OneBox by Allegro - Finding the greenest area

Task 1

Load dataset with *OneBox* parcel machines (`data/oneboxes.json`) and convert *lat/lon* into a geometry column. Assign it to `one_boxes_gdf` variable

In [2]:
```
data_path = '../../data/oneboxes.json'

### BEGIN SOLUTION
one_boxes_raw = pd.read_json(data_path)
one_boxes_gdf = gpd.GeoDataFrame(
    one_boxes_raw,
    geometry=gpd.GeoSeries.from_xy(one_boxes_raw["lon"], one_boxes_raw["lat"]),
    crs="EPSG:4326",
)
### END SOLUTION
```

In [2]:
```
data_path = '../../data/oneboxes.json'

### BEGIN SOLUTION
one_boxes_raw = pd.read_json(data_path)
one_boxes_gdf = gpd.GeoDataFrame(
    one_boxes_raw,
    geometry=gpd.GeoSeries.from_xy(one_boxes_raw["lon"], one_boxes_raw["lat"]),
    crs="EPSG:4326",
)
### END SOLUTION
```

In [3]:
```
one_boxes_gdf.head()
```

Out[3]:

	id	street	postalCode	city	lat	lon	geometry
0	AL003WPI	Główna 35	05-540	Ustanów	51.999732	21.038228	POINT (21.03823 51.99973)
1	AL068LU1	Janusza Korczaka 2	20-456	Lublin	51.209371	22.582271	POINT (22.58227 51.20937)
2	AL017GD1	Królowej Jadwigi 137 D	80-034	Gdańsk	54.325580	18.606516	POINT (18.60652 54.32558)
3	AL008ETM	Grota Roweckiego 16	97-200	Tomaszów Mazowiecki	51.539197	20.013981	POINT (20.01398 51.53920)
4	AL063LU1	Mełgiewska 7/9	20-209	Lublin	51.247014	22.608377	POINT (22.60838 51.24701)

