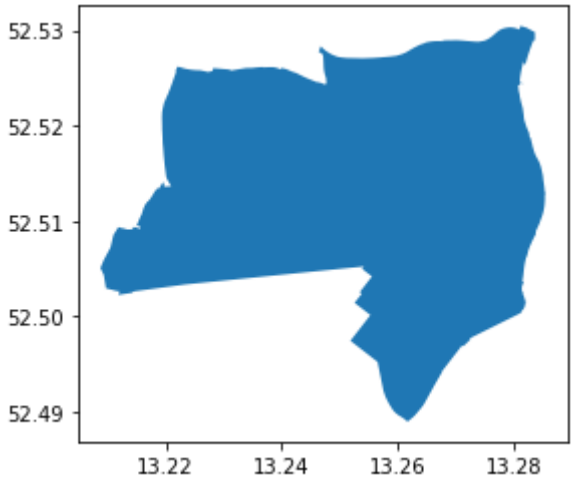
Out[20]: <AxesSubplot:>



problem with one polygon

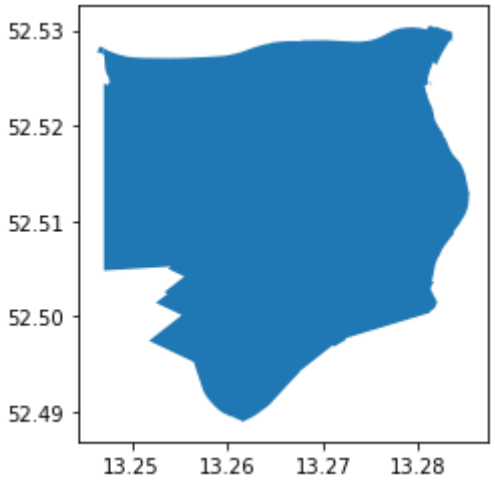
In [21]:
```
kiezeIntersectingFlexzone.loc[25:25].plot() # original
```

Out[21]: <AxesSubplot:>



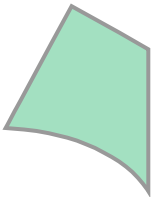
In [22]:
```
faultyPolygon = splitPolygonIntoHalf(kiezeIntersectingFlexzone['geometry'
faultyPolygon.plot()
```

Out[22]: <AxesSubplot:>



In [23]:
```
faultyPolygon[0]
```

Out[23]:



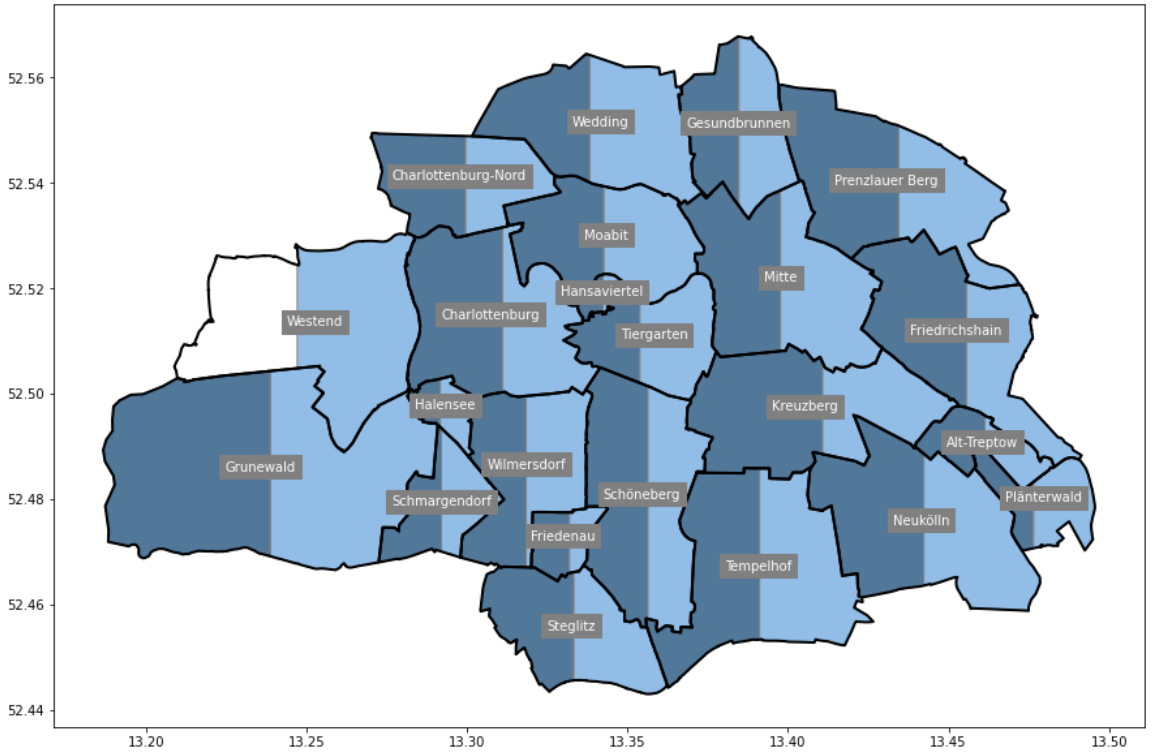
In [24]:
```
print(faultyPolygon[0])
```

```
POLYGON ((13.24707419926635 52.5273623297785, 13.247069207254091 52.52737
115214868, 13.247046290181563 52.52740212285497, 13.247021070989664 52.52
743242077715, 13.246991412059765 52.52746118419698, 13.246959623863546 5
2.52748908879089, 13.246923782539975 52.52751510232846, 13.24688601439824
52.527540099304005, 13.24684465306203 52.527562882776316, 13.246801623230
015 52.52758448626836, 13.246755531610269 52.52760359675062, 13.246708052
44819 52.527621398900294, 13.246658100713464 52.52763647559928, 13.246562
847578733 52.527659606624844, 13.246499838652138 52.52767127984153, 13.24
6435239123254 52.52767907494787, 13.246342015986535 52.52768893019432, 1
3.246352023567553 52.52770737580036, 13.246388732138362 52.52777509060569
4, 13.246396607581023 52.52778962020635, 13.24641766687021 52.52782846484
415, 13.246578615375943 52.52812529543564, 13.246680708352573 52.52831361
294615, 13.246928001673572 52.52817384806656, 13.247041171563687 52.52810
99123752, 13.247060657620079 52.52809889446365, 13.24707419926635 52.5280
9124681867, 13.24707419926635 52.5273623297785))
```

split all kieze

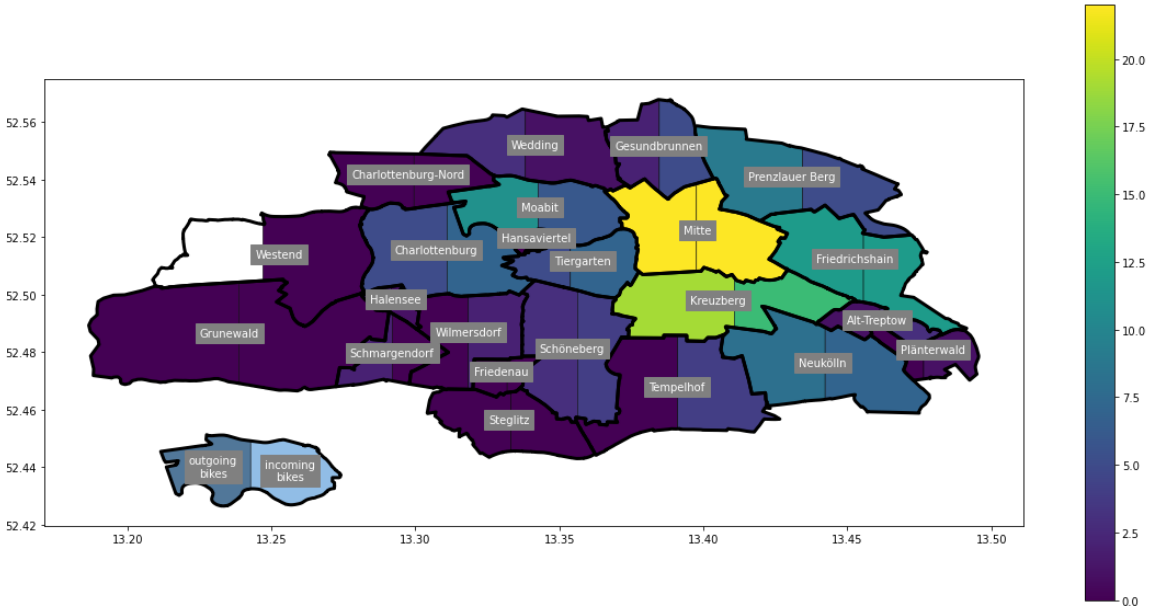
In [25]:
```
splittedKieze = kiezeIntersectingFlexzone.apply(lambda x: splitPolygonInt

leftSplitKieze = kiezeIntersectingFlexzone.assign(leftSplit=splittedKieze
leftSplitKieze.set_geometry('leftSplit', inplace=True)

rightSplitKieze = kiezeIntersectingFlexzone.assign(rightSplit=splittedKie
rightSplitKieze.set_geometry('rightSplit', inplace=True)
```

