Geo-Data and Geo-Data Management

Lecture 2: Vector Data — Geometry and CRS

Yingjing Huang

yingjing.huang@univie.ac.at

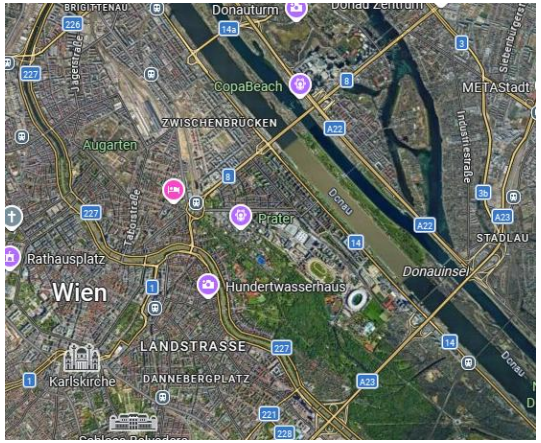
2026W

Recap: Geodata in the Wild

Week 1 gave us many ways to describe places and processes



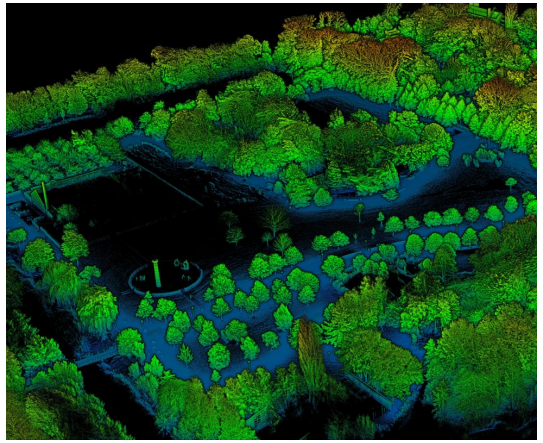
Digital map



Raster imagery



GPS trace



Point cloud

Source | **Information type** | **Representation** | **Storage & access**

Information type and representation are different choices; this week we focus on vector representation.

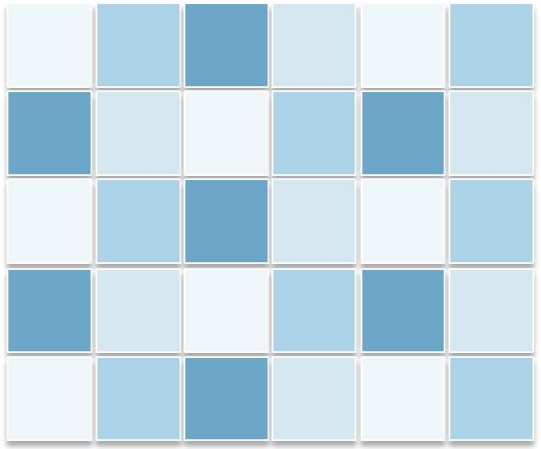
From Geodata to Vector and Raster

VECTOR



Points · Lines · Polygons

RASTER



Grids · Cells · Pixel values

WEEK 2: VECTOR · WEEK 3: INTEGRATION

WEEK 4: RASTER DATA

Vector and raster are two foundational spatial representations.

Today’s plan

1



Lecture

- Point, Line and Polygon geometry
- Shapely, pandas and GeoPandas
- Map projections, CRS and measurements

2

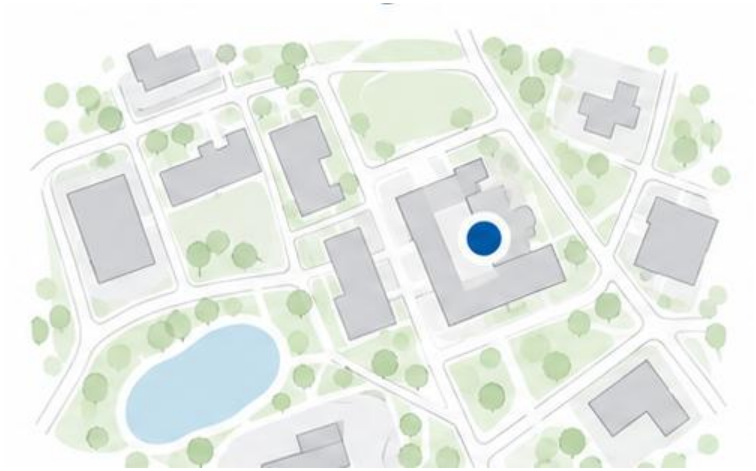
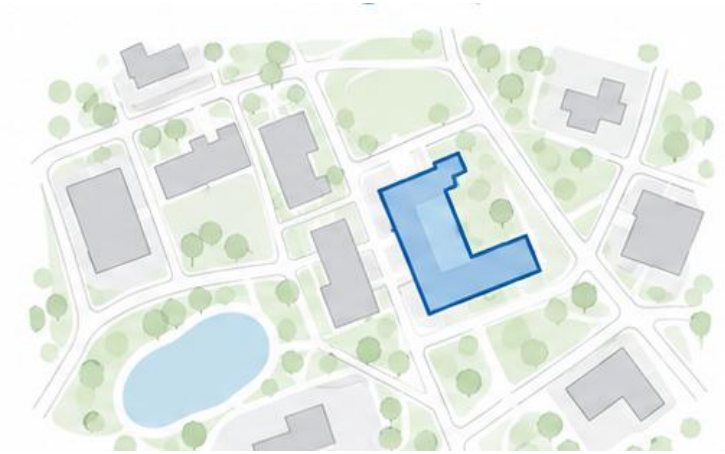


Lab

- Guided notebook: geometry and CRS
- Independent survey-site checkpoint

How Does Vector Data Represent the Real World?

One place can be represented differently depending on the spatial question.

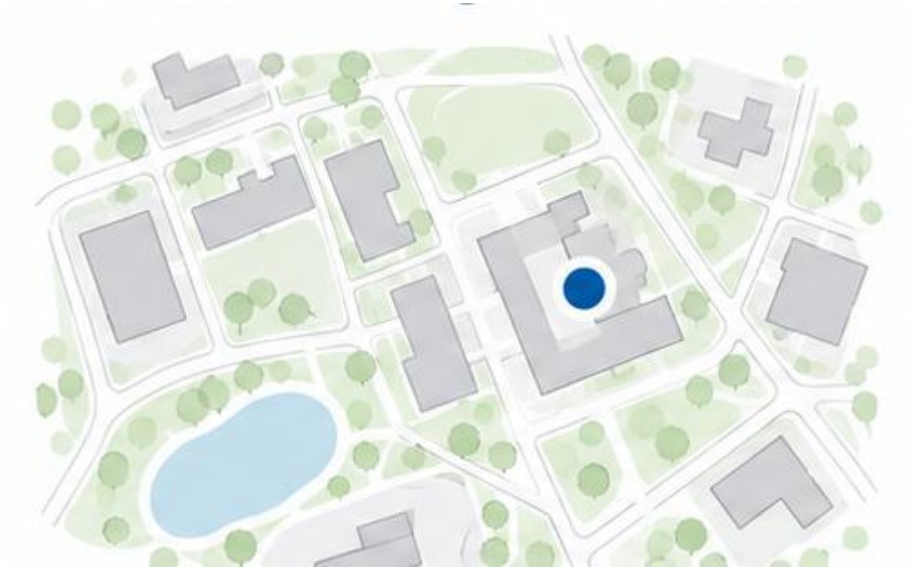
POINT	LINE	POLYGON
		
Where is the place?	Where does the path run?	What area is occupied?

Vector geometry is a modelling choice — we keep the spatial information we need.

One Building, Two Representations

Same building and scale; one representation answers where, the other how much area.

Building as a Point



Where is the building?

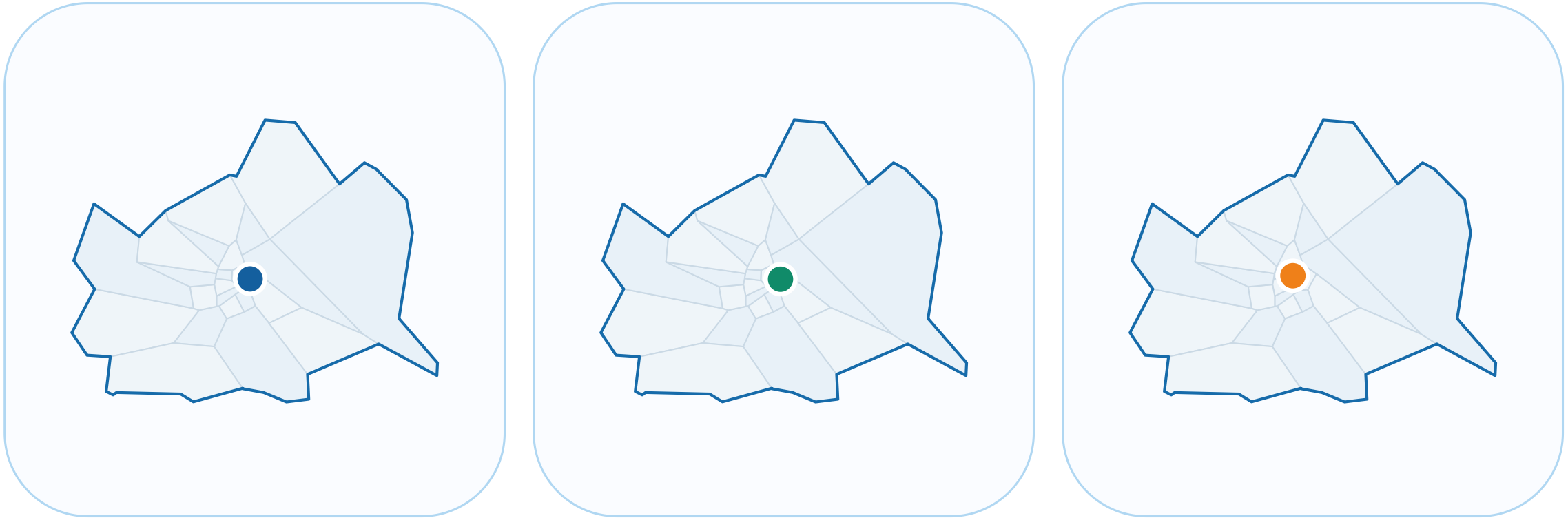
Building as a Polygon



What area does it occupy?

Point: One Coordinate Pair, Different Meanings

Point = one coordinate pair (x, y)