In [4]:
```
len(one_boxes_gdf)
```

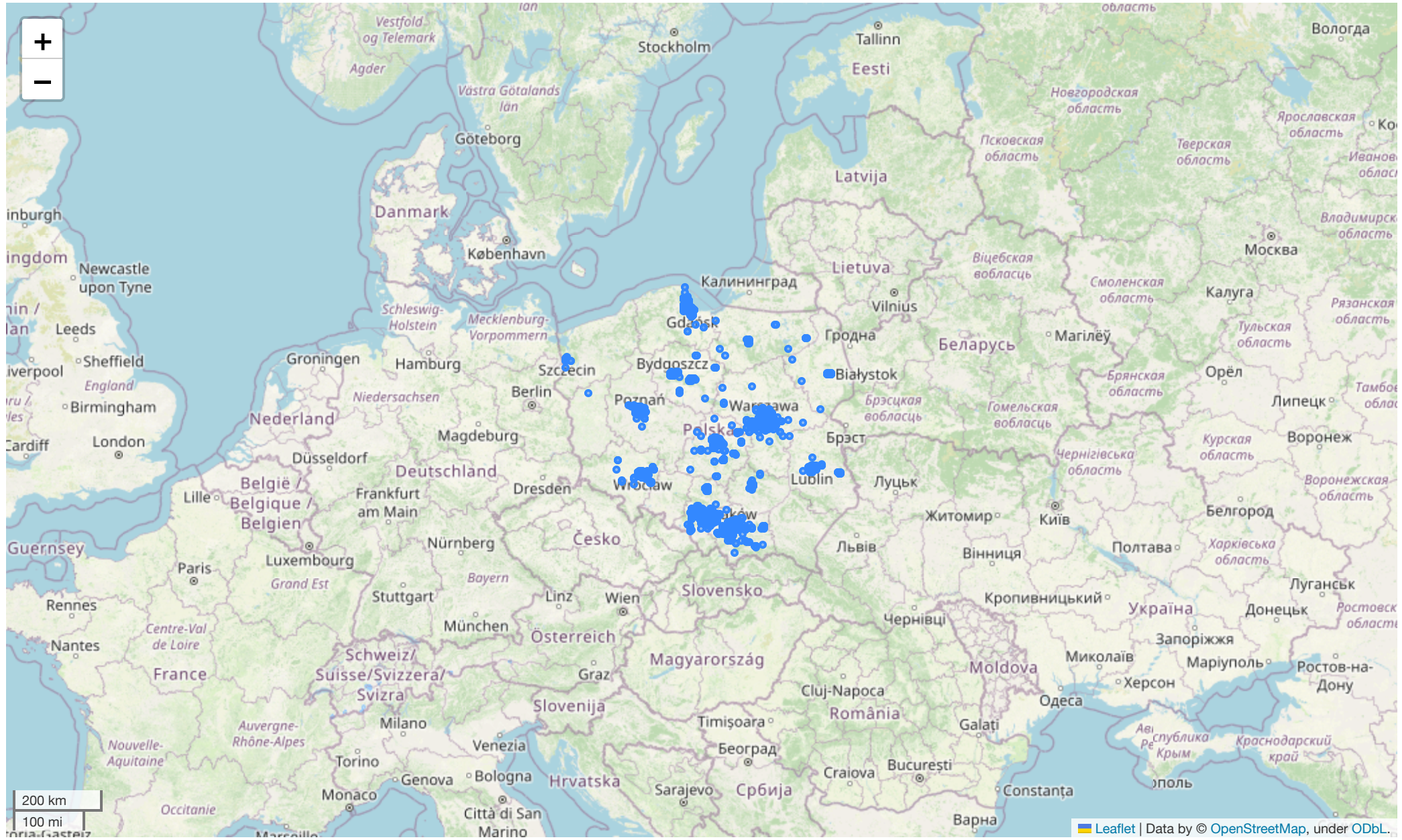
Out[4]: 3204

In [4]:
```
len(one_boxes_gdf)
```

Out[4]: 3204

In [5]:
```
one_boxes_gdf.explore()
```

Out[5]:



Visit our research group at `kraina.ai`

Task 2

Clip the GeoDataFrame to contain only *OneBoxes* placed in Warsaw.

Get the area of Warsaw from the OpenStreetMap. You can use `geocode_to_region_gdf` function from `srai.regionalizers` or `geocode_to_gdf` from `osmnx` library.

Clip the `one_boxes_gdf` using `.clip(other_geometry)` function from the GeoPandas. Assign it to `warsaw_one_boxes` variable.

In [6]:
```
osm_prompt = 'Warsaw, PL'
```

In [6]:
```
osm_prompt = 'Warsaw, PL'
```

In [7]:
```
### BEGIN SOLUTION
from srai.regionalizers import geocode_to_region_gdf

warsaw_region = geocode_to_region_gdf(osm_prompt)
warsaw_one_boxes = one_boxes_gdf.clip(warsaw_region)
### END SOLUTION
```

In [6]:
```
osm_prompt = 'Warsaw, PL'
```

In [7]:
```
### BEGIN SOLUTION
from srai.regionalizers import geocode_to_region_gdf

warsaw_region = geocode_to_region_gdf(osm_prompt)
warsaw_one_boxes = one_boxes_gdf.clip(warsaw_region)
### END SOLUTION
```

In [8]:
```
len(warsaw_one_boxes)
```

Out[8]: 431

Plotting a map of OneBoxes using PyDeck and custom icons

Plotting a map of OneBoxes using PyDeck and custom icons

In [22]:
```
import pydeck as pdk

icon_data = {
    "url": '../../data/onebox_circle.png',
    "width": 128,
    "height": 128,
    "anchorY": 128,
}

warsaw_one_boxes["icon_data"] = [icon_data for _ in warsaw_one_boxes.index]

view_state = pdk.data_utils.compute_view(warsaw_one_boxes[["lon", "lat"]], 0.1)

icon_layer = pdk.Layer(
    type="IconLayer",
    data=warsaw_one_boxes,
    get_icon="icon_data",
    get_size=16,
    size_scale=2,
    get_position=["lon", "lat"],
    pickable=True,
)

pdk.Deck(
    layers=[icon_layer],
    map_style="road",
    initial_view_state=view_state,
    tooltip={"text": "{id}: {street}"}
)
```

```
/Users/kacper.lesniara/Projects/Personal/srai-tutorial/venv/lib/python3.10/site-packages/g
```

Task 3

Create buffers with radius 500m around OneBoxes in Warsaw.

Use the `.buffer(radius)` function from GeoPandas. Remember not to use the standard **WGS84** CRS. Before the operation change the CRS using `to_crs` function and set it to **`EPSG:2180`** which is the Poland compatible CRS with units in meters. After buffering, reproject it back to WGS84 using `EPSG:4326` CRS.