In [26]:
```
leftSplitAx = leftSplitKieze.plot(color=colors[0], edgecolor='grey', figs
rightSplitAx = rightSplitKieze.plot(ax = leftSplitAx, color=colors[2], ed

kiezeBorderAx = kiezeIntersectingFlexzone.plot(ax = rightSplitAx, facecol
_ = kiezeIntersectingFlexzone.apply(lambda x: kiezeBorderAx.annotate(text
```



feeding data

In [27]:
```
leftSplitKieze = leftSplitKieze.assign(tourStartLeftSplitKieze=leftSplitK
rightSplitKieze = rightSplitKieze.assign(tourStartRigthSplitKieze=rightSp

exampleSplit = geopandas.GeoSeries([leftSplitKieze.loc[1]['leftSplit'], r
exampleSplit = exampleSplit.translate(xoff=-0.1, yoff=-0.09)


leftSplitAx = leftSplitKieze.plot(column='tourStartLeftSplitKieze', cmap=
rightSplitAx = rightSplitKieze.plot(ax = leftSplitAx, column='tourStartRi
kiezeBorderAx = kiezeIntersectingFlexzone.plot(ax = rightSplitAx, facecol
_ = kiezeIntersectingFlexzone.apply(lambda x: kiezeBorderAx.annotate(text


exampleSplitLeftAx = exampleSplit[0:1].plot(ax=kiezeBorderAx, color=color
exampleSplitRightAx = exampleSplit[1:2].plot(ax=kiezeBorderAx, color=colo
_ = geopandas.GeoSeries(unary_union(exampleSplit)).plot(ax=kiezeBorderAx,
_ = exampleSplitLeftAx.annotate(text='outgoing\nbikes', xy=exampleSplit[0
_ = exampleSplitLeftAx.annotate(text='incoming\nbikes', xy=exampleSplit[1
```



mass production

In [28]:
```
stop()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last
Input In [28], in <cell line: 1>()
----> 1 stop()

NameError: name 'stop' is not defined
```

bike tour lines

In [ ]:
```
bikeTours = buildBikeTourPairs(bikeDataFileNames, 6)
# bikeTours = bikeTours[:2]

for bikeTour in bikeTours:
    bikes1, ts1 = loadBikeData(bikeTour[0])
    bikes2, ts2 = loadBikeData(bikeTour[1])
    bikeTour = createBikeTour(bikes1, bikes2)
    bikeTourFiltered = bikeTour[bikeTour.within(unary_union(kiezeIntersec


    startPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_x', '
    endPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_y', 'ln

    # plot
    kiezeIntersectingFlexzoneAx = kiezeIntersectingFlexzone.plot(zorder=0
    _ = kiezeIntersectingFlexzone.apply(lambda x: kiezeIntersectingFlexzo

    bikeTourFilteredAx = bikeTourFiltered.plot(zorder=2, ax=kiezeIntersec
    bikeTourFilteredAx.set_title(f"From {getTimeFromTimestamp(ts1)} to {g

    startPointsAx = startPoints.plot(zorder=3, ax=bikeTourFilteredAx, col
    endPoints.plot(zorder=4, ax=startPointsAx, color=colors[1], marker='x


    plt.savefig(f"export/60_min_tick_tourline/{getTimeFromTimestamp(ts1)}
```