City centre

Stephansplatz

Geometric centroid

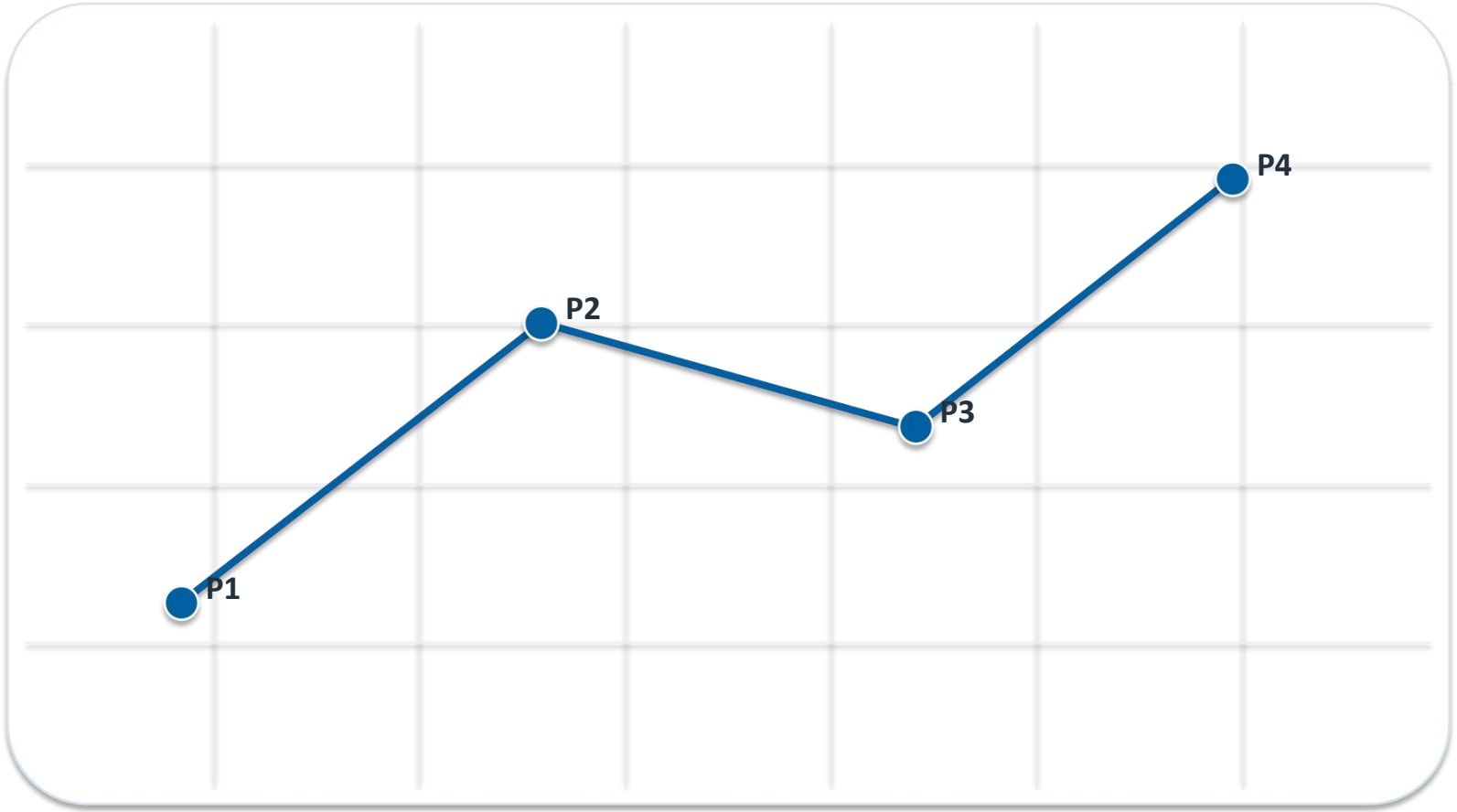
Vienna boundary centroid

Administrative seat

Wiener Rathaus

Boundary data: Statistik Austria (2021) · CC BY 4.0

Line: Connecting Ordered Vertices



Vertex

A coordinate along the path

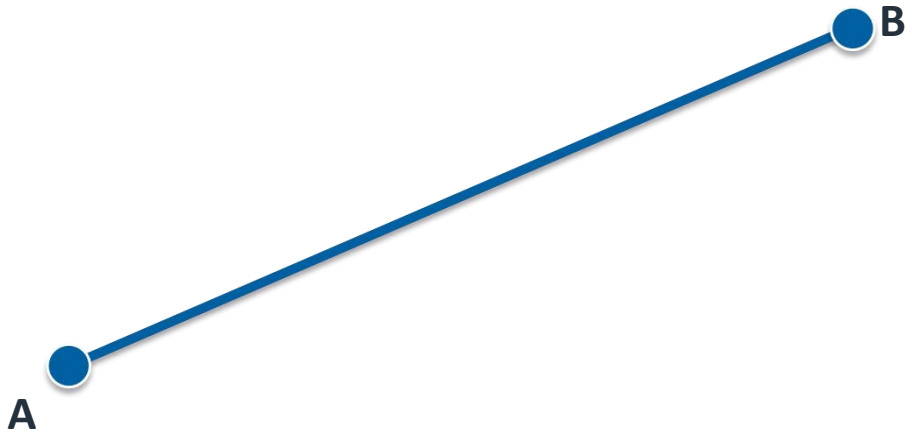
Segment

Connects two adjacent vertices

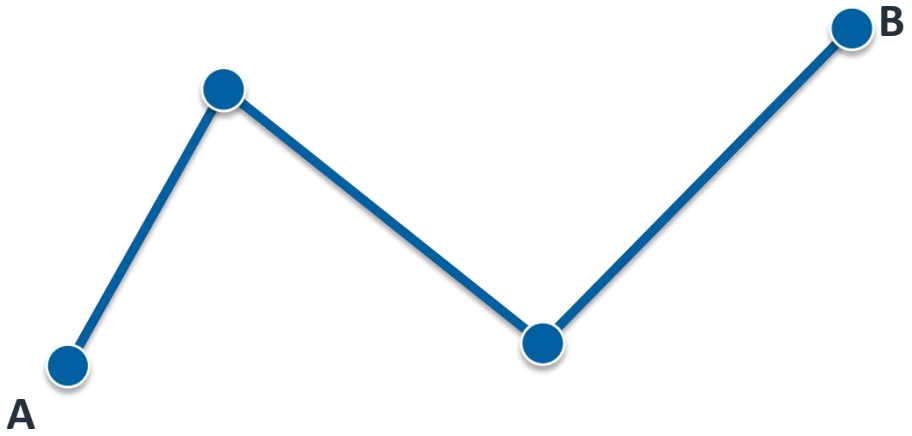
LineString follows ordered vertices (two or more); it represents a path, not road width.

Why Do Vertices Matter?

Straight segment



Intermediate vertices



Same endpoints ≠ same path

Why Represent a Road as a Line?



A road centreline represents its path; a road polygon represents the surface it occupies.

Polygon: A Boundary That Encloses an Area

A closed boundary separates the polygon interior from the exterior.



BOUNDARY

A closed ring of vertices

INTERIOR

Area inside the ring

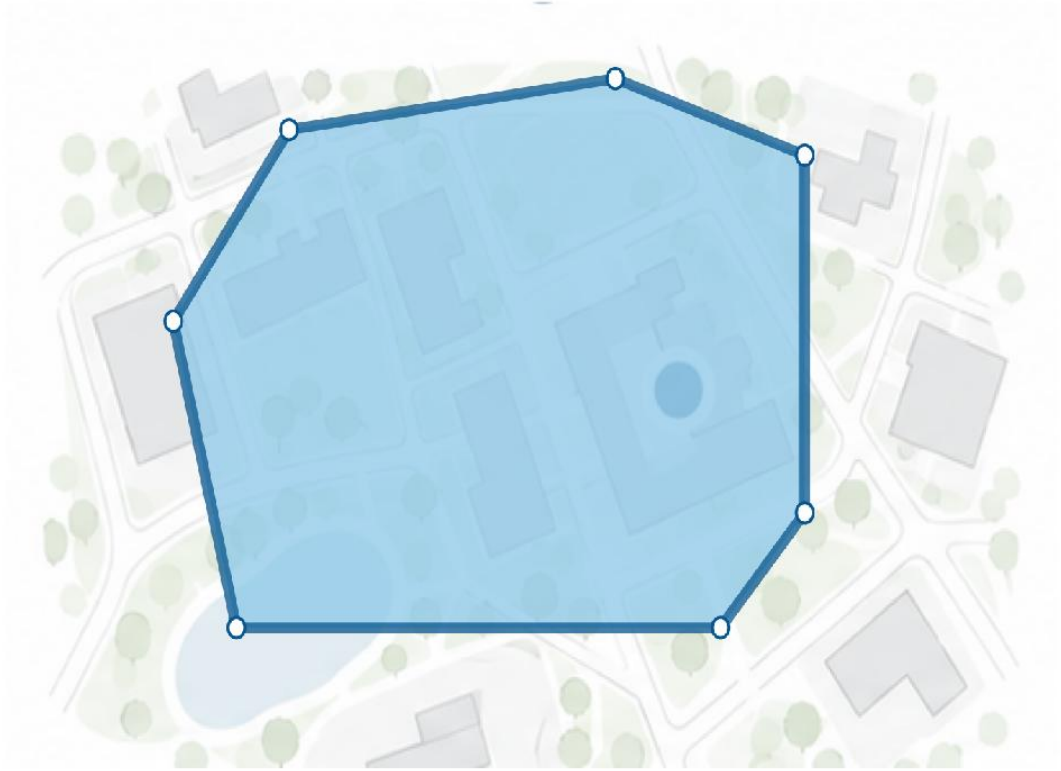
EXTERIOR

Space outside the polygon

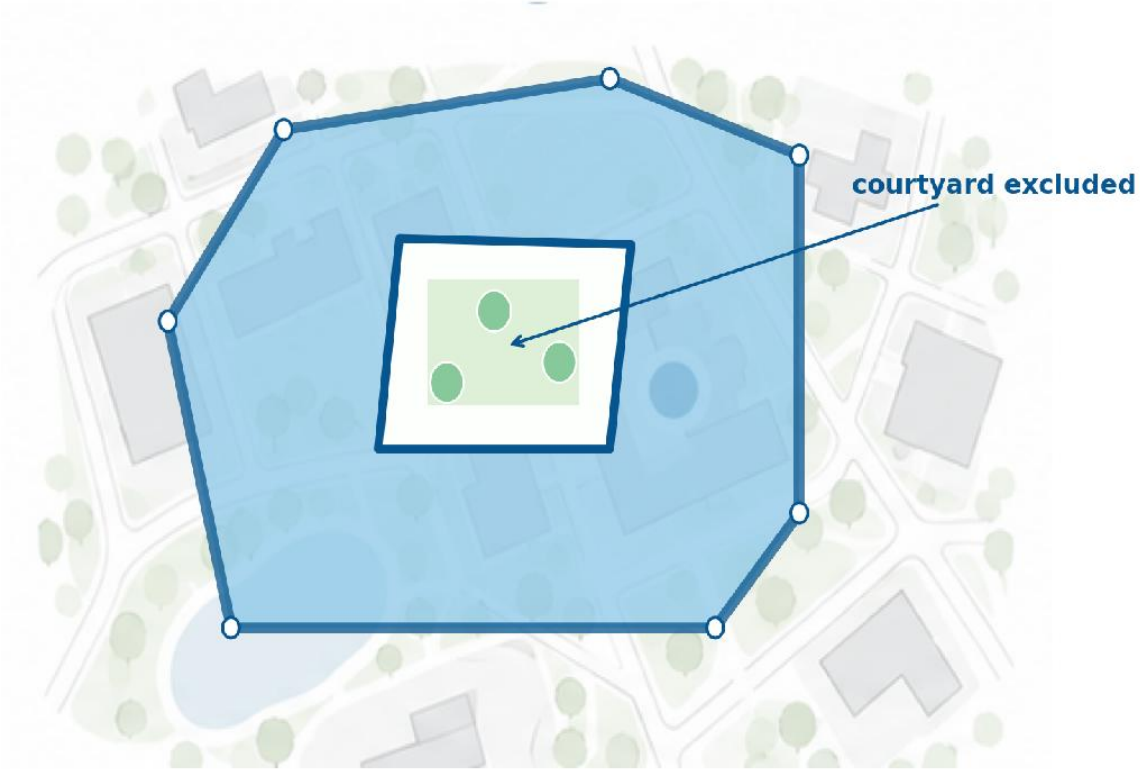
A footprint represents occupied 2D area — not the building height or floors.

Polygons Can Have Holes

Without a hole



With an interior ring



A hole is excluded from the polygon's area.

Geometry Choice Changes the Questions We Can Ask

POINT

Chosen location

Where is it?



LINE

Route shape

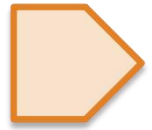
Where does it run?



POLYGON

Extent / inclusion

How much area?

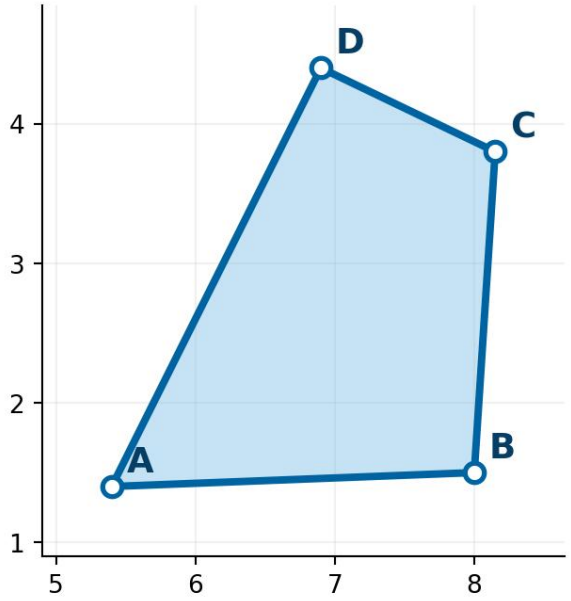


We have chosen the geometry. How can we work with it in Python?

How Can Python Work with a Geometry?

Coordinates become a geometry object — then Python can work with it.

POLYGON



CREATE THE GEOMETRY

```
from shapely.geometry import Polygon
vertices = [
    (5.4, 1.4), (8.0, 1.5),
    (8.15, 3.8), (6.9, 4.4)
]
geom = Polygon(vertices)
```

OPERATIONS

01 **Inspect**

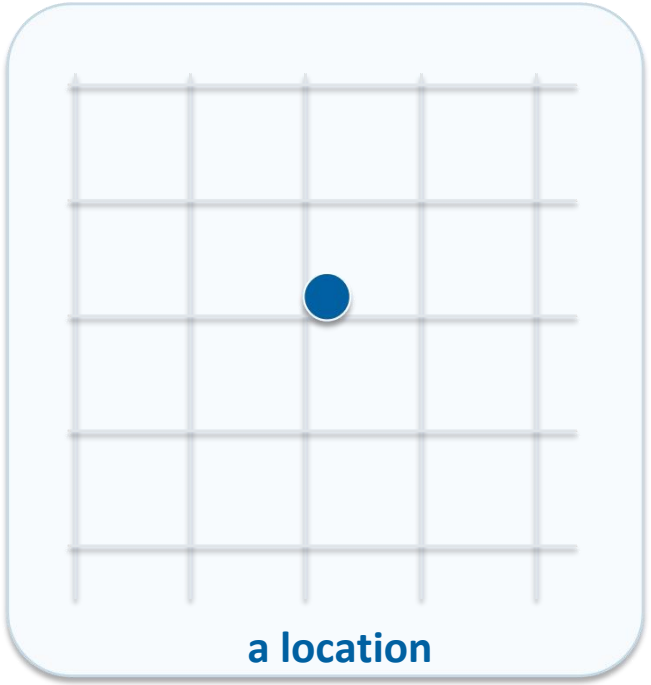
02 **Measure**

03 **Analyse**

Shapely provides geometry objects we can inspect, measure and analyse.

Shapely: Working with Geometry in Python

A geometry becomes a Python object that we can inspect and work with.



a location

SHAPELY POINT

```
p = Point(16.37, 48.21)
```

geometry object

Inspect

Read coordinates and type

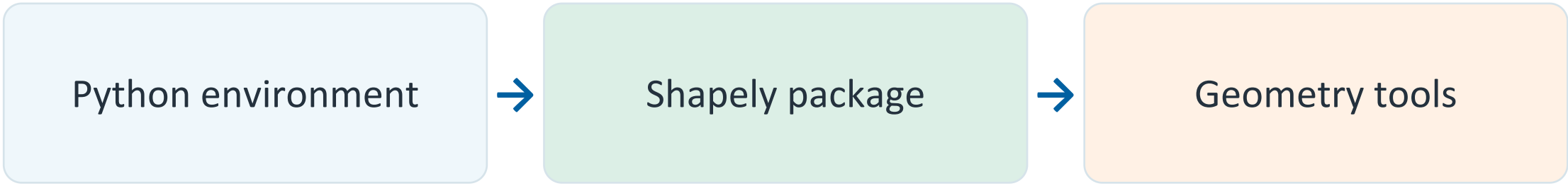
Analyse

Measure or compare shapes

A Shapely object stores geometry; the CRS must be tracked separately.

What Is a Python Package?

An environment contains installed packages; Shapely provides geometry tools.