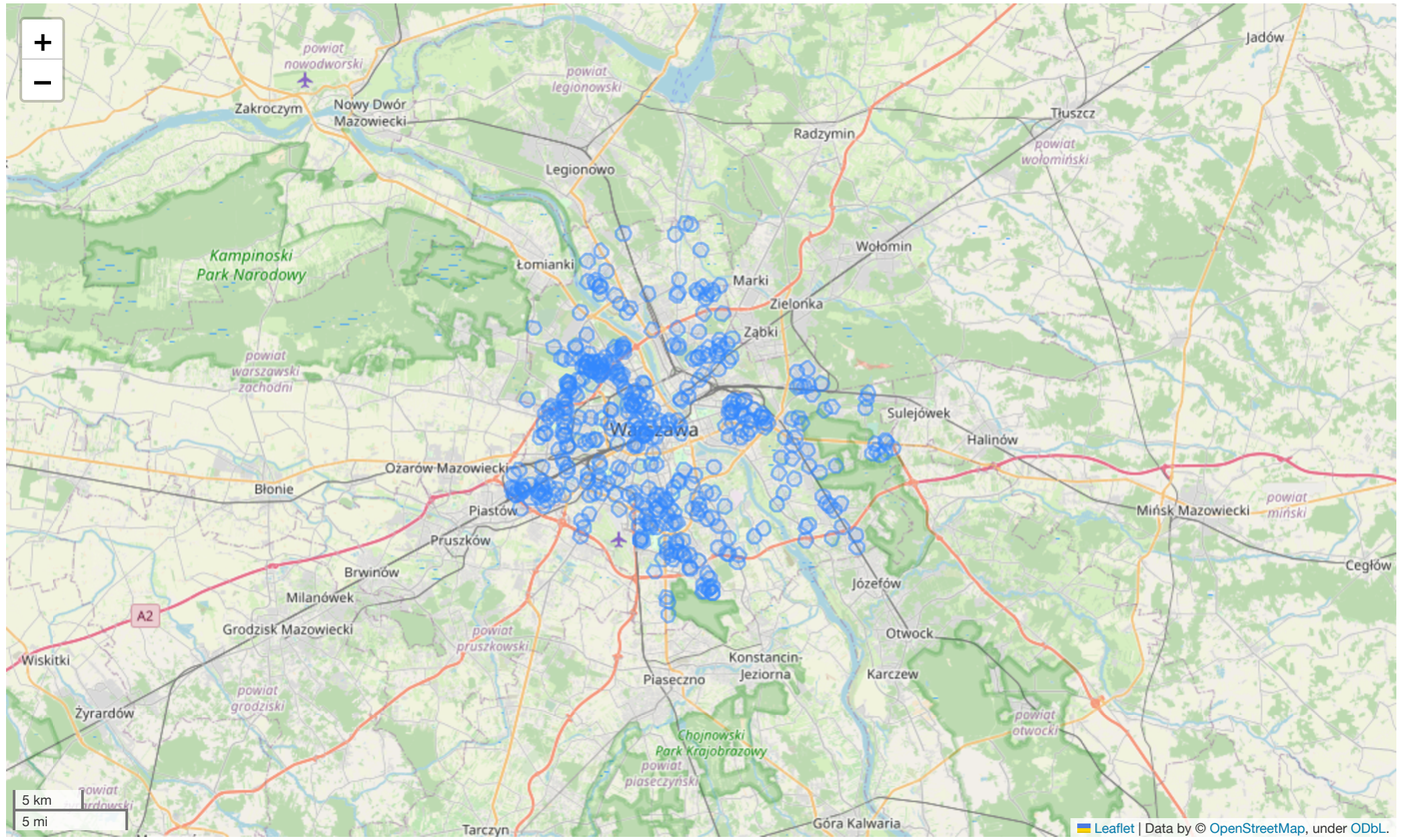
Create a new GeoDataFrame using data from `warsaw_one_boxes` variable and use created buffers as a `geometry` column.