bike tour heatmap

In [ ]:
```
bikeTourAmountStarts = []
bikeTourAmountEnds = []

bikeTours = buildBikeTourPairs(bikeDataFileNames, 6)
# bikeTours = bikeTours[:2]

for bikeTour in bikeTours:
    bikes1, ts1 = loadBikeData(bikeTour[0])
    bikes2, ts2 = loadBikeData(bikeTour[1])
    bikeTour = createBikeTour(bikes1, bikes2)
    bikeTourFiltered = bikeTour[bikeTour.within(unary_union(kiezeIntersec


    startPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_x', '
    endPoints = getPointGdfFromLatAndLngDf(bikeTourFiltered[['lat_y', 'ln

    splittedKieze = kiezeIntersectingFlexzone.apply(lambda x: splitPolygo

    leftSplitKieze = kiezeIntersectingFlexzone.assign(leftSplit=splittedK
    leftSplitKieze.set_geometry('leftSplit', inplace=True)
    rightSplitKieze = kiezeIntersectingFlexzone.assign(rightSplit=splitte
    rightSplitKieze.set_geometry('rightSplit', inplace=True)

    leftSplitKieze = leftSplitKieze.assign(tourStartLeftSplitKieze=leftSp
    rightSplitKieze = rightSplitKieze.assign(tourStartRigthSplitKieze=rig


    bikeTourAmountStarts.append(leftSplitKieze['tourStartLeftSplitKieze']
    bikeTourAmountEnds.append(rightSplitKieze['tourStartRigthSplitKieze']


    exampleSplit = geopandas.GeoSeries([leftSplitKieze.loc[1]['leftSplit'
    exampleSplit = exampleSplit.translate(xoff=-0.1, yoff=-0.09)


    # plot
    leftSplitAx = leftSplitKieze.plot(column='tourStartLeftSplitKieze', c
    rightSplitAx = rightSplitKieze.plot(ax = leftSplitAx, column='tourSta
    kiezeBorderAx = kiezeIntersectingFlexzone.plot(ax = rightSplitAx, fac
    _ = kiezeIntersectingFlexzone.apply(lambda x: kiezeBorderAx.annotate(
    kiezeBorderAx.set_title(f"From {getTimeFromTimestamp(ts1)} to {getTim


    exampleSplitLeftAx = exampleSplit[0:1].plot(ax=kiezeBorderAx, color=c
    exampleSplitRightAx = exampleSplit[1:2].plot(ax=kiezeBorderAx, color=
    _ = geopandas.GeoSeries(unary_union(exampleSplit)).plot(ax=kiezeBorde
    _ = exampleSplitLeftAx.annotate(text='outgoing\nbikes', xy=exampleSpl
    _ = exampleSplitLeftAx.annotate(text='incoming\nbikes', xy=exampleSpl

    plt.savefig(f"export/60_min_tick_tourheat/{getTimeFromTimestamp(ts1)}
```

most used bikes

In [31]:
```
%%time
def loadMultipleBikeData(bikeFilesName):
    bikeData = {}
    for bikeFileName in bikeFilesName:
        with open(bikeFileName, 'r') as file:
            content = json.load(file)
            bikes = content['countries'][0]['cities'][0]['places']
            timestamp = bikeFileName[-15:-5]
            bikeData[timestamp] = pd.DataFrame(bikes)

    return bikeData

bikeDataFileNames = loadBikeDataFileNames("data")
bikeDataFrameList = loadMultipleBikeData(bikeDataFileNames[::6])
```

```
CPU times: user 13.4 s, sys: 407 ms, total: 13.8 s
Wall time: 17 s
```

In [32]:
```
bikes = pd.concat(bikeDataFrameList.values(), keys=bikeDataFrameList.keys
```

In [33]:
```
def getFirstLevelIndexes(df):
    return df.index.get_level_values(0).unique()

def getFirstLevel(df, i):
    firstLevelIndexes = getFirstLevelIndexes(df)
    return df.loc[firstLevelIndexes[i]]

preprocessBikeData(getFirstLevel(bikes, 0)).head()
```

Out[33]:

	lat	lng	bike_numbers
0	52.504157	13.335328	18125
0	52.504157	13.335328	14860
0	52.504157	13.335328	13152
1	52.496986	13.291210	16199
2	52.498323	13.296157	19283

In [34]:
```
%%time
bikeTourList = {}

for i, (start, end) in enumerate(zip(getFirstLevelIndexes(bikes)[:-1], ge
    bikeTourStartBikes = preprocessBikeData(bikes.loc[start])
    bikeTourEndBikes = preprocessBikeData(bikes.loc[end])

    bikeTourList[start] = createBikeTour(bikeTourStartBikes, bikeTourEndB
```

```
CPU times: user 1min 50s, sys: 147 ms, total: 1min 50s
Wall time: 1min 52s
```

In [35]:
```
bikeTours = pd.concat(bikeTourList.values(), keys=bikeTourList.keys())
```

only count bike tours that are in berlin

In [36]:
```
%%time
filteredBikeTours = bikeTours[bikeTours.within(unary_union(kieze['geometr
```

```
CPU times: user 35.3 s, sys: 8.76 ms, total: 35.3 s
Wall time: 35.9 s
```

In [37]:
```
len(bikeTours), len(filteredBikeTours)
```

Out[37]:
```
(23083, 22974)
```

amount of tours

In [38]:
```
filteredBikeTours.groupby('bike_numbers')['length'].count().describe()
```

Out[38]:
```
count    3759.000000
mean        6.111732
std         3.961868
min         1.000000
25%         3.000000
50%         5.000000
75%         9.000000
max        35.000000
Name: length, dtype: float64
```

bike tours length

In [39]:
```
# per one tour
filteredBikeTours['length'].describe()
```

Out[39]:
```
count    22974.000000
mean      2052.653278
std       1473.353065
min        100.032328
25%        979.949659
50%       1746.904628
75%       2805.159812
max      13965.488980
Name: length, dtype: float64
```

In [40]:
```
# sum of all tours for each bike
filteredBikeTours.groupby('bike_numbers')['length'].sum().describe()
```

Out[40]:
```
count     3759.000000
mean     12545.266404
std       8985.855832
min        100.745455
25%       5481.430975
50%      10900.799483
75%      17879.115371
max      58961.029127
Name: length, dtype: float64
```

In [41]:
```
# per bike per tour
(filteredBikeTours.groupby('bike_numbers')['length'].sum() / filteredBike
```

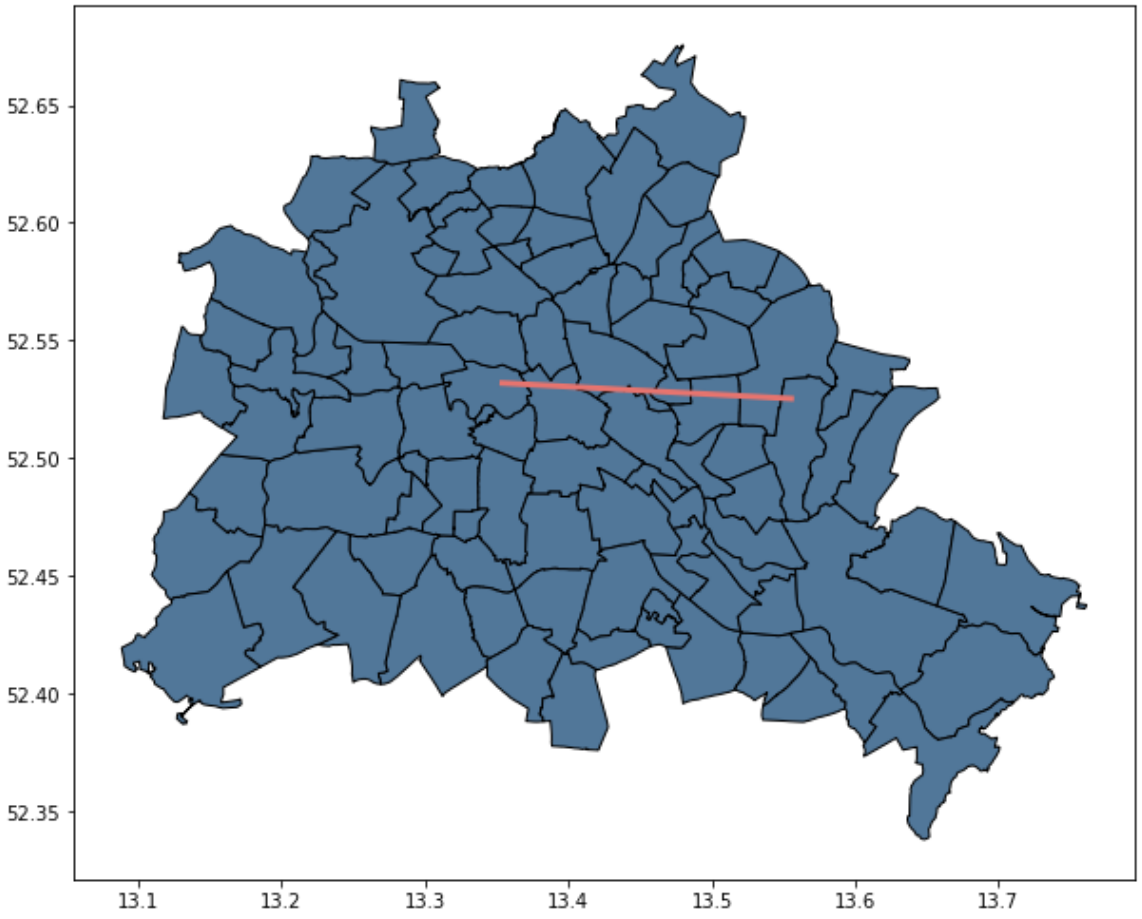
Out[41]:
```
count     3759.000000
mean      2058.045987
std        914.525207
min        100.745455
25%       1540.385388
50%       1975.048344
75%       2480.951615
max      11963.768986
Name: length, dtype: float64
```

In [44]:
```
# longest bike tour
filteredBikeTours['length'].sort_values(ascending=False).head(10)
```

Out[44]:
```
1661720402  1370    13965.488980
1661724002  1311    13965.488980
1661572802  1322    13072.416703
1661551202  721     13072.416703
1661562002  1433    13070.677653
1661551202  722     13070.677653
1661796003  2022    11963.768986
1662109203  2591    11526.226811
1662098403  3468    11102.246279
1662033603  3832    10884.623501
Name: length, dtype: float64
```

In [120…
```
ax = kieze.plot(color=colors[0], edgecolor='k', figsize=(10,10))
filteredBikeTours.loc['1661720402'].loc[[1370]].plot(color=colors[1], lin
```

Out[120]: <AxesSubplot:>



In [43]:
```
# bike with the most bike tours
filteredBikeTours.groupby('bike_numbers')['length'].count().sort_values(a
```

Out[43]:
```
bike_numbers
16430    35
19012    33
18165    25
14268    25
18152    23
Name: length, dtype: int64
```

In [137…
```
longestTour = filteredBikeTours[filteredBikeTours['bike_numbers'] == '164
longestTour.loc[:, 'tourIndex'] = np.arange(len(longestTour))

ax = kieze.plot(color=colors[0], edgecolor='k', figsize=(10,10))
longestTour.plot(ax=ax, column='tourIndex', cmap='plasma', linewidth=3)
```

```
/home/kang/uni/proceed-bpms-new-scalable-architecture-test/venv-iosl-proc
eed/lib/python3.10/site-packages/geopandas/geodataframe.py:1472: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

Out[137]: <AxesSubplot:>