



**One Health**  
Student Conference  
USAMV București



# **SPATIAL ANALYSIS OF GREEN INFRASTRUCTURE IN CLUJ- NAPOCA AND ITS IMPLICATIONS FOR URBAN AIR QUALITY**

Students: Cristian Puie, Andrei Cîmpean  
Coordinators: Lect. dr. Iulia Coroian, Conf. Dr. Luisa Andronie, Prof. Dr. Ioana Pop

December 3-5, 2025, București



# Introduction

- This study investigates the distribution and surface area of green spaces in the city of Cluj-Napoca, aiming to assess their impact on air quality.
- Geospatial data were obtained through the Copernicus platform, including information on green areas, built infrastructure, the hydrographic network, and other relevant components of the urban environment.
- These datasets were integrated in AutoCad and processed in PyCharm, where spatial analyses were carried out to accurately delineate vegetated areas and evaluate their distribution across the city.



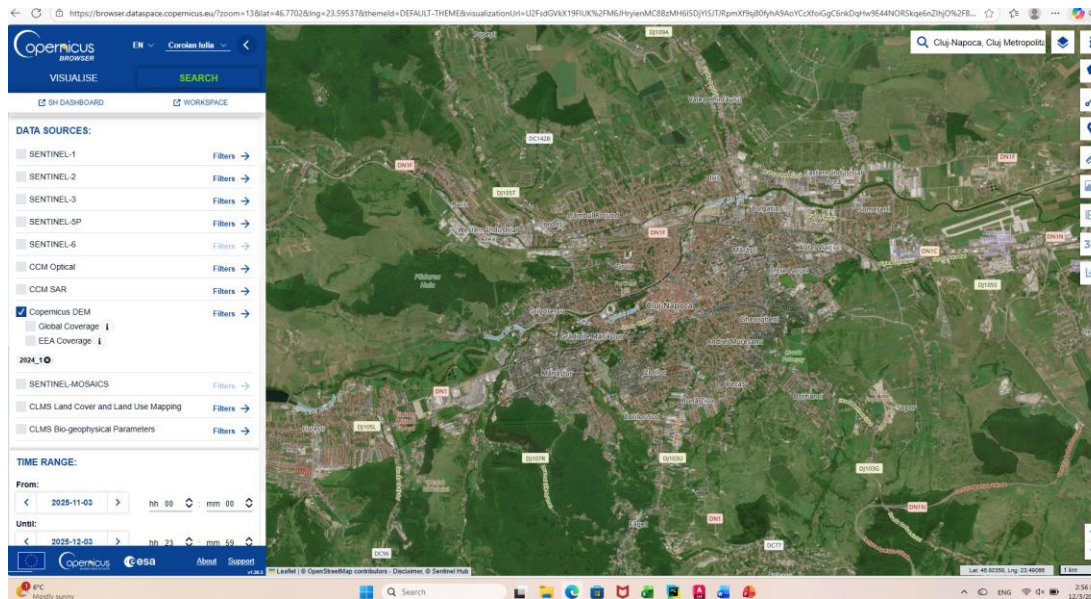
# Introduction

- Based on these analyses, the indicator of green space per capita was calculated, providing insight into the level of urban sustainability and the contribution of vegetation to improving air quality.
- Beyond the quantitative assessment, the study formulates evidence-based recommendations for the expansion and optimization of green spaces in areas identified as vegetation-deficient.
- These proposals aim to enhance ecological connectivity, improve air quality, and strengthen the resilience of the urban environment. The findings underscore the critical role of continuous monitoring and strategic planning of green infrastructure, highlighting the interdependence between urban development, ecosystem health, and population well-being.



# Materials and methods

1. Copernicus: Copernicus (satellite data) – used as a source of geospatial data for territory analysis



2. Autocad: used for managing and viewing CAD drawings



3. PyCharm: the main programming language, used for process automation and data analysis.





# Results and discussions

## Why nature is important for people?

- **Mental and physical health:** Urban green spaces reduce stress, anxiety, and depression, improve mood, and encourage physical activity.
- **Social benefits:** Nature in cities supports social interaction, community bonding, and engagement through activities like community gardening.
- **Climate regulation:** Urban vegetation cools cities, reduces flood risks, stores carbon, and stabilizes local microclimates.
- **Air quality and noise:** Trees filter pollutants and act as noise barriers, improving residents' living conditions.
- **Biodiversity:** Green infrastructure supports local species, pollinators, and essential ecosystem services.



# Results and discussions

## Recommendations for an ideal balance

- The World Health Organization (WHO) recommends — as a minimum level — approximately 9 m<sup>2</sup> of green space per person, preferably more.
- As an ideal or more ambitious target, some guidelines/analyses suggest around 50 m<sup>2</sup> per capita.
- In the European / urban-administrative context, a minimum of 26 m<sup>2</sup> of green space per inhabitant is often mentioned.

So, if a city “aligns with reference standards,” ideally it should provide 26–50 m<sup>2</sup> of green space per person; the minimum acceptable level (for health, access, well-being) is 9–10 m<sup>2</sup>.



# Results and discussions

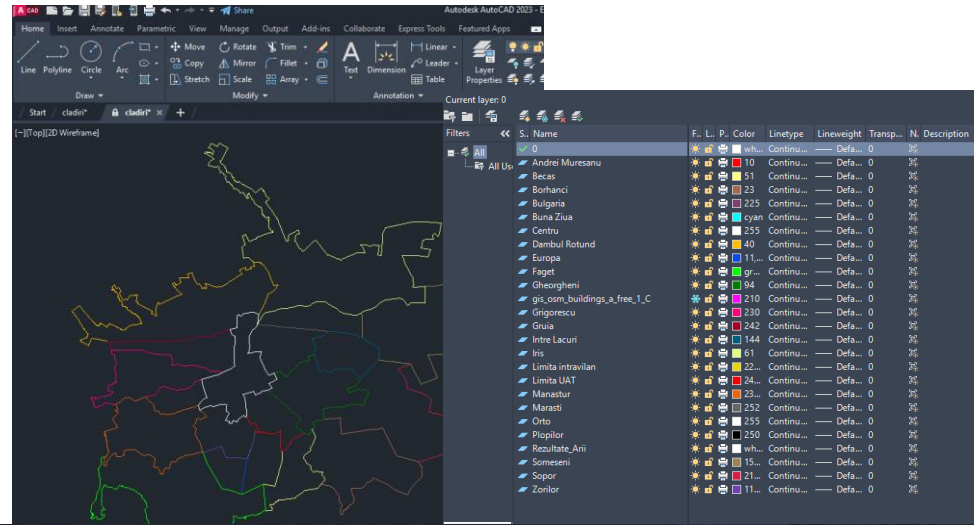
## Relationship between AutoCAD and Python

- **Automation and scripting:** Python can control AutoCAD's API to create and edit drawings, manage layers, and automate CAD tasks.
- **pyautocad:** Provides an easier interface to AutoCAD's COM/ActiveX system, enabling programmatic drawing of lines, circles, text, and more.
- **ezdxf:** Allows reading and writing DXF files directly in Python, enabling CAD manipulation without running AutoCAD.
- **Practical uses:** Automate repetitive tasks, extract or edit data, link CAD with external sources (Excel, databases), and build custom tools combining AutoCAD with Python.

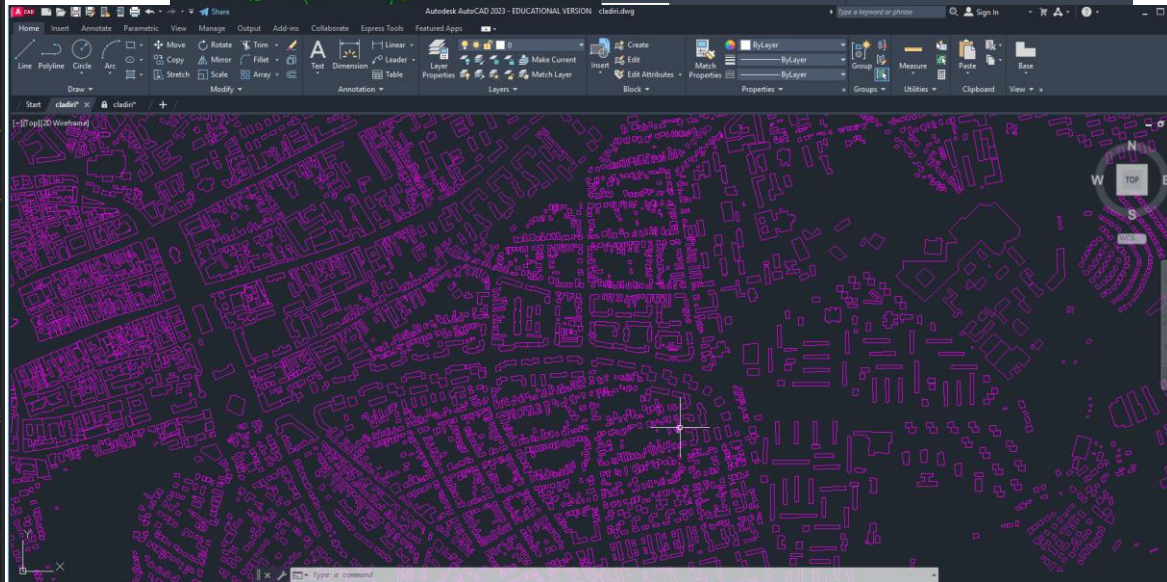


# Results and discussions

This script connects to AutoCAD, opens a base DWG file (buildings), and inserts a second DWG file (neighbourhoods) into it as a block at a specified insertion point.



```
cluj.py x
1 from pyautocad import Autocad, APoint
2
3 # --- CĂILE CĂTRE FISIERE ---
4 dwg_base = r"C:\usamv\martin_disertatie\MyProject8\cladiri.dwg"
5 dwg_insert = r"C:\usamv\martin_disertatie\Disertatie (1)\Disertatie\Ca
6
7
8 # --- PORNEȘTE / CONECTEAZĂ AUTOCAD ---
9 acad = Autocad(create_if_not_exists=True)
10 acad.app.Visible = True
11
12 # --- DESCHIDE DOAR FISIERUL PRINCIPAL ---
13 doc = acad.app.Documents.Open(dwg_base)
14 doc.Activate()
15
16 print("Fisierul principal deschis:", doc.Name)
17
18 # --- MODELSPACE AL DESENULUI ---
19 ms = doc.ModelSpace
20
21 # --- PUNCTUL DE INSERȚIE PENTRU AL DOILEA DWG ---
22 # APoint este helper din pyautocad pentru puncte 2D/3D
23 insert_point = APoint(x_or_seq: 0, y: 0, z: 0)
24
25 # --- INSERĂ AL DOILEA DWG CA BLOCK ÎN PRIMUL ---
26 # InsertBlock(InsertionPoint, "cale_sau_num_block", Xscale, Yscale, Zs
27 ms.InsertBlock(insert_point, dwg_insert, 1, 1, 1, 0)
28
29 print("Al doilea DWG a fost inserat în fisierul principal!")
30
31 # (opțional) salvezi direct fișierul de bază cu nume nou:
32 # doc.SaveAs(r"C:\usamv\martin_disertatie\MyProject8\cladiri_combinat.c
33
34
```





# Results and discussions

```
cluj.py x
34
35 # =====
36 # CALCUL ARIE FĂRĂ CLĂDIRI
37 # =====
38
39
40 import ezdxf
41 from shapely.geometry import Polygon
42 import pandas as pd
43
44 # =====
45 # CONFIGURARE CĂI FISIERE
46 # =====
47
48 path_cartiere = r"C:\usamv\dxfl\CartiereCluj-Napoca.dxf"
49 path_cladiri = r"C:\usamv\dxfl\cladiri.dxf"
50
51 # clădirile sunt pe acest layer (din diagnostic)
52 LAYER_CLADIRI = "gis_osm_buildings_a_free_1_C"
53
54
55 # =====
56 # HELPER: LWPOLYLINE -> Polygon
57 # =====
58
59 def lwpoly_to_polygon(lwpoly): 2 usages
60     """
61     Transformă o LWPOLYLINE în Polygon Shapely.
62     Presupune că polilinia este închisă (Closed = True).
63     """
64     points = [(x, y) for x, y, *_ in lwpoly.get_points("xy")]
65     if points[0] != points[-1]:
66         points.append(points[0]) # închidem poligonul, de siguranță
67     return Polygon(points)
```

This block of code prepares the system to:

1. Load the neighborhood boundaries from a DXF file.
2. Load the building footprints from another DXF file.
3. Convert AutoCAD polylines into real geometric polygons.
4. Prepare to compute the following:
  1. Total area of each neighborhood
  2. Area occupied by buildings
  3. Area of free land (neighborhood area minus building area)

This is part of a **GIS-style spatial analysis** implemented in Python using DXF files instead of shapefiles



## Results and discussions

```
70 # =====
71 C:\Users\coroi_v69cb3u\PyCharmMiscProject\cluj.py
72 # =====
73
74 doc_cart = ezdxf.readfile(path_cartiere)
75 msp_cart = doc_cart.modelspace()
76
77 cartiere_polygons = []
78
79 for e in msp_cart.query('LWPOLYLINE'):
80     # fiecare layer este numele unui cartier (Centru, Gruia, etc.)
81     nume_cartier = e.dxf.layer
82     poly = lwpoly_to_polygon(e)
83     cartiere_polygons.append({"nume": nume_cartier, "geom": poly})
84
85 print(f"Am găsit {len(cartiere_polygons)} cartiere.")
86 all_bounds = [c["geom"].bounds for c in cartiere_polygons] # (minx, m
87
88 minx = min(b[0] for b in all_bounds)
89 miny = min(b[1] for b in all_bounds)
90 maxx = max(b[2] for b in all_bounds)
91 maxy = max(b[3] for b in all_bounds)
92
93 print("Extrema desenului (cartiere):")
94 print("minx, miny, maxx, maxy =", minx, miny, maxx, maxy)
95
```

This part of the script processes the **DXF file containing the boundaries of the Cluj-Napoca districts** . It reads each polygon, converts it into a Shapely geometry, stores it, and computes the spatial extent of the entire dataset.



## Results and discussions

```
96 # =====
97 # CITIM CLĂDIRI
98 # =====
99
100 doc_clad = ezdxf.readfile(path_cladiri)
101 msp_clad = doc_clad.modelspace()
102
103 cladiri_polygons = []
104
105 for e in msp_clad.query('LWPOLYLINE'):
106     if e.dxf.layer == LAYER_CLADIRI:
107         poly = lwpoly_to_polygon(e)
108         cladiri_polygons.append(poly)
109
110 print(f"Am găsit {len(cladiri_polygons)} clădiri.")
111
```

This part of the script **reads the DXF file containing the building footprints**, filters out only the objects that belong to the *building layer*, and converts each building polyline into a geometric polygon that can be used for spatial calculations.



## Results and discussions

```
111
112
113 # =====
114 # CALCUL ARII PE CARTIER
115 # =====
116
117 results = []
118
119 for c in cartiere_polygons:
120     nume = c["nume"]
121     cartier_poly = c["geom"]
122
123     area_cartier = cartier_poly.area
124
125     # aria clădirilor care cad în interiorul cartierului (intersecția)
126     area_cladiri = 0.0
127     for b in cladiri_polygons:
128         if b.intersects(cartier_poly):
129             inter = b.intersection(cartier_poly)
130             area_cladiri += inter.area
131
132     area_fara_cladiri = area_cartier - area_cladiri
133
134     results.append({
135         "cartier": nume,
136         "area_cartier": area_cartier,
137         "area_cladiri": area_cladiri,
138         "area_fara_cladiri": area_fara_cladiri,
139     })
```

This section of the script computes, for each neighborhood, the total area, the built-up area, and the remaining free (unbuilt) area. For every neighborhood polygon, the code checks which building polygons intersect it, calculates the intersection area, and sums these values to obtain the total built-up surface. Finally, it subtracts the building area from the total neighborhood area and stores the results for later analysis.



## Results and discussions

```
# =====  
# | TABEL CU REZULTATE LĂNGĂ DESEN  
# =====  
  
doc.Activate()  
ms = doc.ModelSpace  
  
RESULT_LAYER = "Rezultate_Arii"  
  
# creare layer pentru rezultate  
layers = doc.Layers  
try:  
    result_layer = layers.Item(RESULT_LAYER)  
except Exception:  
    result_layer = layers.Add(RESULT_LAYER)  
  
result_layer.Color = 7 # alb
```

This part of the script prepares AutoCAD to display the results directly in the drawing. It activates the current document, accesses the ModelSpace, and creates (or retrieves) a dedicated layer called “Results\_Area”.



# Results and discussions

```
# =====  
# INSERAREA TABELULUI ÎN AUTOCAD (dreapta-sus)  
# =====  
  
offset_x = 0.1 * (maxx - minx) # 10% din lățime ca distanță  
insertion_x = maxx + offset_x # la dreapta hărții  
insertion_y = maxy # sus  
  
insertion_point = APoint(insertion_x, insertion_y, z: 0)  
  
# lățimea tabelului - poți crește sau micșora  
table_width = (maxx - minx) * 2 # 70% din lățimea orașului  
  
mtext = ms.AddMText(insertion_point, table_width, table_text)  
mtext.Layer = RESULT_LAYER  
mtext.Color = 256 # BYLAYER = alb  
  
# mărime text proporțională cu înălțimea orașului  
try:  
    mtext.Height = (maxy - miny) * 0.02  
except:  
    pass  
  
print("Tabelul COMPLET a fost inserat în DWG (dreapta-sus).")
```

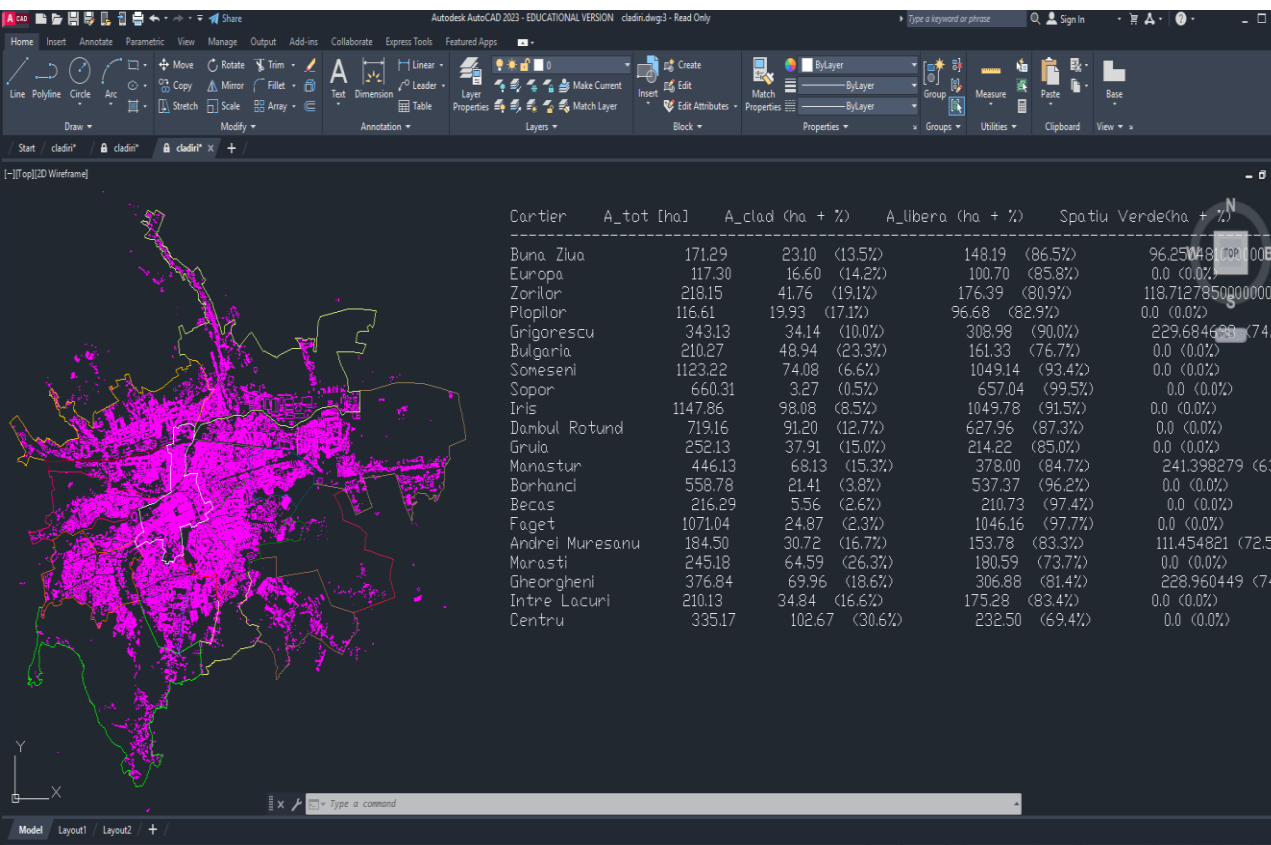
This code inserts the results table into the AutoCAD drawing, positioning it automatically in the upper-right corner of the map. It computes an offset to place the table outside the map boundary, calculates the table width based on the map size, and then creates an MTEXT object that contains the table content.



# Results and discussions

The image shows the complete result of the automated Python workflow, where the script analyzes the geometry of Cluj-Napoca's neighborhoods and generates a table summarizing several spatial indicators directly inside the AutoCAD drawing. Overall, the image demonstrates that the **entire automated workflow is functioning correctly:**

data is loaded, processed, analyzed, and the final statistics are inserted back into AutoCAD in a clean, readable format.





## Results and discussions

**Based on the total area of the Zorilor district (218.15 ha) and the surface classified as green space (118.71 ha, representing 54.4% of the territory), the following assessment of green-space availability per inhabitant can be made. The Zorilor district demonstrates a very high level of green-space provision, with approximately 47.5 m<sup>2</sup> of green area per resident.**

**This result indicates:**

- a healthy and well-balanced distribution of vegetated areas,
- a high level of environmental quality,
- strong potential for recreational activities and overall quality of life.

**Although the value does not fully reach the WHO recommendation of 50 m<sup>2</sup> per person, Zorilor is very close to this benchmark and clearly exceeds the conditions found in many other urban areas in Romania.**



## Results and discussions

**The Mănăștur district, with approximately 12.9 m<sup>2</sup> of green space per inhabitant, exceeds the WHO minimum threshold but remains below the urban-quality standards recommended by the European Union.**

From an urban-planning perspective:

- The existing green space provides basic functions (microclimate regulation, minimal recreational areas),
- but it is insufficient to meet modern quality-of-life standards
- and falls below the average of comparable European urban areas.

Given that Mănăștur is the city's largest and most densely populated district, it should be considered a priority for the expansion and better connectivity of green spaces.



## Conclusions and recommendations

- This study demonstrates how the integration of AutoCAD with Python—implemented through PyCharm—enables a powerful, automated workflow for spatial analysis and urban assessment.
- By combining CAD data with Python libraries, it becomes possible to extract building footprints, calculate areas for each neighborhood, identify green spaces, remove built-up surfaces, and automatically generate structured tables directly inside the DWG environment.
- This automated pipeline significantly reduces manual effort, eliminates human error, and ensures reproducible, transparent calculations across large urban datasets.
- The seamless connection between AutoCAD and Python transforms what would traditionally require hours of manual digitization into a fully scripted, scalable, and scientifically robust process—highlighting the essential role of computational tools in modern geospatial analysis and urban planning.



## Conclusions and recommendations

For neighborhoods with low green-space availability—such as Mănăștur and Bună Ziua—it is essential to implement targeted urban greening strategies that directly address existing gaps. Recommended interventions include:

- **Establishing tree-planting corridors** along major streets and pedestrian routes to create continuous ecological and shading networks.
- **Promoting rooftop and vertical gardens** on residential and commercial buildings to compensate for limited ground-level green areas.
- **Transforming vacant or underused parcels** into micro-parks or community green spaces that enhance local accessibility.
- **Expanding current parks or developing new pocket parks**, particularly in areas with high population density and limited recreational infrastructure.

# Thank you for your attention!

Cîmpeanu ANDREI-GABRIEL  
Puie CRISTIAN-IOAN

Mobil: 0770 234 875

0735 827 506

E-mail: [cristian-ioan.puie@student.usamvcluj.ro](mailto:cristian-ioan.puie@student.usamvcluj.ro)

[andrei-gabriel.cimpeanu@student.usamvcluj.ro](mailto:andrei-gabriel.cimpeanu@student.usamvcluj.ro)

Address: Calea Mănăştur, Nr. 3-5, 400372 Cluj-Napoca



December 3-5, 2025, Bucureşti



**One Health**  
Student Conference  
USAMV Bucureşti