In [10]:
```
### BEGIN SOLUTION
buffers_geometry = warsaw_one_boxes["geometry"].to_crs('EPSG:2180').buffer(500).to_crs('EPSG:
warsaw_one_boxes_with_buffers = gpd.GeoDataFrame(
    data=warsaw_one_boxes,
    geometry=buffers_geometry
)
### END SOLUTION
```

In [10]:
```
### BEGIN SOLUTION
buffers_geometry = warsaw_one_boxes["geometry"].to_crs('EPSG:2180').buffer(500).to_crs('EPSG:
warsaw_one_boxes_with_buffers = gpd.GeoDataFrame(
    data=warsaw_one_boxes,
    geometry=buffers_geometry
)
### END SOLUTION
```

In [23]:
```
warsaw_one_boxes_with_buffers.explore(style_kwds=dict(opacity=0.5, fillOpacity=0.1))
```

Out[23]:



Task 4

Load greenery data from OpenStreetMap using `OSMPbfLoader` from `srai.loaders.osm_loaders`. Initialize it and use function `.load()` with Warsaw geometry and OSM tags filter.

Filter is already provided and contains specific tags defining greenery objects.

Remove `Point` geometries from the collected GeoDataFrame. Use can use `geom_type` attribute of the GeoDataFrame.

Assign the GeoDataFrame to `greenery` variable.

In [12]:
```
greenery_osm_tags_filter = {
    "leisure": ["garden", "park"],
    "natural": ["wood", "scrub", "heath", "grassland"],
    "landuse": ["grass", "orchard", "flowerbed", "forest", "greenfield", "meadow"],
}
```

In [12]:
```
greenery_osm_tags_filter = {
    "leisure": ["garden", "park"],
    "natural": ["wood", "scrub", "heath", "grassland"],
    "landuse": ["grass", "orchard", "flowerbed", "forest", "greenfield", "meadow"],
}
```

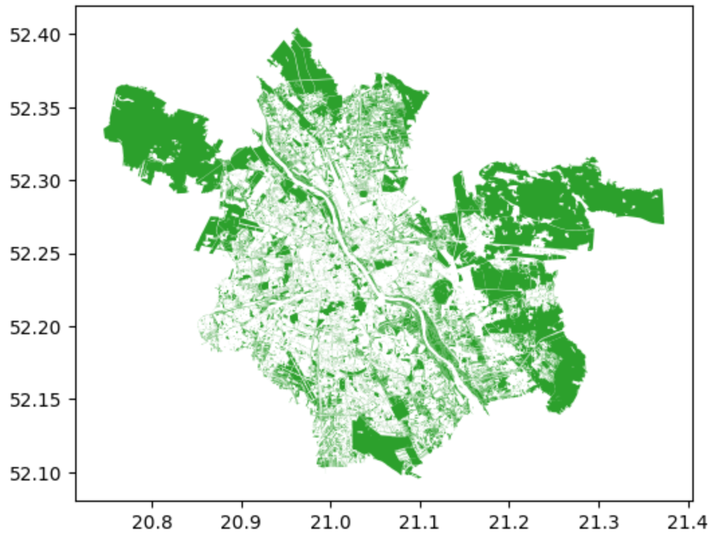
In [13]:
```
### BEGIN SOLUTION
from srai.loaders.osm_loaders import OSMPbfLoader

greenery = OSMPbfLoader().load(
    warsaw_region,
    greenery_osm_tags_filter
)
greenery = greenery[greenery.geom_type != 'Point']
### END SOLUTION
```

```
[Warsaw, Masovian Voivodeship, Poland] Counting pbf features: 5249564it [00:07, 687827.66i
t/s]
[Warsaw, Masovian Voivodeship, Poland] Parsing pbf file #1: 100%|████| 5249564/5249564 [0
0:20<00:00, 262390.10it/s]
```

In [14]:
```
greenery.plot(color="tab:green")
```

Out[14]: <Axes: >



Task 5

Intersect buffers around OneBoxes with downloaded greenery geometries. After intersection, group all intersected geometries into a single one using OneBox id.

To intersect all geometries, you can use `overlay` function from GeoPandas. It will intersectiond between each OneBox buffer and greenery object.

To join those intersections into a single geometry, use `dissolve` function from GeoPandas. It's an equivalent of `groupby` function from Pandas and automatically creates a `union` of all geometries in a group.

Assign the final GeoDataFrame to `warsaw_one_boxes_with_greenery` variable.