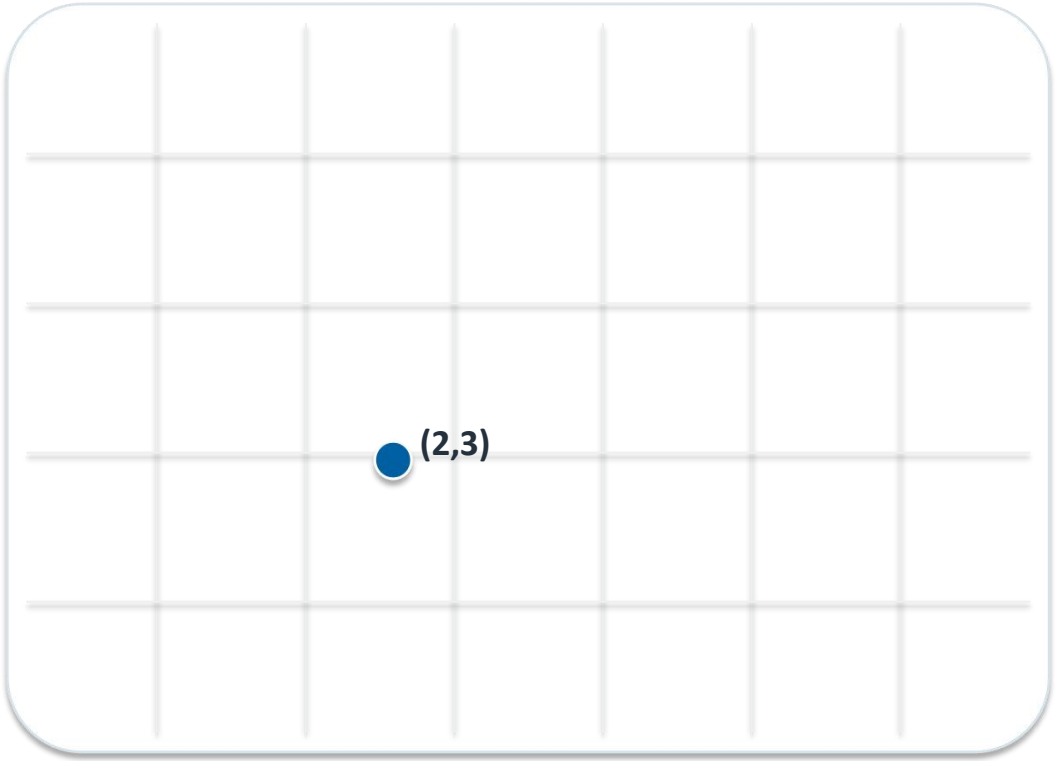



```
from shapely.geometry import Point
```

Where the tool comes from

What we want to use: Point

A Geometry Can Be an Object




```
p = Point(2, 3)
```

p refers to a specific object

has data · supports operations

The variable p names this specific Point instance; other Point objects may hold different coordinates.

An object is a specific thing with data and behaviours.

Class and Instance: Point vs p

The class is a type; the object stores its own coordinates; p names that same object.

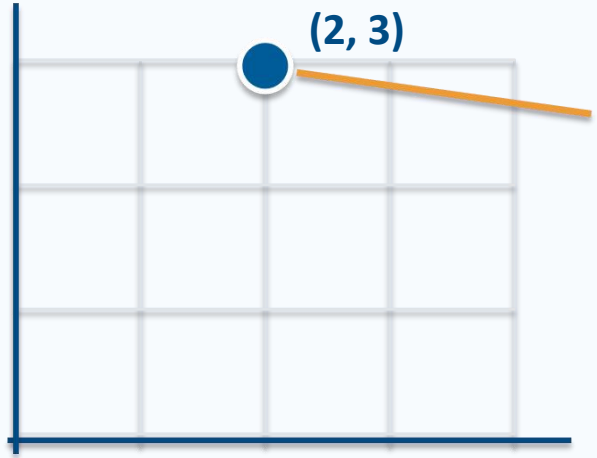
CLASS / TYPE

Point

Defines what Point objects are and can do.

creates

ONE INSTANCE / OBJECT

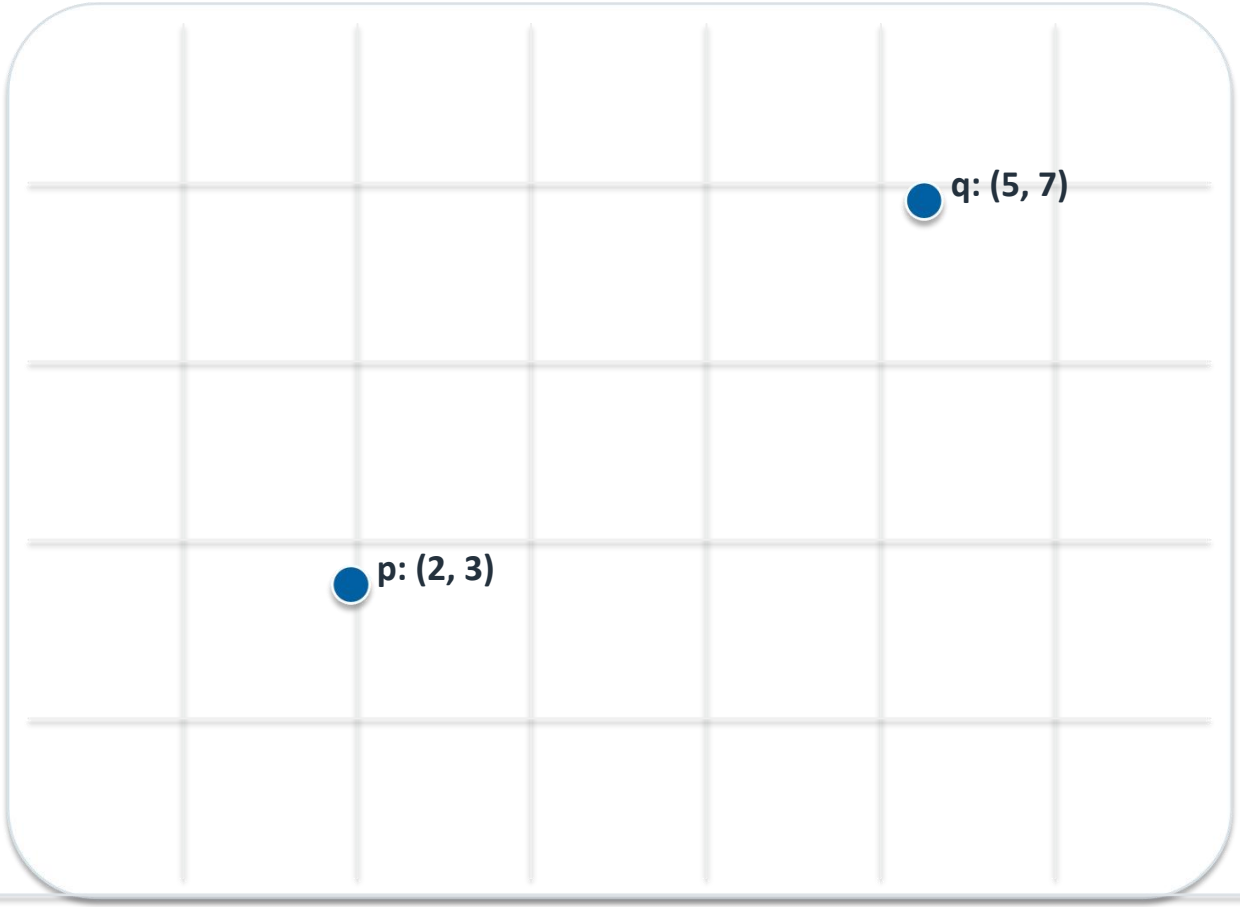



```
p = Point(2, 3)
```

p is the name referring to this same instance.

Remember: Point is a class; Point(2, 3) makes an instance; p is its variable name.

One Class, Multiple Instances

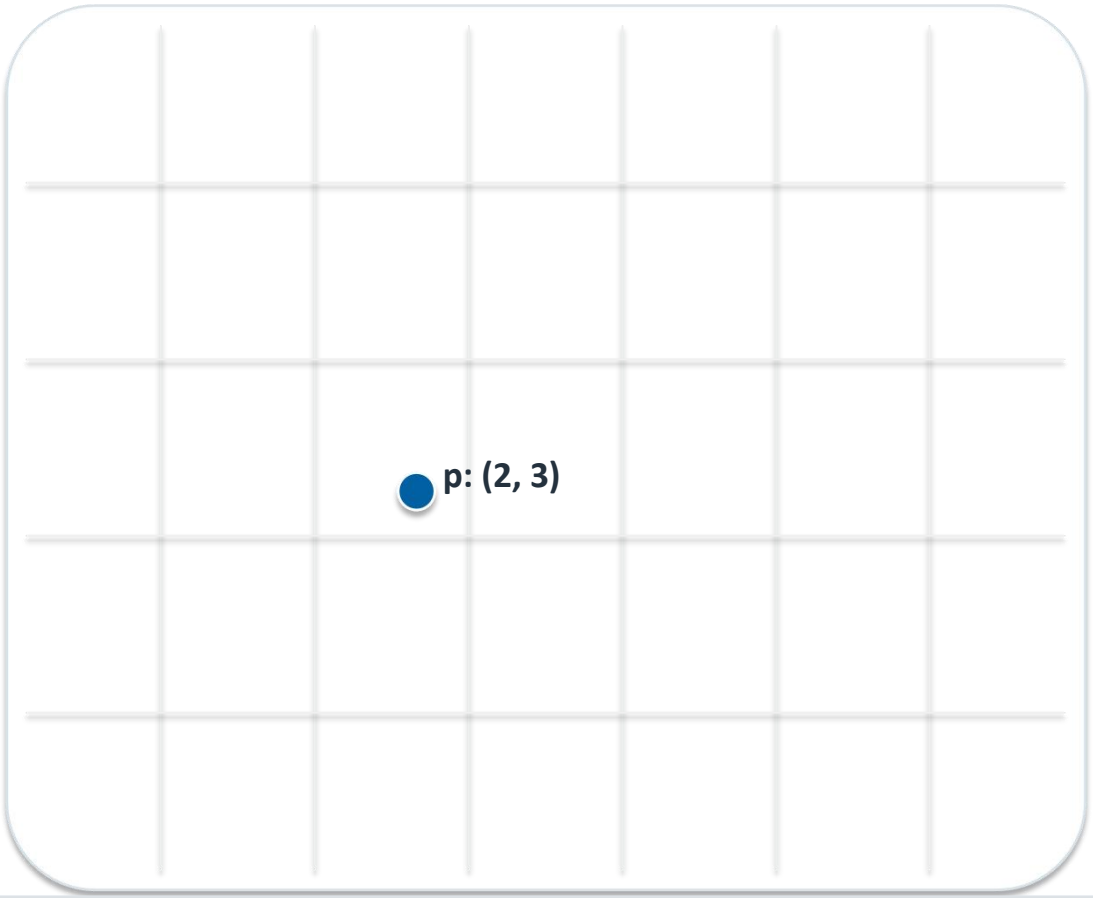



```
p = Point(2, 3)
q = Point(5, 7)
```

Both are Point instances

Same class, different data.

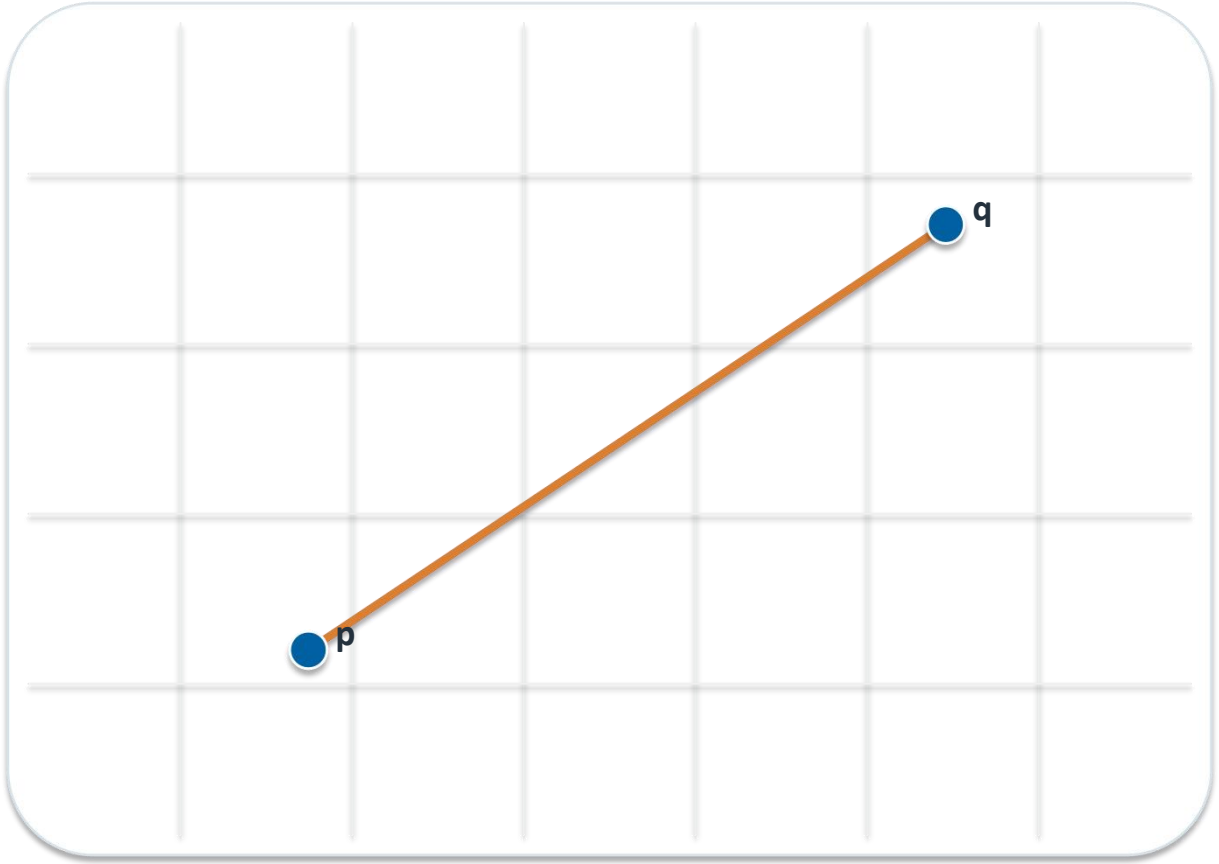
Attributes: What Does an Object Know?




```
p.x        → 2
p.y        → 3
p.geom_type → 'Point'
```

Attributes describe an object — no parentheses.

Methods: What Can an Object Do?




```
p.distance(q)  →  5
```

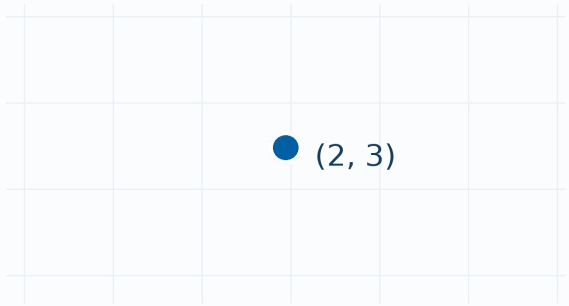
- p object performing the operation
- q argument / another Point
- 5 result in coordinate units

Method: calculate or do something · Attribute: read a property

Creating Point, LineString and Polygon Objects

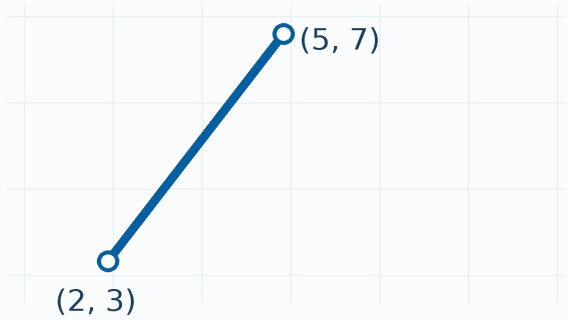
Three classes, three spatial representations.

Point



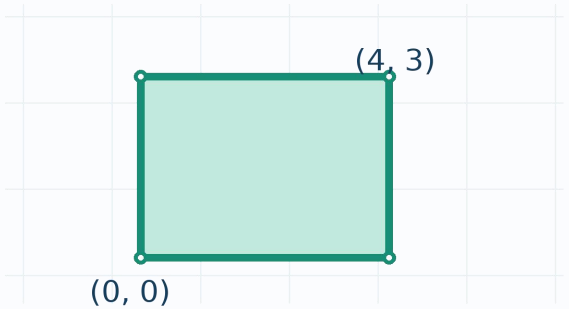

```
Point(2, 3)
```

LineString




```
LineString([
    (2, 3), (5, 7)])
```

Polygon




```
Polygon([
    (0, 0), (4, 0),
    (4, 3), (0, 3)])
```

```
from shapely.geometry import Point, LineString, Polygon
```

Line is the concept; LineString is the Shapely class. Polygon rings close automatically.

Shapely: Checking Geometry Validity

Both polygons were created from vertex lists before checking validity.

Simple polygon

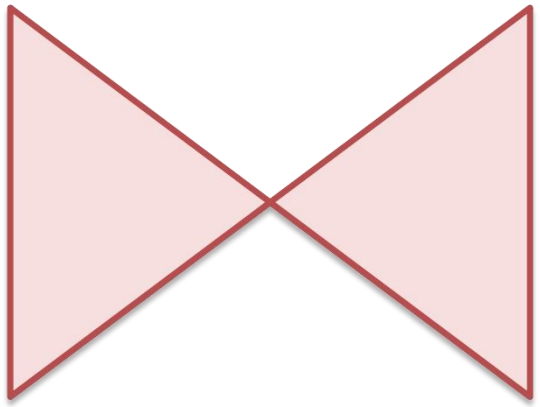
`valid_poly`



```
valid_poly.is_valid  →  True
```

Self-intersecting polygon

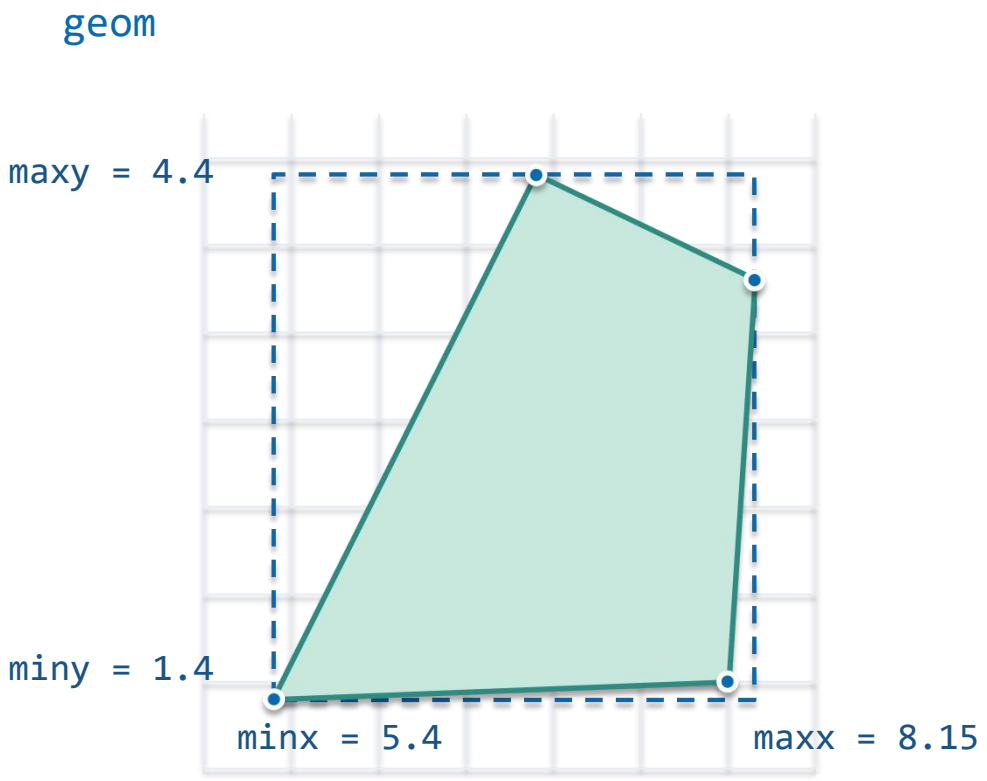
`bowtie_poly`



```
bowtie_poly.is_valid  →  False
```

is_valid checks polygon structure.

Shapely: Inspecting Geometry Bounds



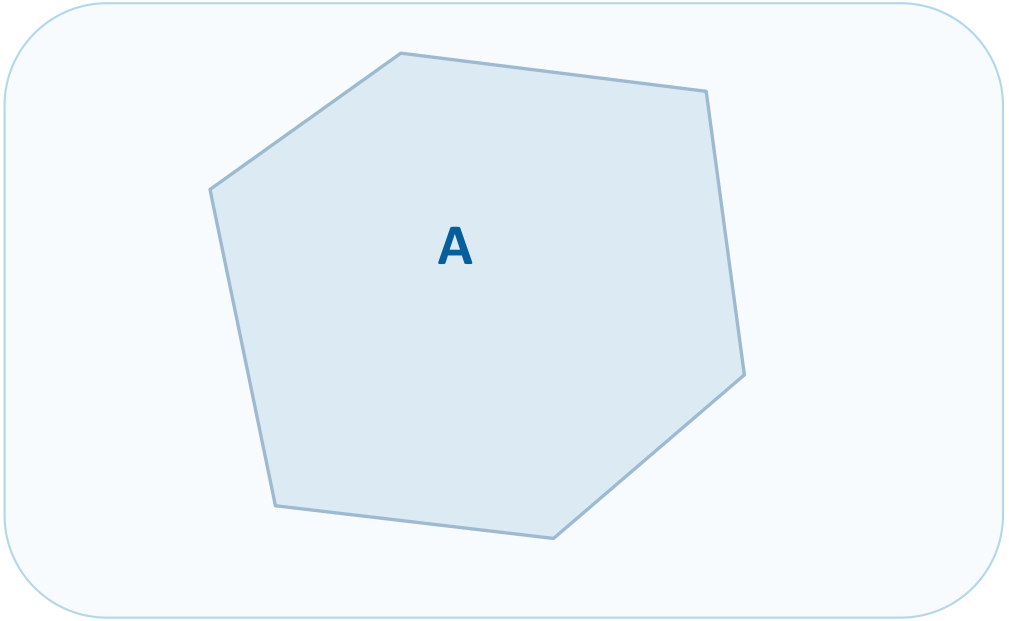

```
geom.bounds
→ (5.4, 1.4, 8.15, 4.4)
```

Axis-aligned minimum bounding rectangle

Bounds summarize a geometry's spatial extent.

Geometry Alone Is Rarely Enough

A polygon defines an area. A feature also carries descriptive attributes.



District A · polygon geometry

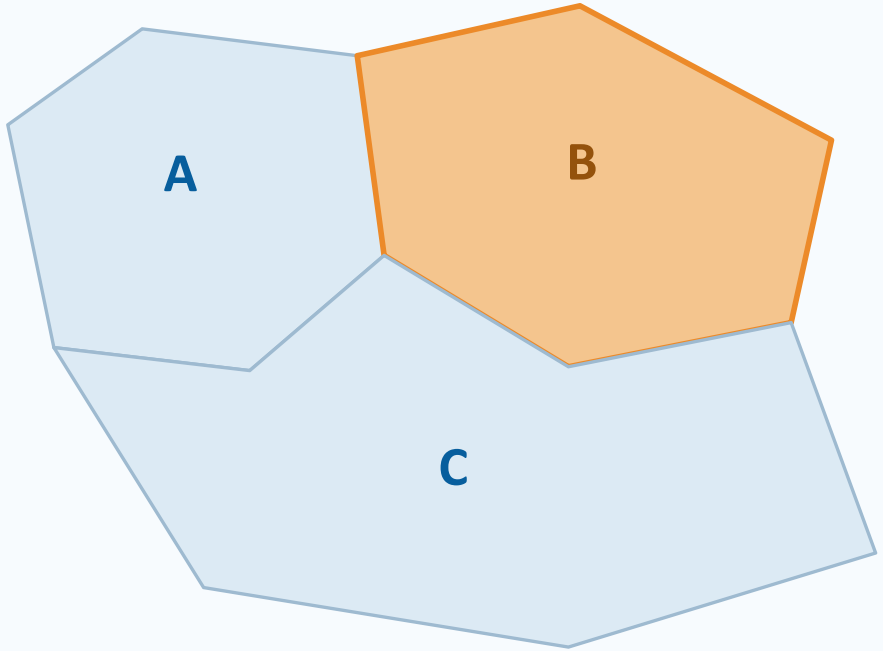
FEATURE ATTRIBUTES

name	A
ID	01
population	42,000

Geometry gives the boundary; attributes describe the district.

Feature Tables: Connecting Map and Attributes

MAP · DISTRICT FEATURES

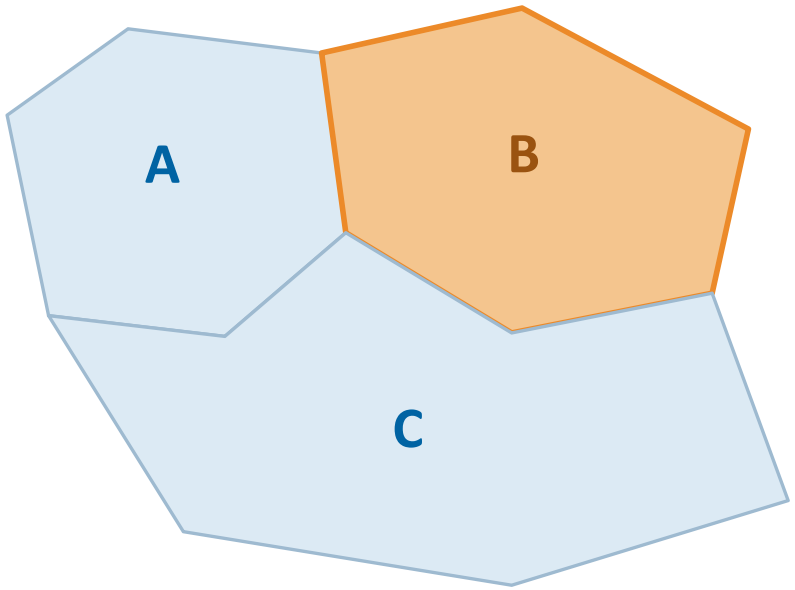


ATTRIBUTE TABLE

ID	Name	Population
01	A	42,000
02	**B**	**36,000**
03	C	51,000

Selected polygon B ↔ highlighted record B

One Row, One Feature



ID	Name	Population
01	A	42,000
02	**B**	**36,000**
03	C	51,000

The selected district on the map is the same feature as the highlighted record in the table.

Three district polygons → three feature records.

pandas: DataFrame for Attribute Records

name	category
Site A	museum
Site B	school
Site C	library

```
import pandas as pd

df = pd.DataFrame({
    "name": [
        "Site A", "Site B", "Site C"
    ],
    "category": [
        "museum", "school", "library"
    ]
})
```

A DataFrame stores records as rows and fields as columns.

A DataFrame Is Also an Object

A DataFrame stores rows and columns

name	category
Site A	museum
Site B	school
Site C	library

An object also exposes attributes and methods

ATTRIBUTE

```
df.columns
```

```
Index(['name', 'category'], dtype='object')
```

METHOD

```
df.head()
```

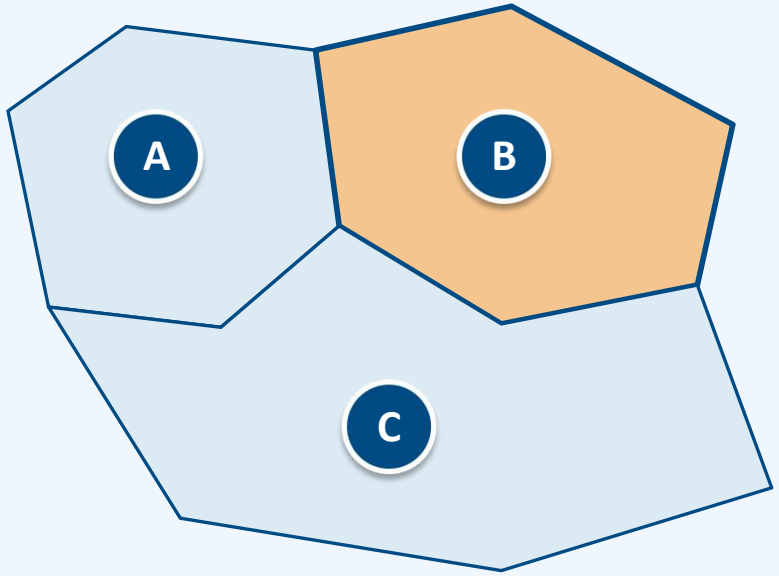
Up to five rows (all three shown here)

Same dot notation: read an attribute or call a method.

GeoPandas: A Table with Spatial Geometry

One Polygon feature corresponds to one record, with attributes, geometry and CRS.

MAP · DISTRICT POLYGONS



District polygons · one layer

GeoDataFrame · geometry = Polygon

CRS metadata

id	name	population	geometry
01	A	42,000	
02	B	36,000	
03	C	51,000	

Geometry is a column; the CRS applies to the layer.

Polygon B ↔ GeoDataFrame row B.

Building a GeoDataFrame from Data and Geometry

Combine two Point geometries, attributes and their known CRS.

```
import geopandas as gpd
from shapely.geometry import Point

p = Point(16.370, 48.210)
q = Point(16.380, 48.210)

gdf = gpd.GeoDataFrame(
    {"name": ["A", "B"]},
    geometry=[p, q],
    crs="EPSG:4326"
)
```

GeoDataFrame inputs

`geometry=[p, q]`

Point objects

`crs="EPSG:4326"`

WGS 84 lon/lat

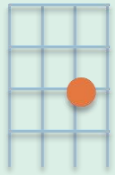
Attributes + geometry objects + known CRS → GeoDataFrame

Coordinates: Why x and y Are Not Always Longitude and Latitude

```
Point(x, y)
```



Geographic (lon, lat)
(16.370°, 48.210°)
EPSG:4326 · degrees



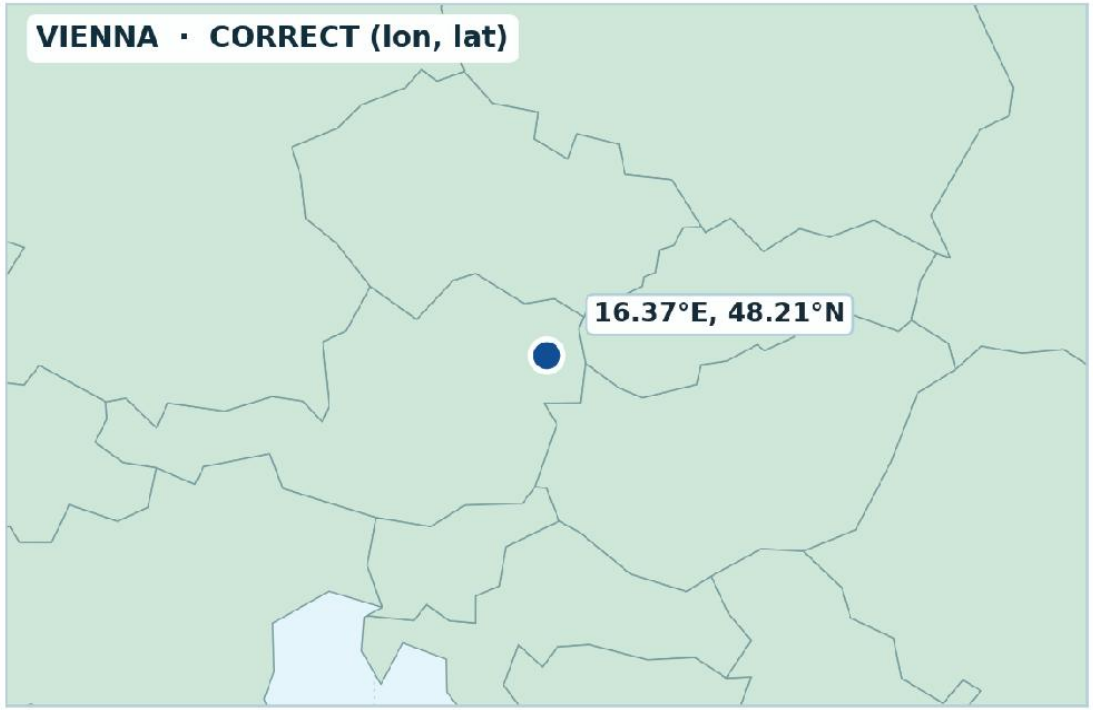
Projected (Easting, Northing)
(601,779 m, 5,340,549 m)
EPSG:32633 · metres

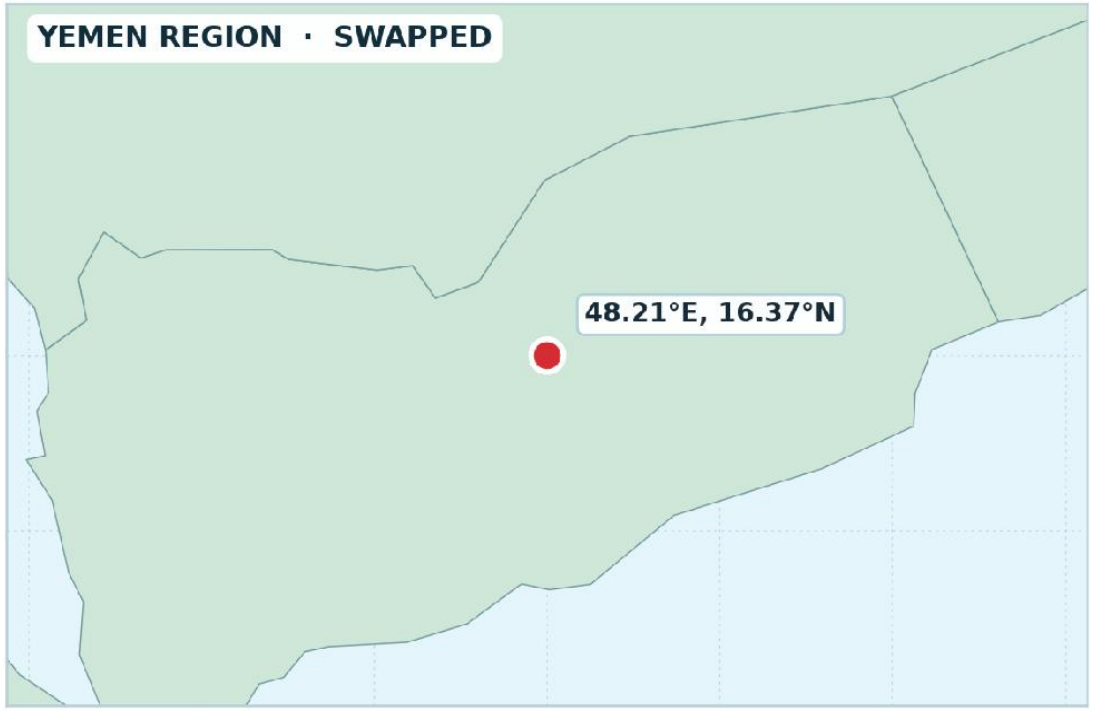
For geographic examples, Shapely/GeoPandas use x = longitude and y = latitude.
EPSG:4326 formally defines its axes in latitude, longitude order.

Know the CRS and coordinate order before interpreting x and y.

What Happens If We Swap Longitude and Latitude?

The code still runs if the coordinate order is wrong—but the location moves.





Point(16.37, 48.21) is near Vienna. Swapping x and y is not.

Coordinates Need a Reference System

Datum / reference frame

Anchors coordinates to an Earth reference frame (e.g. WGS 84).

Axes and order

Defines axis directions and order, such as longitude and latitude.

CRS



Example CRS IDs:
EPSG:4326 · EPSG:32633

Units

Angular coordinates use degrees; many projected coordinates use metres.

Projection (if projected)

Maps Earth-referenced positions to planar x/y using defined parameters.

A CRS explains coordinates; a map projection is only part of a projected CRS.

Geographic CRS



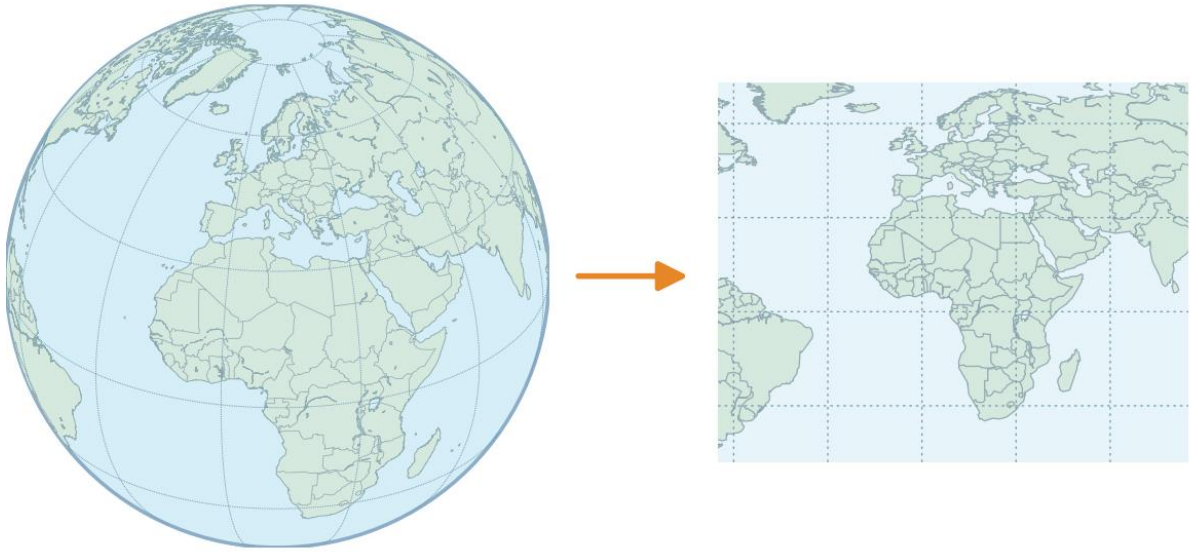
EPSG:4326
WGS 84 geographic CRS
Latitude and longitude
Angular units: degrees

Longitude measures east/west position; latitude measures north/south position. Both are angles.

Geographic CRS ≠ map projection. Longitude/latitude describe angular location.

Projected CRS: From the Curved Earth to a Flat Map

A projection mathematically maps locations on the Earth onto planar coordinates.



CURVED EARTH

FLAT COORDINATE GRID

GEOGRAPHIC CRS

longitude / latitude
angular units (degrees)

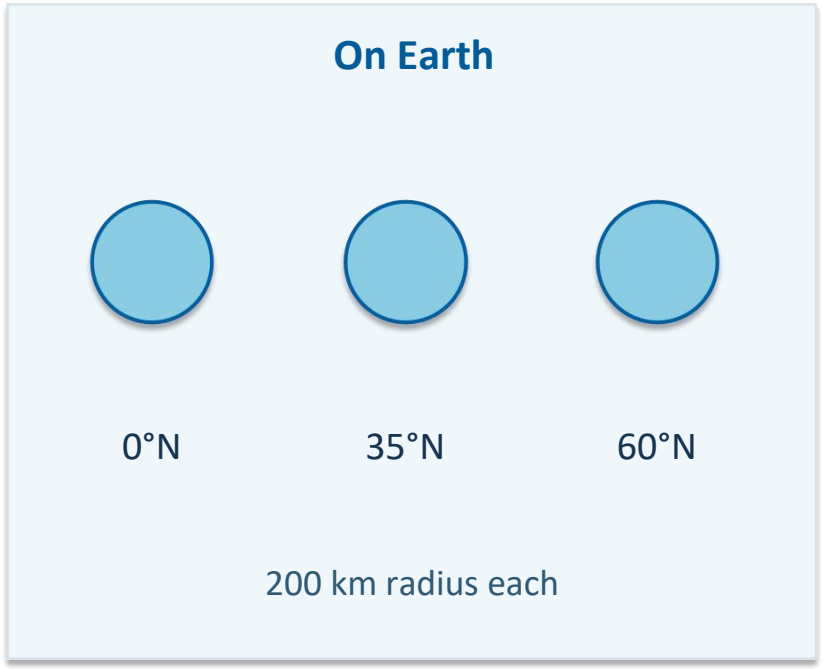
↓ PROJECTION

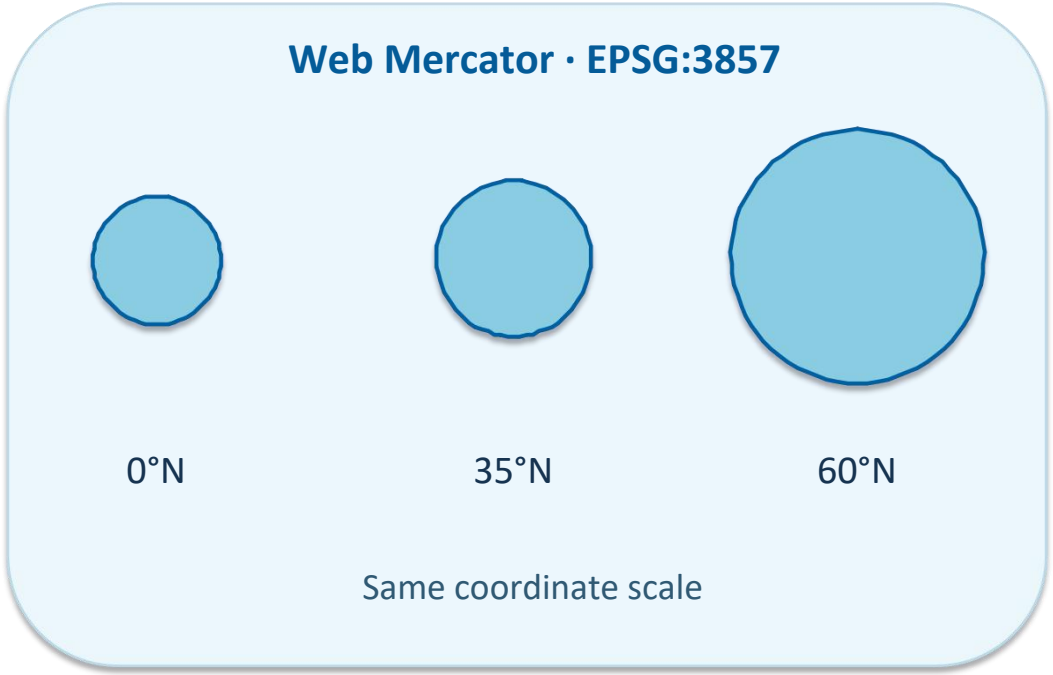
PROJECTED CRS
planar x / y; often metres

A projected CRS also specifies the reference frame, axes, units and projection parameters.

Why Does Flattening the Earth Cause Distortion?

Same ground radius at three latitudes



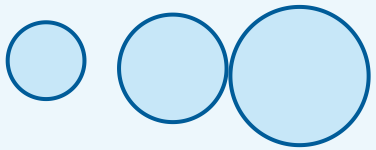


Mercator enlarges regions towards the poles.

What Can Different Projections Preserve?

A projection can prioritize one property, but the guarantee has a limited scope.

CONFORMAL



Local angles and very small shapes

Not equal area; scale can vary.

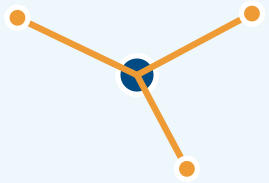
EQUAL-AREA



Relative areas are preserved

Shapes and angles may distort.

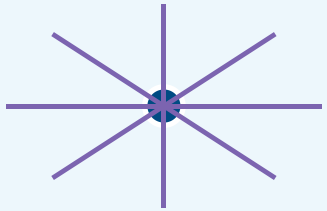
EQUIDISTANT



Distance along selected lines or from a centre

Not every pair of points.

TRUE DIRECTION (CENTRE)



Bearings measured from one chosen centre

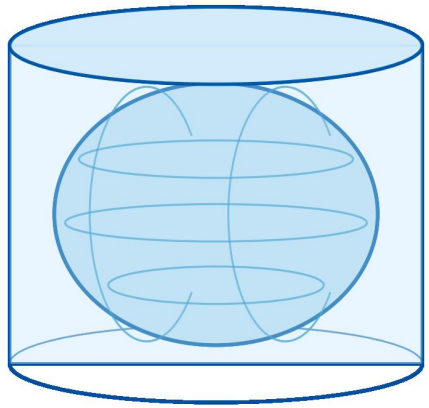
Not every direction between all points.

Key distinction: projection properties describe distortion; families describe construction.

Three Common Map Projection Families

Three geometric models for organizing map projections.

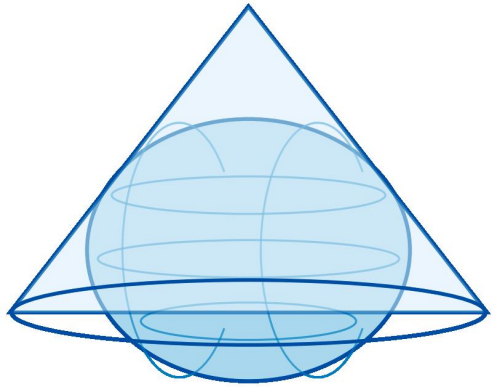
CYLINDRICAL



A cylindrical surface

e.g. Mercator

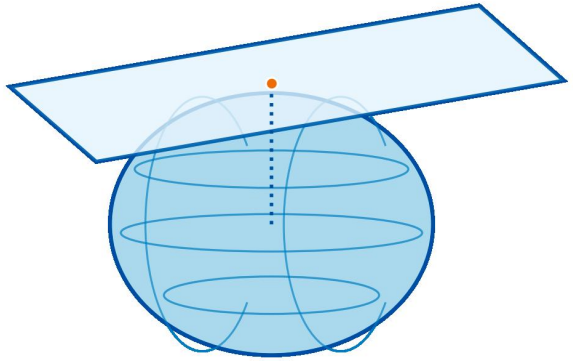
CONIC



A conical surface

e.g. Albers Equal Area

AZIMUTHAL



A plane

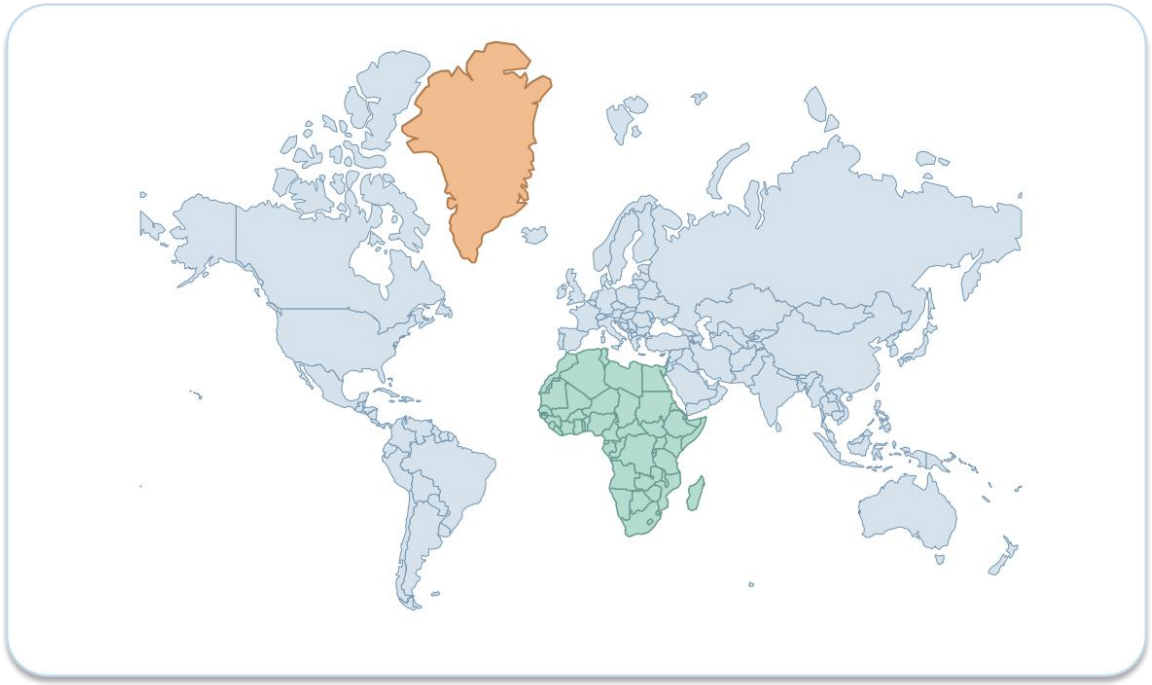
e.g. Lambert Azimuthal Equal Area

Projection family and distortion property are two different classifications.

Mercator vs Equal-Area World Maps

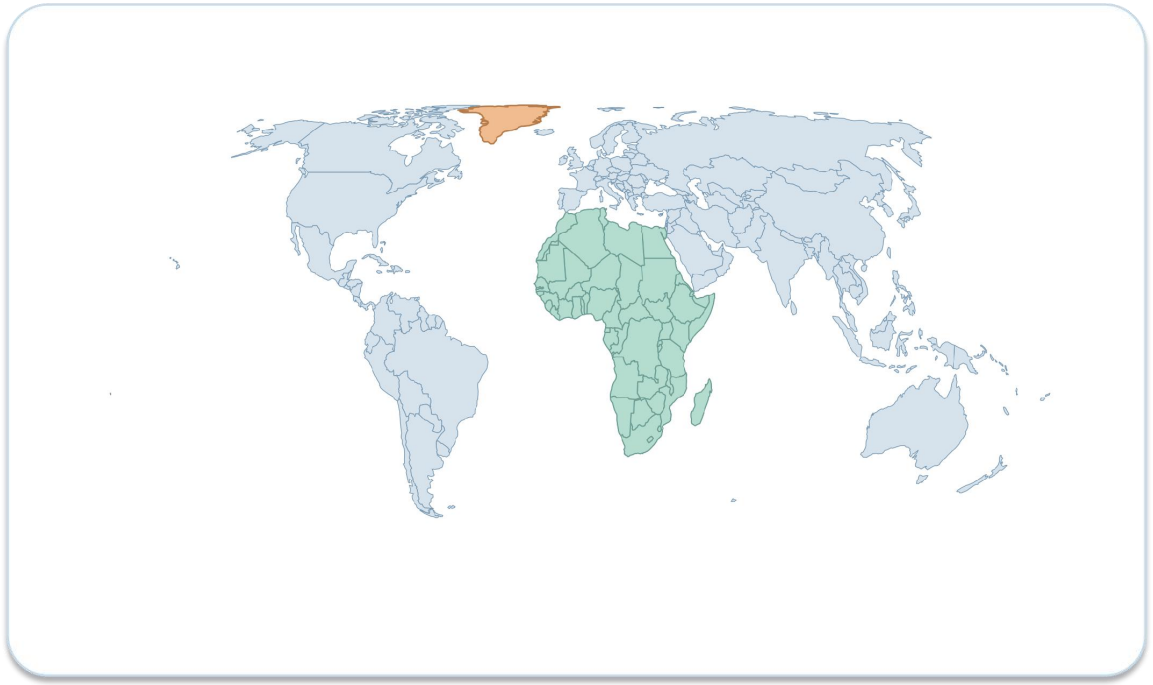
The same country boundaries look different under different projections.

MERCATOR · conformal



High latitudes appear enlarged

EQUAL EARTH · equal-area



Relative areas are preserved

Compare Greenland and Africa: apparent map size depends on projection.

UTM: Local Projections in Different Zones

UTM uses 60 zones, typically 6° wide, between 80°S and 84°N.



ZONE 33N

12°E – 18°E

WGS 84 / UTM 33N

EPSG:32633

Projected in metres

Vienna at 16.37°E falls inside UTM zone 33N.

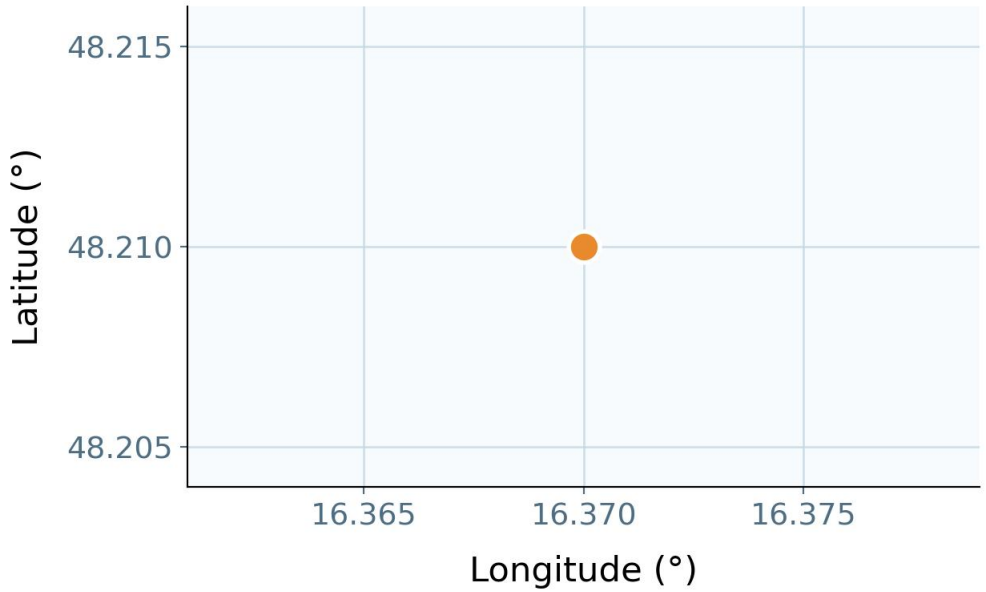
EPSG Codes Identify Coordinate Reference Systems

CRS identifier	Definition	Units / purpose
EPSG:4326	WGS 84 geographic	Degrees
EPSG:32633	WGS 84 / UTM zone 33N	Metres · Vienna region
EPSG:3035	ETRS89 / LAEA Europe	Metres · equal-area Europe

An EPSG code identifies a full CRS, not only a projection method.

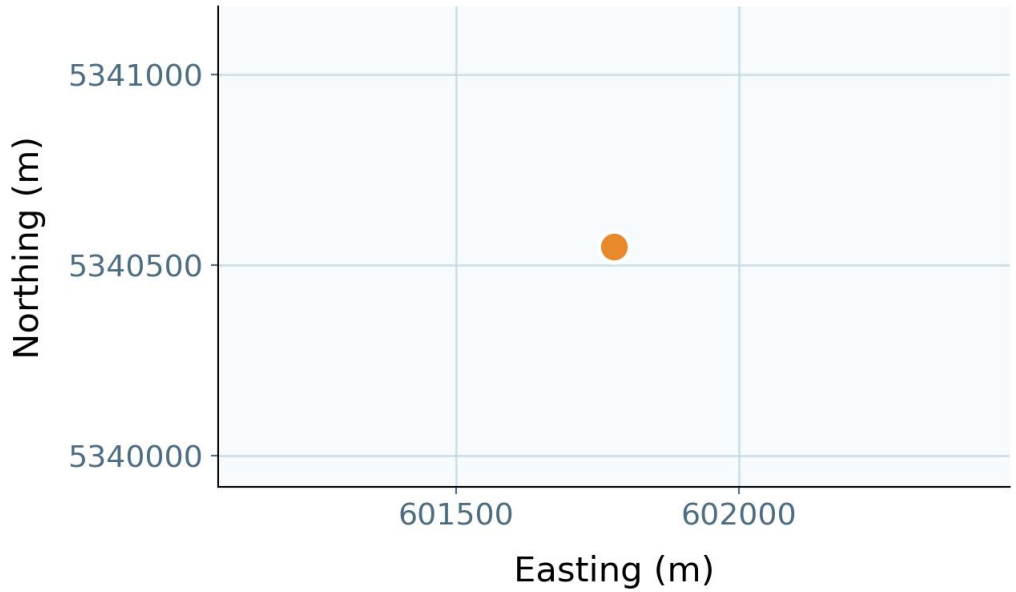
Same Place, Different Coordinates

GEOGRAPHIC · EPSG:4326



16.3700° E, 48.2100° N

PROJECTED · EPSG:32633



601,779 m E, 5,340,549 m N

Same location on Earth · different coordinates and units.

Choosing a CRS: Match the Task and the Area

Spatial task	Appropriate example	Why
Exchange geographic positions	EPSG:4326	WGS 84 geographic coordinates
Local measurements · Vienna	EPSG:32633	Local UTM projection, metres
Compare areas · Europe	EPSG:3035	European equal-area projection

Choose based on region, purpose, distortion and measurement units.

Assign a CRS: Interpret Existing Coordinates

We know the source is WGS 84, but its CRS metadata is missing.

BEFORE

(16.37, 48.21)

CRS: missing



AFTER · set_crs()

(16.37, 48.21)

CRS: EPSG:4326

CRS assignment adds metadata; the coordinate numbers stay the same.

GeoPandas: CRS Assignment in Code

Input: WGS 84 coordinates without CRS metadata

In the notebook, missing_crs is a GeoDataFrame created without CRS metadata.

```
labelled = missing_crs.set_crs("EPSG:4326")
```

GEOMETRY

Same coordinate values

METADATA

CRS = EPSG:4326

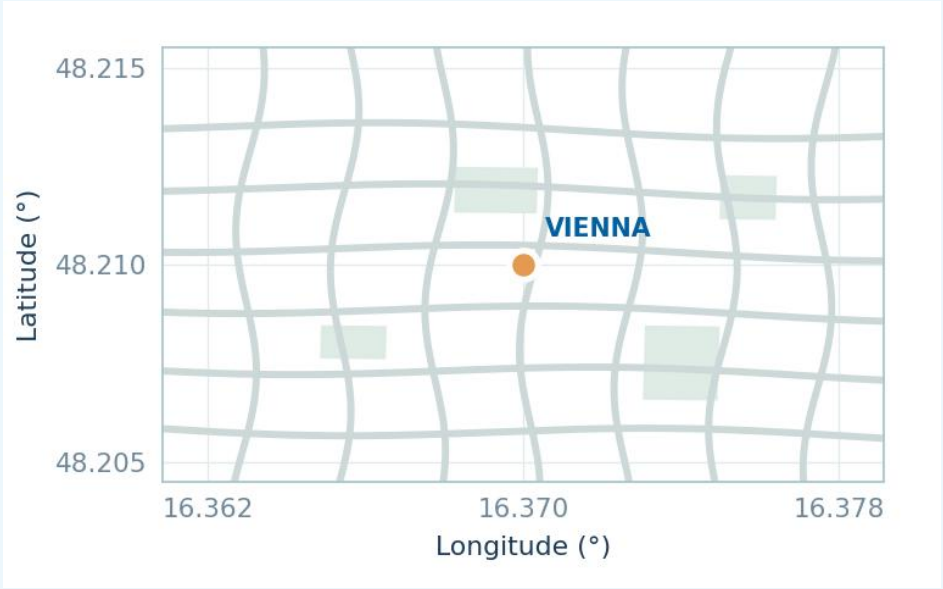
set_crs() labels the original coordinates; to_crs() transforms them.

Transform Coordinates: Change Numbers, Not Location

The same point stays in Vienna; the CRS determines the coordinates used to describe it.

GEOGRAPHIC · EPSG:4326

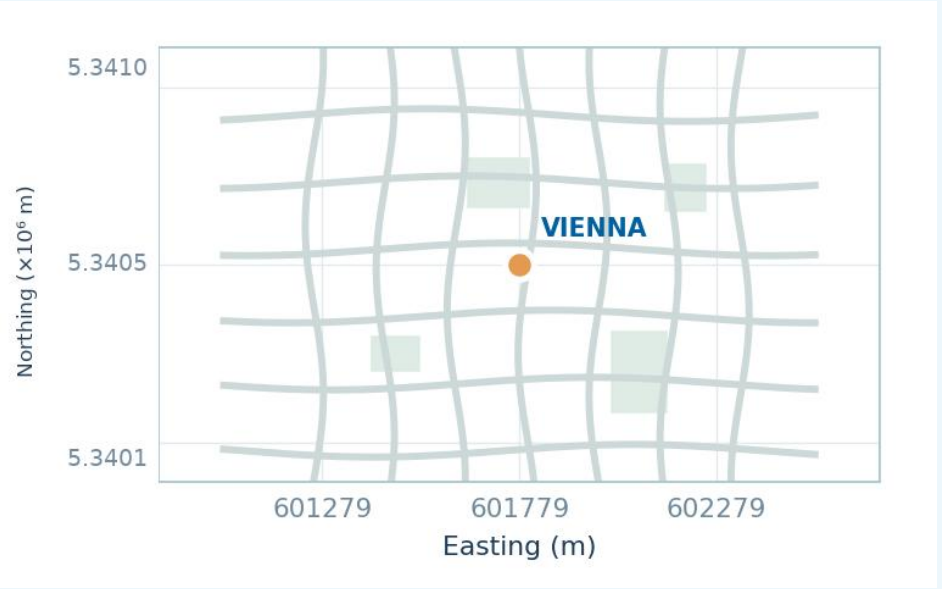
Angles (longitude, latitude)



(16.3700°, 48.2100°)

PROJECTED · EPSG:32633

Distances in a local grid (metres)



(601,779 m E, 5,340,549 m N)

GeoPandas: CRS Transformation in Code

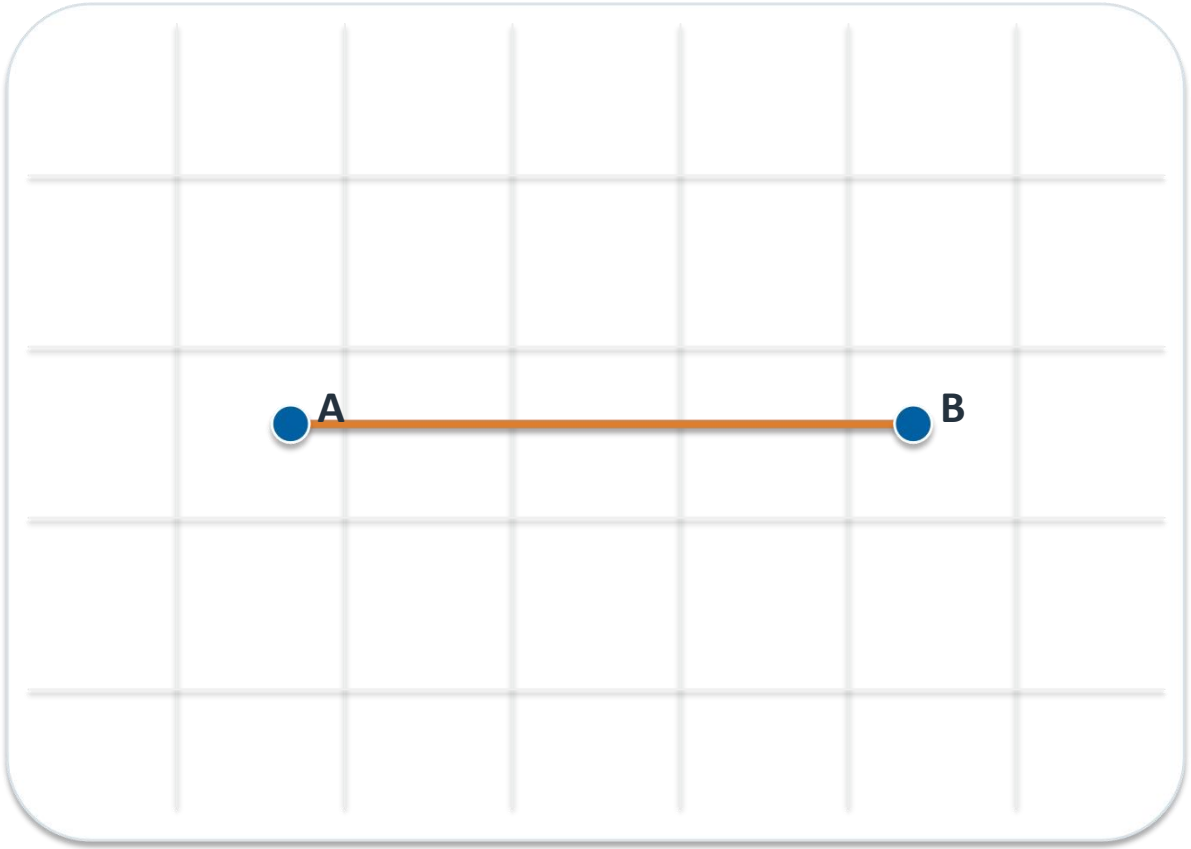
```
gdf_utm = gdf.to_crs("EPSG:32633")
```

Before	After
EPSG:4326 → degrees	EPSG:32633 → metres
(16.37, 48.21)	(601779, 5340549)

GeoPandas uses the source CRS to calculate the new coordinates.

Distance in Geographic Coordinates: What Are the Units?

```
A = Point(16.37, 48.21);  B = Point(16.38, 48.21)  # WGS 84 lon/lat
```




```
round(A.distance(B), 2)
→ 0.01
```

0.01 coordinate degrees — not metres

Shapely calculates planar distance from coordinate values.

Distance After Projection: Metres

The points stay in place. The coordinate units—and the distance result—change.

SAME TWO LOCATIONS NEAR VIENNA



A (16.37°E, 48.21°N)

B (16.38°E, 48.21°N)

GEOGRAPHIC · EPSG:4326

0.01 degrees

→

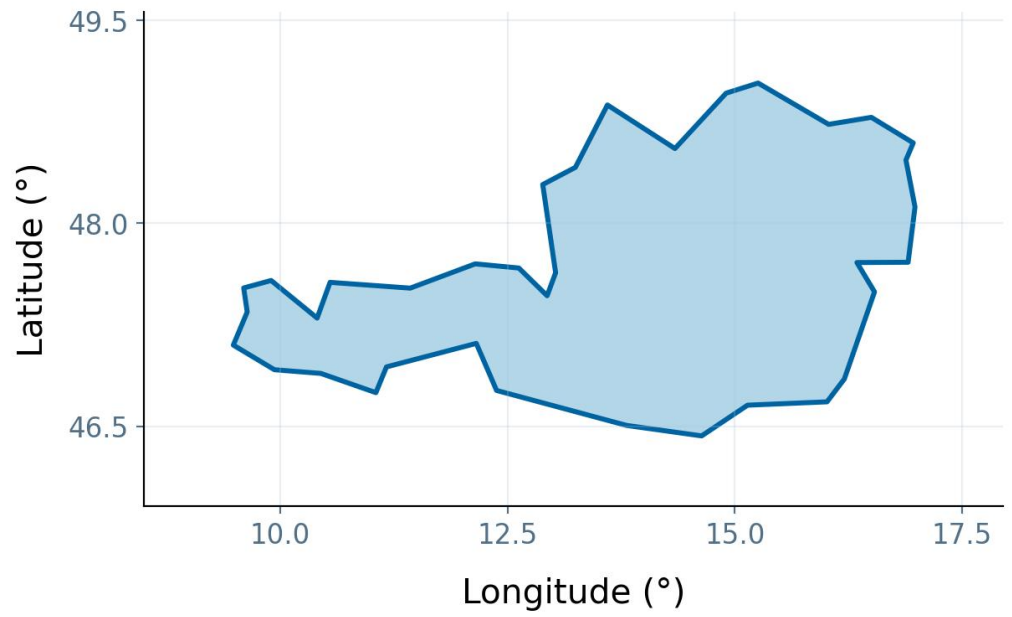
PROJECTED · EPSG:32633

≈ 743 metres

Transform to a suitable projected CRS, then measure.

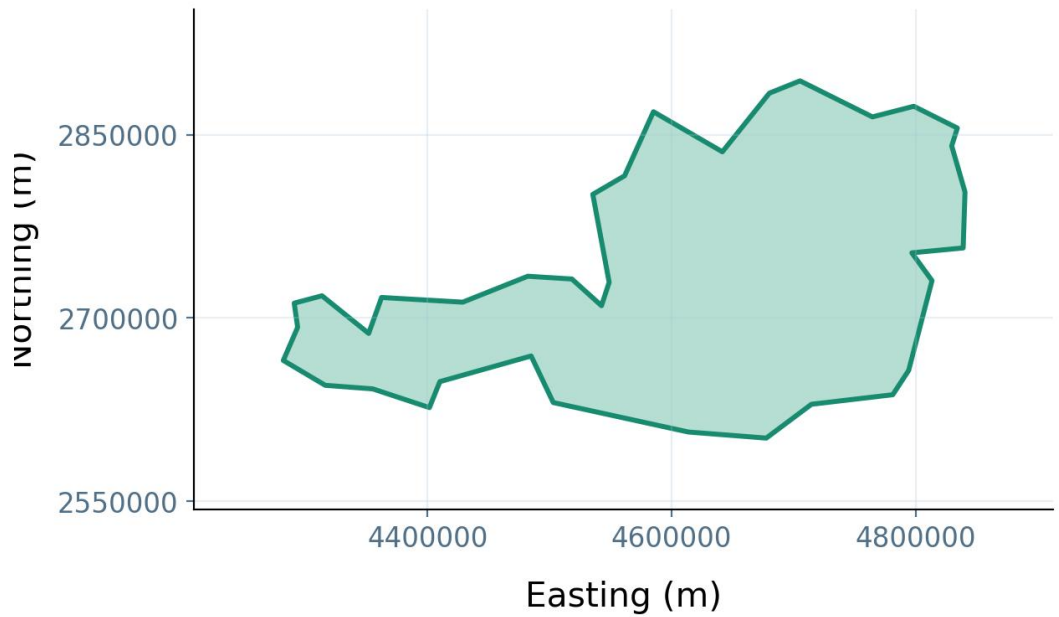
Area: Why Projection Choice Matters

GEOGRAPHIC · EPSG:4326



Angular coordinates · degrees

EQUAL-AREA · EPSG:3035



Projected coordinates · metres

For comparing areas in Europe, EPSG:3035 preserves area in m².

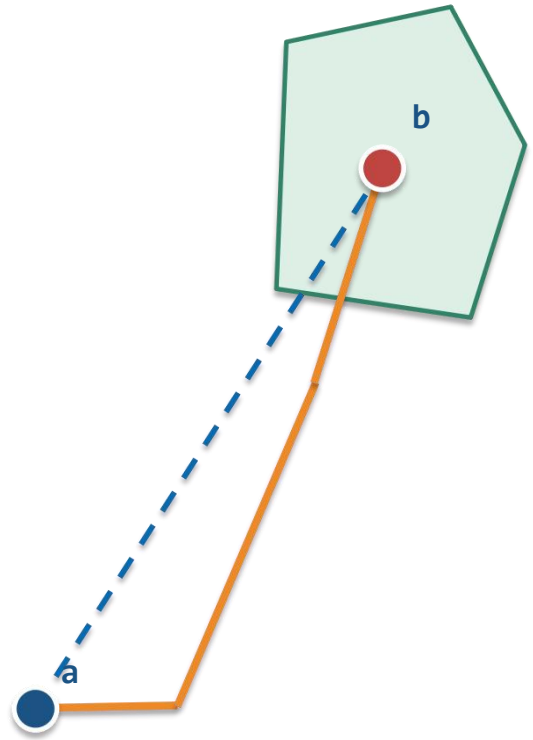
Measuring Geometries with GeoPandas

Lab preview · Synthetic pilot near Vienna; sites_utm, routes_utm and areas_utm come from the notebook.

Blue dashed
straight a–b

Orange
given route

Green
polygon area




```
a, b = sites_utm.geometry
```

```
a.distance(b)
```
Straight-line a–b distance

```
routes_utm.geometry.length
```
Along the given route

```
areas_utm.geometry.area
```
Polygon area

Projected CRS · EPSG:32633

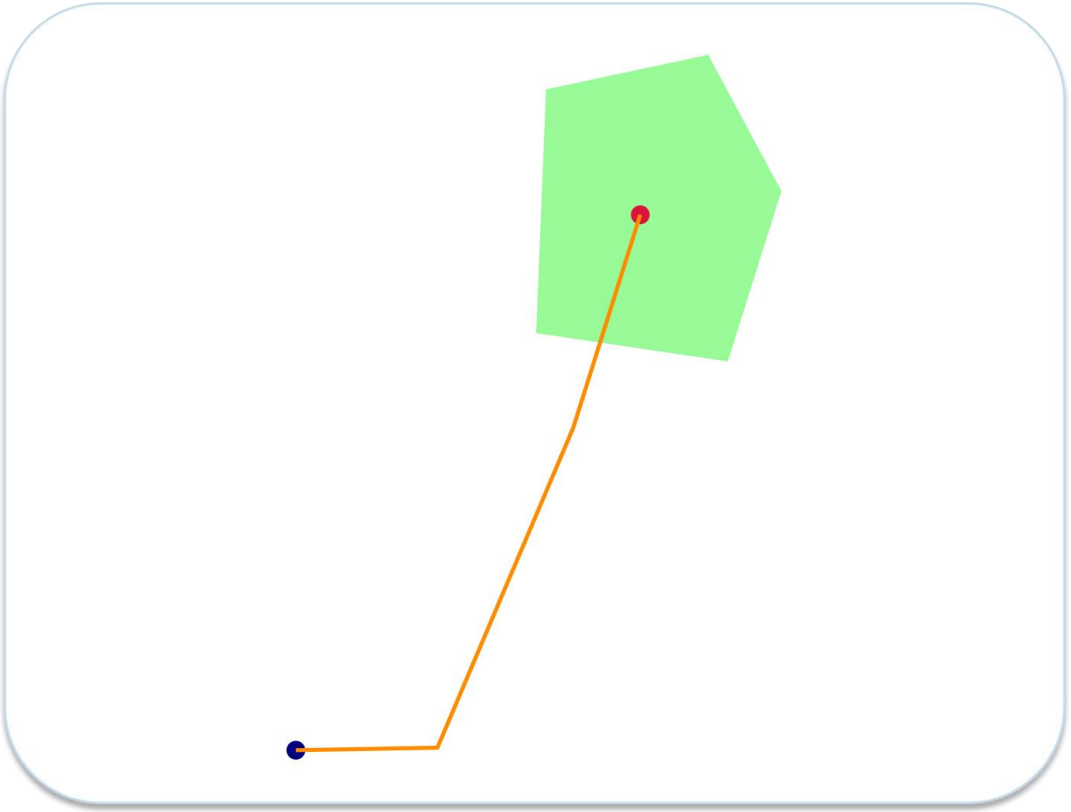
Distance and route length: metres · Polygon area: m²

Matplotlib and GeoPandas: Plotting Geometries

PYTHON CODE

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(7, 4))
areas_utm.plot(ax=ax, color="palegreen")
routes_utm.plot(
    ax=ax, color="darkorange")
sites_utm.plot(
    ax=ax, color=["navy", "crimson"])
ax.set_aspect("equal")
ax.set_axis_off()
plt.show()
```

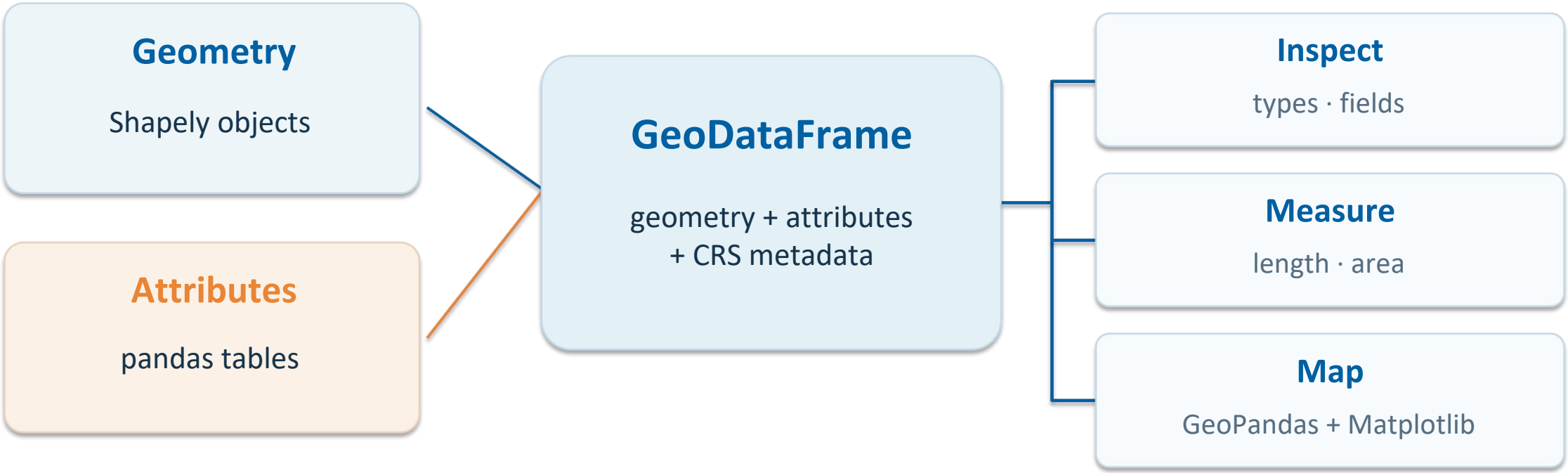
MAP OUTPUT



Point, LineString and Polygon layers share one projected coordinate system.

How Vector Data Concepts and Python Tools Fit Together

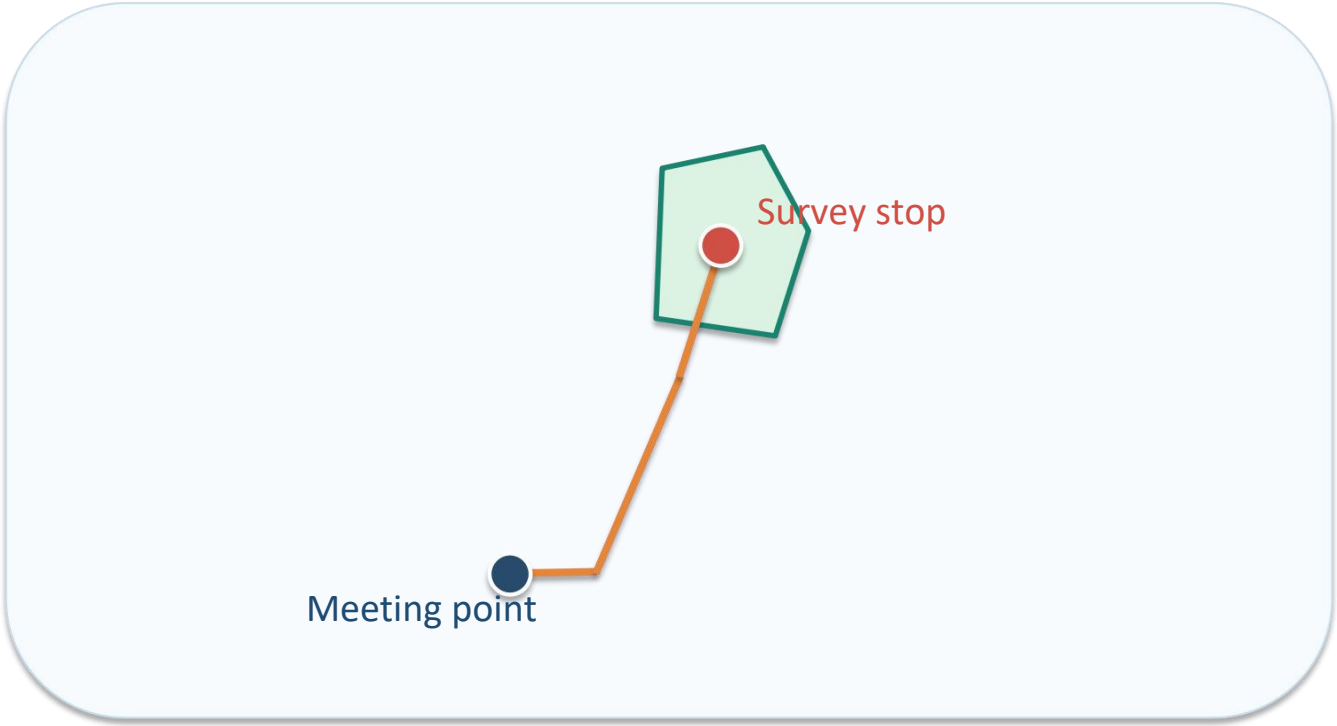
Shapely works with geometry objects; pandas handles tables; GeoPandas combines both.



Geographic meaning comes from the data and its CRS; tools help us analyse it.

Guided Lab Scenario: A Pilot Field Visit

Synthetic pilot microclimate visit: assess a supplied route to a survey zone near Vienna.



POINT

Meeting point / survey stop

LINESTRING

Supplied route geometry

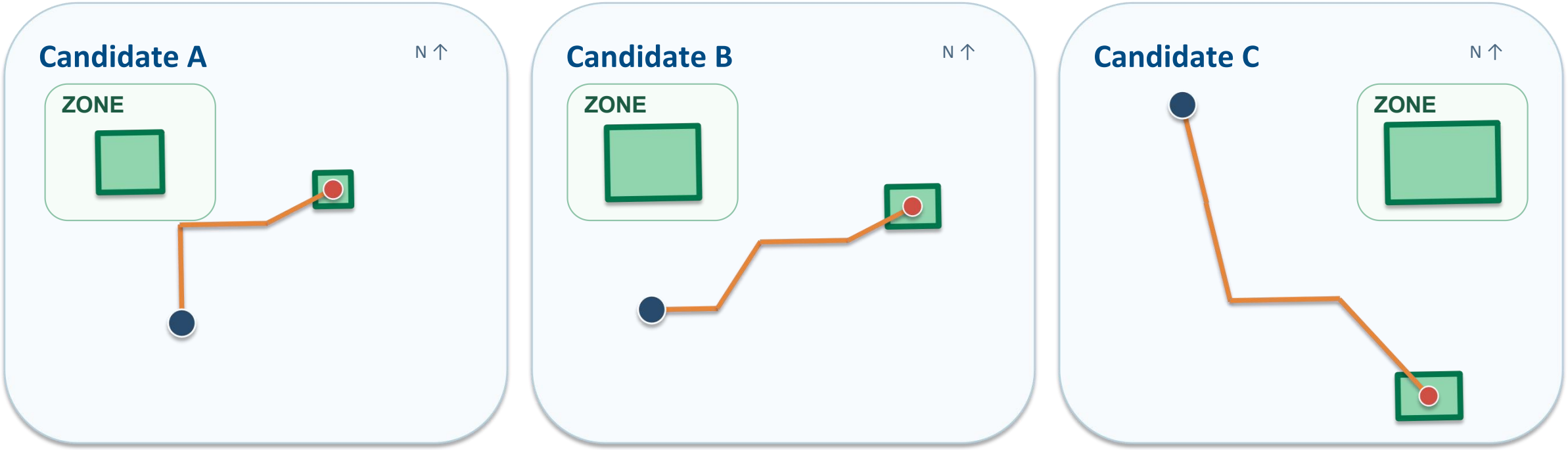
POLYGON

Planned survey zone

Guided: model the route and zone, measure length and area, then map them.

Checkpoint Scenario: Which Survey Zone Should We Choose?

After the guided pilot, compare three new candidate survey zones using the supplied data.



Decision: area ≥ 18,000 m² → shortest supplied route among qualifying zones.

Output: comparison table + labelled map + justified choice.

LAB TIME