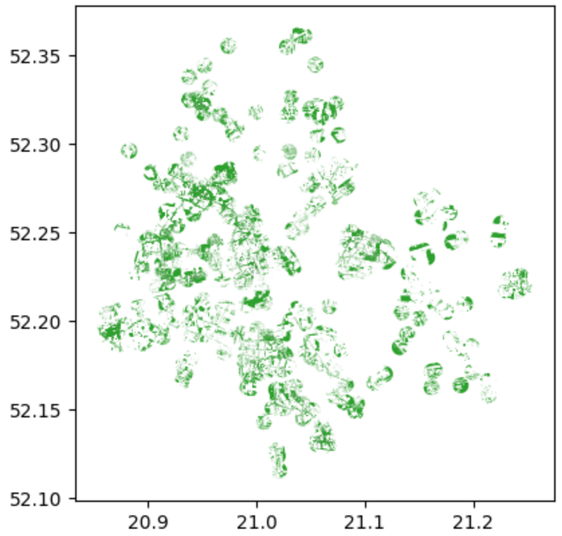
In [15]:
```
### BEGIN SOLUTION
warsaw_one_boxes_with_greenery = gpd.overlay(warsaw_one_boxes_with_buffers, greenery)
warsaw_one_boxes_with_greenery = warsaw_one_boxes_with_greenery.dissolve(by='id')
warsaw_one_boxes_with_greenery
### END SOLUTION
```

Out[15]:

id	geometry	street	postalCode	city	lat	lon	icon_data	landuse	lei
AL001WA1	MULTIPOLYGON (((20.97187 52.30431, 20.97213 52...	Piechocka 3	03-197	Warszawa	52.305324	20.979004	{'url': '../../data/onebox_circle.png', 'width...	grass	ga
AL001WB1	MULTIPOLYGON (((20.89004 52.22683, 20.89002 52...	Lazurowa 69	01-314	Warszawa	52.229377	20.895071	{'url': '../../data/onebox_circle.png', 'width...	grass	pa
AL001WD1	MULTIPOLYGON (((20.96723 52.28030, 20.96729 52...	Klaudyny 28A	01-684	Warszawa	52.283053	20.972099	{'url': '../../data/onebox_circle.png', 'width...	grass	pa
AL001WE1	MULTIPOLYGON (((21.01955 52.19389, 21.01950 52...	Merliniego 4	02-511	Warszawa	52.195668	21.026253	{'url': '../../data/onebox_circle.png', 'width...	meadow	pa
AL001WF1	MULTIPOLYGON (((21.08220 52.23964, 21.08214 52...	Al. Stanów Zjednoczonych 72	04-036	Warszawa	52.243419	21.086133	{'url': '../../data/onebox_circle.png', 'width...	grass	pa
...	...	...	...	...	...	...	...	...	...
AL063WW1	MULTIPOLYGON (((20.89018 52.20427, 20.89119 52...	Pola Karolińskie 4	02-401	Warszawa	52.208122	20.891644	{'url': '../../data/onebox_circle.png', 'width...	grass	ga
AL064WW1	MULTIPOLYGON (((20.94682 52.19605, 20.94688 52...	Instalatorów 23	02-237	Warszawa	52.199867	20.950657	{'url': '../../data/onebox_circle.png', 'width...	grass	pa
AL065WW1	MULTIPOLYGON (((20.91083 52.20732, 20.91058 52...	Redaktorska 3	02-441	Warszawa	52.211349	20.914066	{'url': '../../data/onebox_circle.png', 'width...	grass	pa

In [16]:
```
warsaw_one_boxes_with_greenery.plot(color='tab:green')
```

Out[16]: <Axes: >



Task 6

Find the OneBox with most greenery area around it. Calculate it in a proper CRS (you can use `EPSG:2180` again). Plot this best OneBox greenery from the buffer on a map.

In [20]:
```
### BEGIN SOLUTION
warsaw_one_boxes_with_greenery['greenery_area'] = warsaw_one_boxes_with_greenery['geometry'].
warsaw_one_boxes_with_greenery = warsaw_one_boxes_with_greenery.sort_values(by='greenery_area
warsaw_one_boxes_with_greenery.head()

best_onebox_id = warsaw_one_boxes_with_greenery.index[0]
warsaw_one_boxes_with_greenery.loc[[best_onebox_id]].explore(color="green")
### END SOLUTION
```

Out[20]:

In [21]:
```
m = warsaw_one_boxes_with_buffers.merge(
    warsaw_one_boxes_with_greenery["greenery_area"].reset_index(), on="id"
).explore(
    "greenery_area",
    cmap="BuGn",
    tiles="CartoDB positron",
    style_kwds=dict(opacity=0.5, fillOpacity=0.1),
)

warsaw_one_boxes.explore(m=m, color='green')
```

Out[21]: