Geo-Data and Geo-Data Management

Lecture 5: Raster and Vector–Raster Integration

Yingjing Huang

yingjing.huang@univie.ac.at

2026W

Today’s plan

1



Lecture

- Raster ↔ Raster: Mosaic
- Raster ↔ Raster: Difference
- Point ↔ Raster: Sampling
- Polygon ↔ Raster: Zonal statistics
- Midterm briefing + practice quiz

2

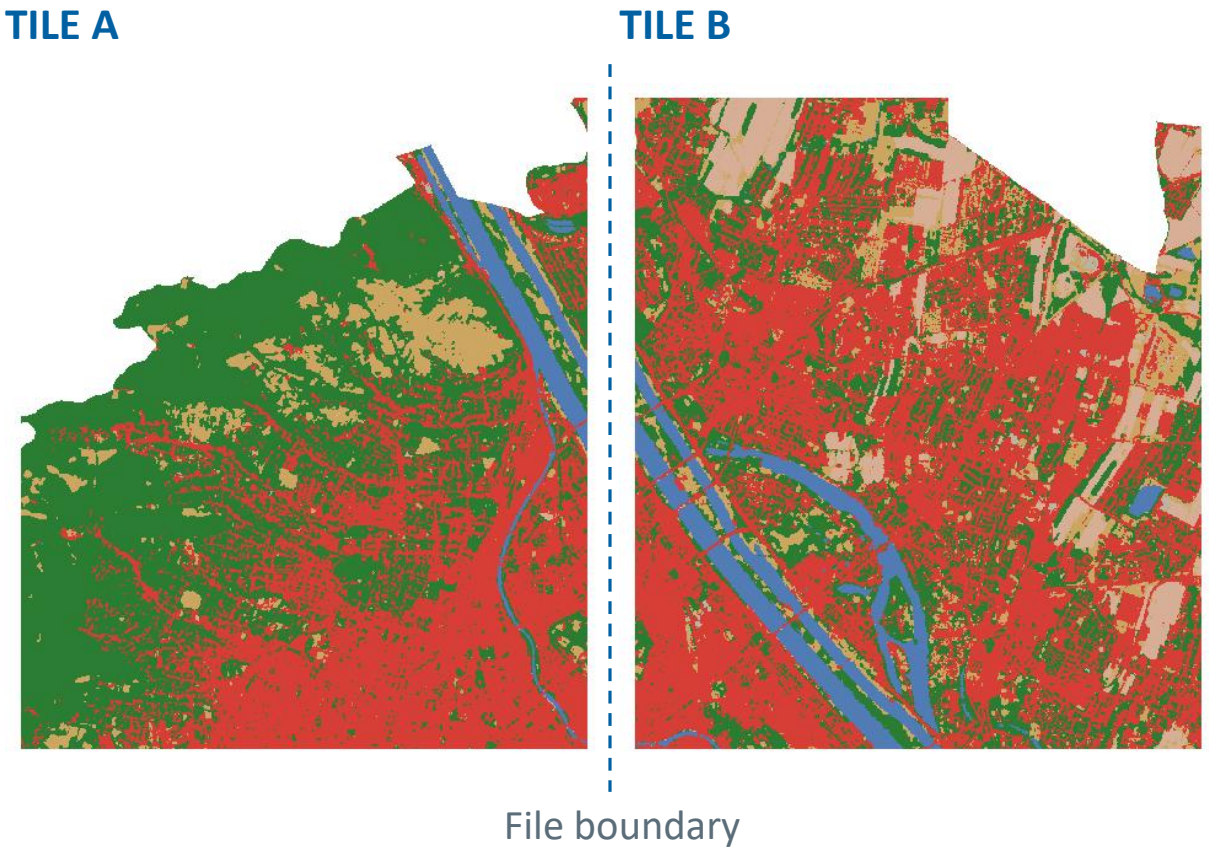


Lab

- Guided: synthetic + real Vienna example
- Independent checkpoint: tree cover across 23 districts

What If Our Study Area Spans Several Raster Tiles?

One study area can cross a raster file boundary.



OUR QUESTION

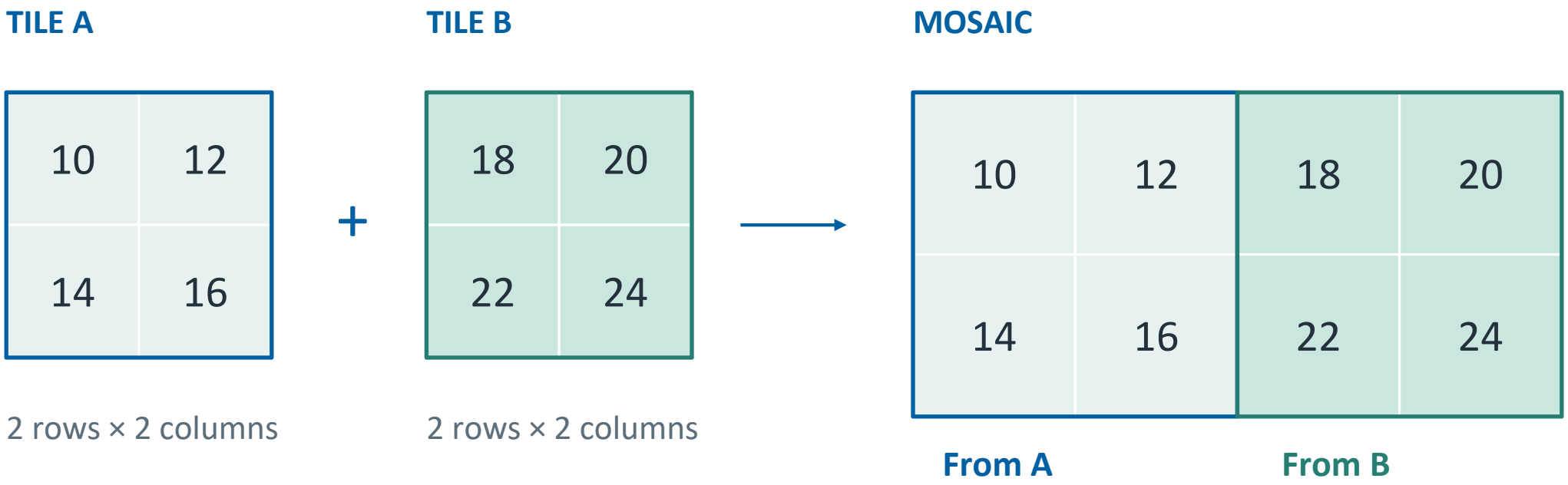
How do we analyse one continuous area?

Use both tiles, after checking that their grids fit together.

Source: ESA WorldCover 2021 tiles retained from the course deck.

Raster Mosaic: Joining Two Raster Tiles

Join the columns of two adjacent, aligned tiles.

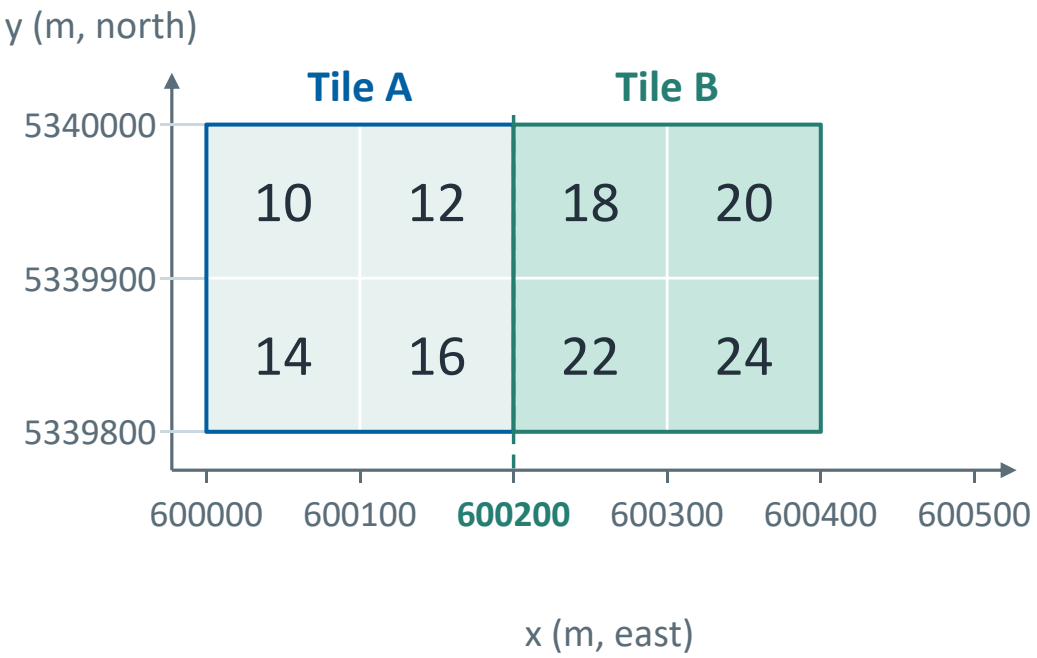


More columns and wider coverage. The original cell values stay the same.

Are These Two Raster Tiles Really Adjacent?

Check where the edges lie, not just the array shapes.

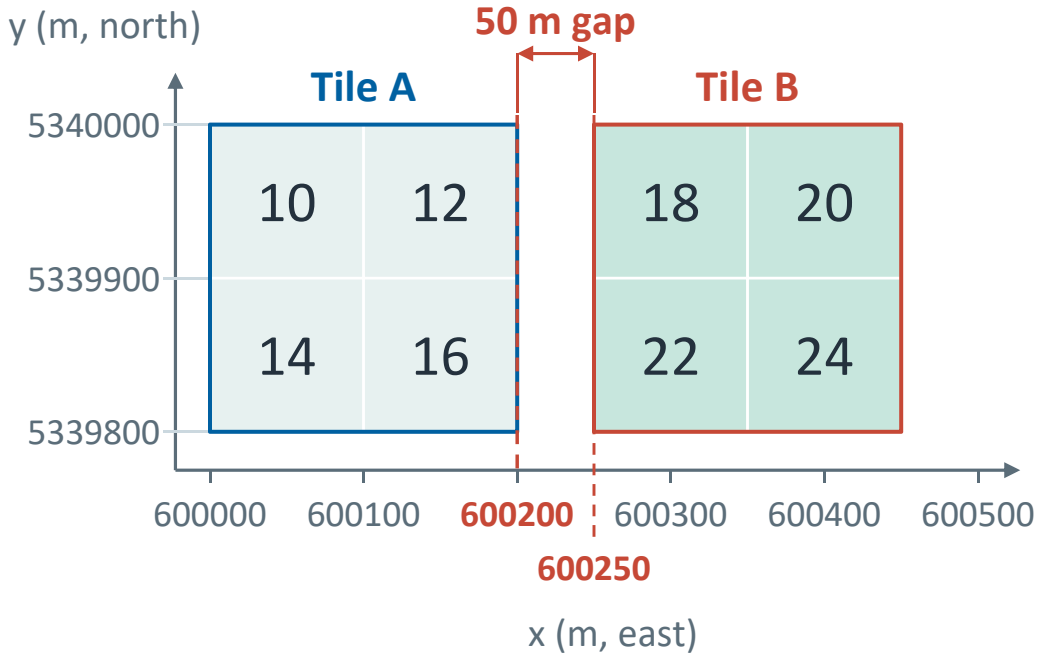
A AND B ALIGNED



A right = B left = 600200

The two edges meet; the rows also line up.

B SHIFTED 50 m EAST



600250 – 600200 = 50 m

The same values now occupy different locations.

Adjacency is a geographic check, not an array-shape check.

Read and Compare Raster Bounds

Read the file boundaries, then compare A's right edge with B's left edge.

READ FROM THE TWO FILES

```
with rasterio.open(path_a) as src_a:
    a_right = src_a.bounds.right

with rasterio.open(path_b) as src_b:
    b_left = src_b.bounds.left

print(a_right, b_left)
```

path_a and path_b are the paths to the two raster files.

Here: same CRS, north-up grids, same cell size and aligned rows.

COMPARE THE TWO CASES

Case	A right	B left
Aligned	600200	600200
	Edges meet.	
Shifted	600200	600250
	50 m gap.	

Read the bounds, compare the edges, then join the aligned arrays.

What If We Do Not Know How the Tiles Meet?

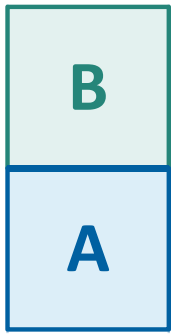
Where is B, and do the tile edges line up?

Side by side



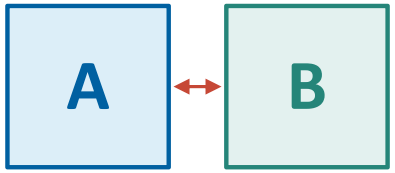
Matching top and bottom

Above or below



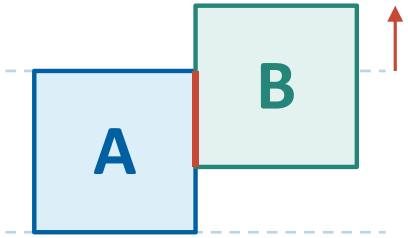
Matching left and right

Gap



Edges do not touch

Offset



Only part of an edge meets

Overlap



Some area is covered twice

Can we join them directly? Check the grids first, then compare the boundaries.

Check the Grid Before Comparing Edges

Use the same CRS and cell size; this example uses unrotated, north-up grids.

CHECK THE OPEN RASTERS

Both CRS values must be defined and match.

```
src_a.crs == src_b.crs                    True
```

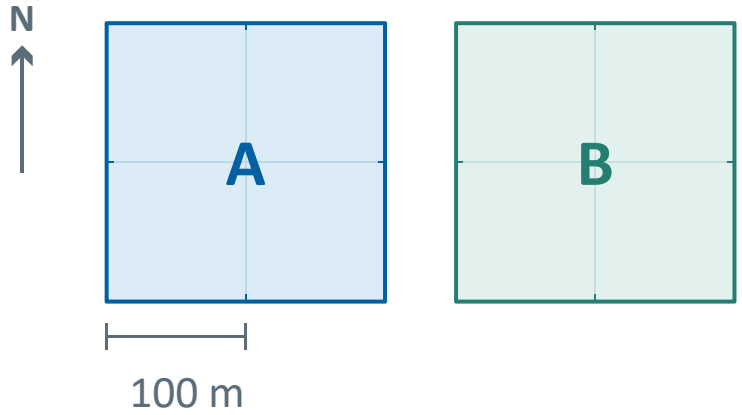
Cell sizes must match.

```
np.allclose(src_a.res, src_b.res,         True
    rtol=0, atol=1e-6)
```

NAME THE TWO BOUNDS

```
a = src_a.bounds
b = src_b.bounds
```

UNROTATED, NORTH-UP GRIDS



Same cell size and orientation

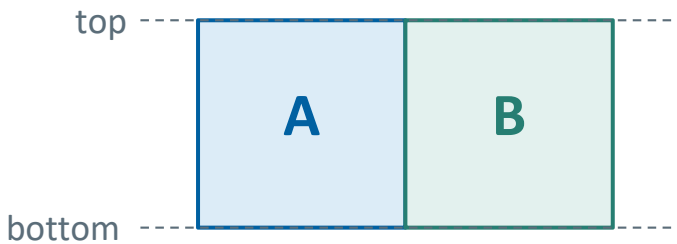
a and b each contain:
left, bottom, right and top

NumPy's np.allclose() uses a tiny absolute tolerance here. Orientation checks stay in the notebook.

Step 1: Do the Top and Bottom Match?

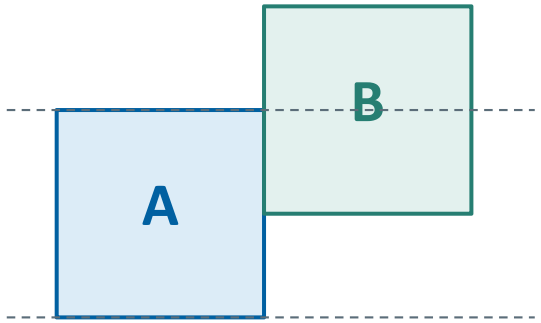
For left/right neighbours, first check whether the vertical ranges match.

MATCHING VERTICAL EXTENT



same_y = True

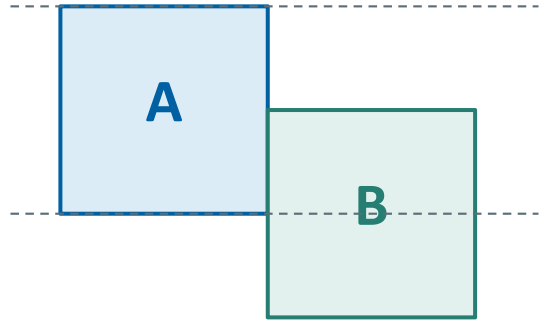
SHIFTED NORTH



top and bottom do not both match

same_y = False

SHIFTED SOUTH



top and bottom do not both match

same_y = False

```
same_y = np.allclose(
    [a.bottom, a.top],
    [b.bottom, b.top],
    rtol=0, atol=1e-6)
```

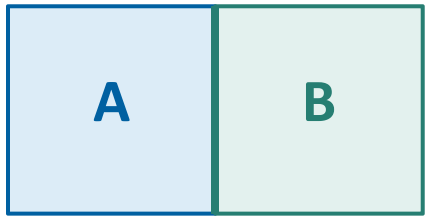
Compare the two vertical ranges.

same_y is True only when both the top and bottom boundaries match.

Step 2: Is B to the Left or Right?

Once same_y is True, compare which left/right edges meet.

B EAST OF A



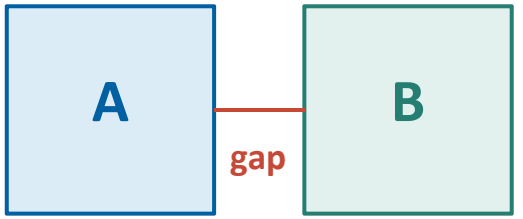
east = True

B WEST OF A



west = True

SAME Y, BUT NO TOUCHING EDGE



east = False west = False

```
east = same_y and np.allclose(
    a.right, b.left, rtol=0, atol=1e-6)
west = same_y and np.allclose(
    a.left, b.right, rtol=0, atol=1e-6)
```

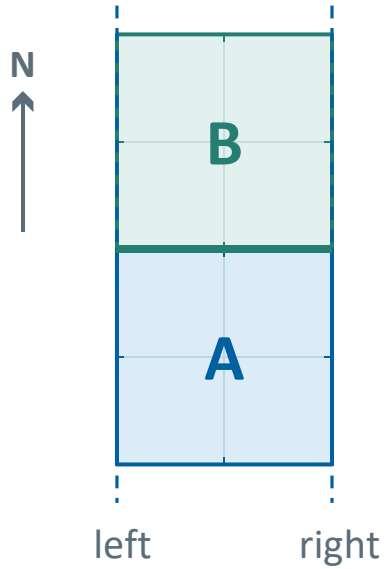
Only after same_y = True

First match the vertical extent. Then check which left/right edges actually meet.

Above or Below: Match the Left and Right

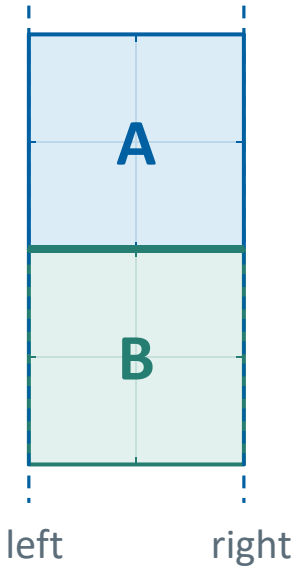
For a vertical join, the full horizontal extent must match.

B NORTH OF A



north = True

B SOUTH OF A



south = True

TEST B'S POSITION RELATIVE TO A

```
same_x = np.allclose(
    [a.left, a.right],
    [b.left, b.right],
    rtol=0, atol=1e-6)
north = same_x and np.allclose(
    a.top, b.bottom,
    rtol=0, atol=1e-6)
south = same_x and np.allclose(
    a.bottom, b.top,
    rtol=0, atol=1e-6)
```

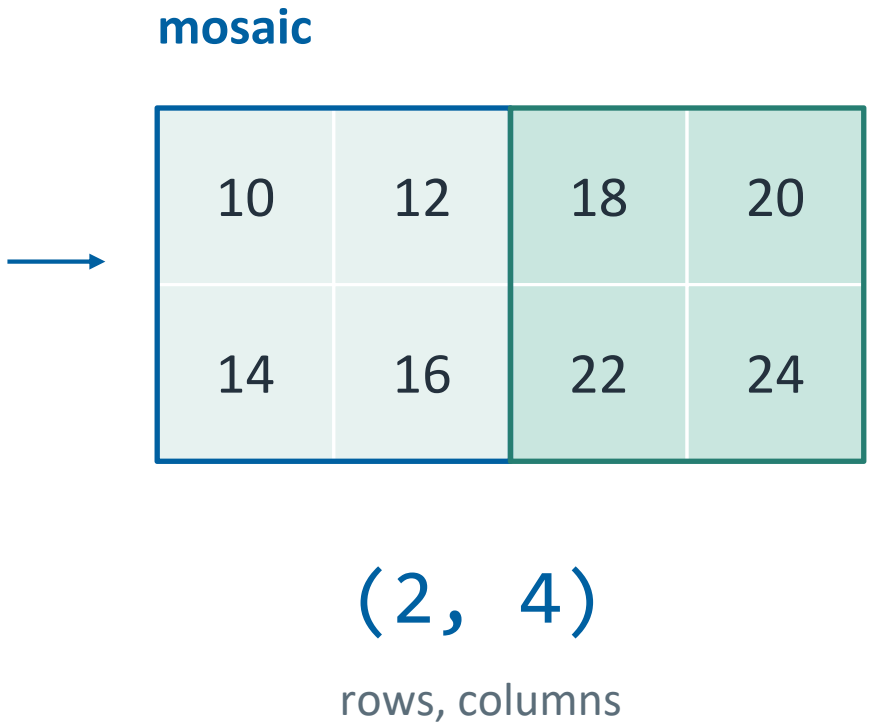
First match the horizontal extent.
Then check which top/bottom edges meet.

Joining Raster Tiles with NumPy

Once alignment is checked, np.hstack() joins columns.

```
tile_a = np.array([[10, 12],
                   [14, 16]])
tile_b = np.array([[18, 20],
                   [22, 24]])

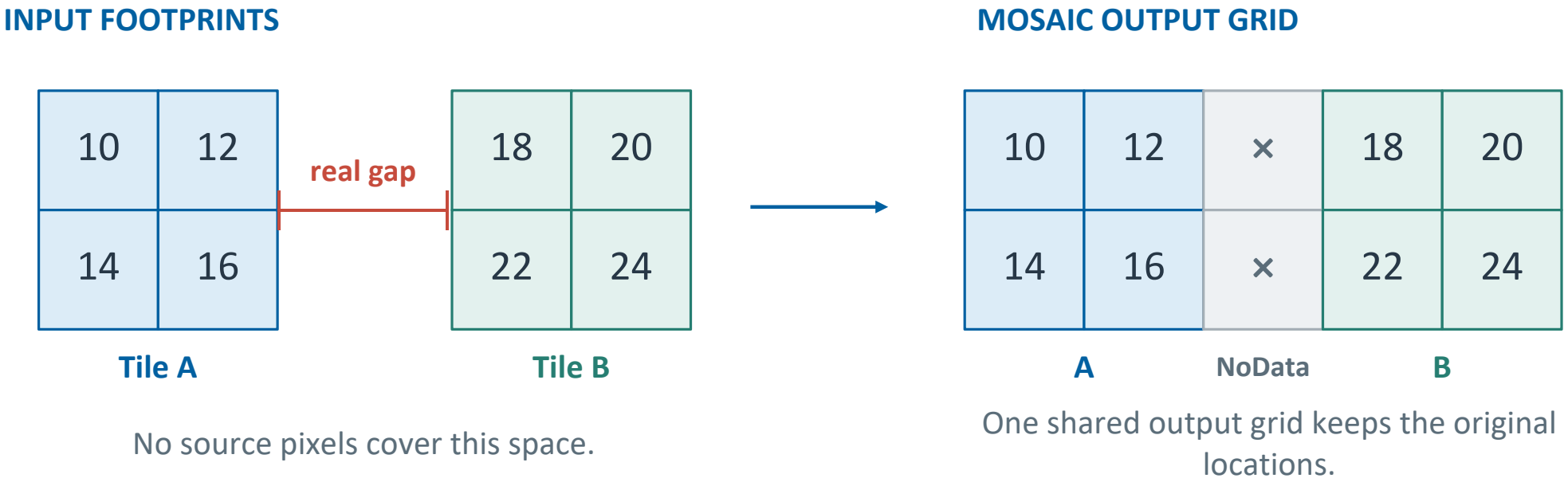
mosaic = np.hstack([tile_a, tile_b])
print(mosaic.shape)  # (2, 4)
```



np.hstack() returns an array of values

Optional: A Mosaic Can Include Gaps

Tiles do not have to touch perfectly; uncovered cells can remain NoData in the output grid.



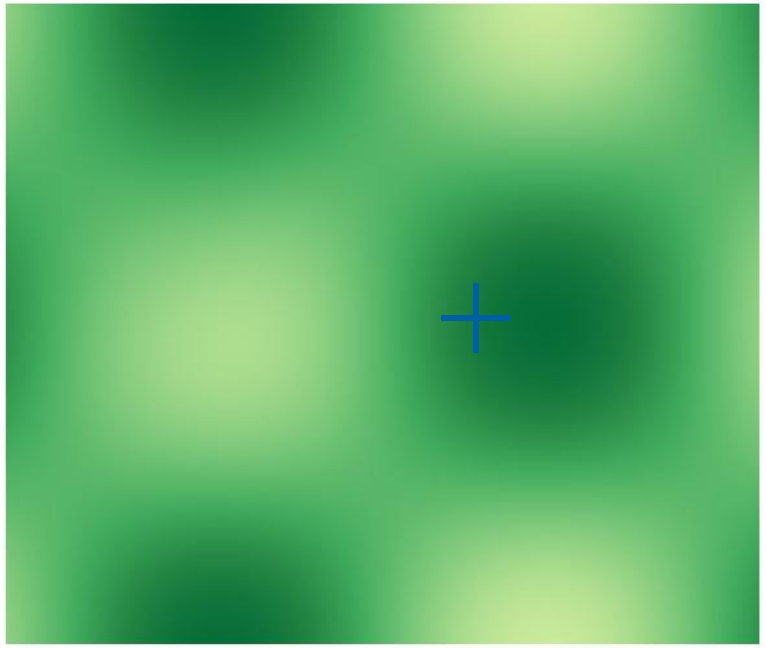
The mosaic preserves the geographic gap instead of shifting B next to A.

Uncovered cells can be represented as NoData.

How Has Vegetation Changed Between Two Dates?

This time we compare matching locations, rather than joining neighbouring tiles.

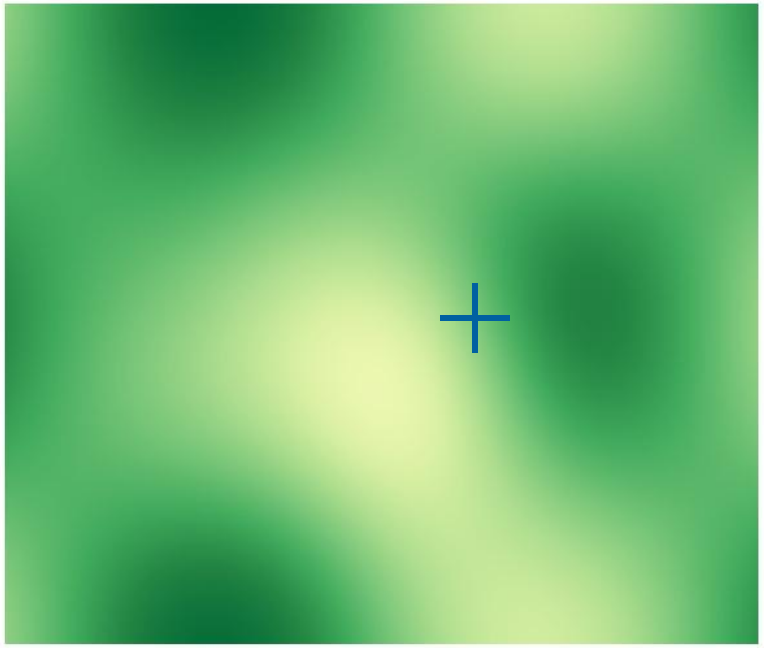
DATE 1



↔

Same location

DATE 2

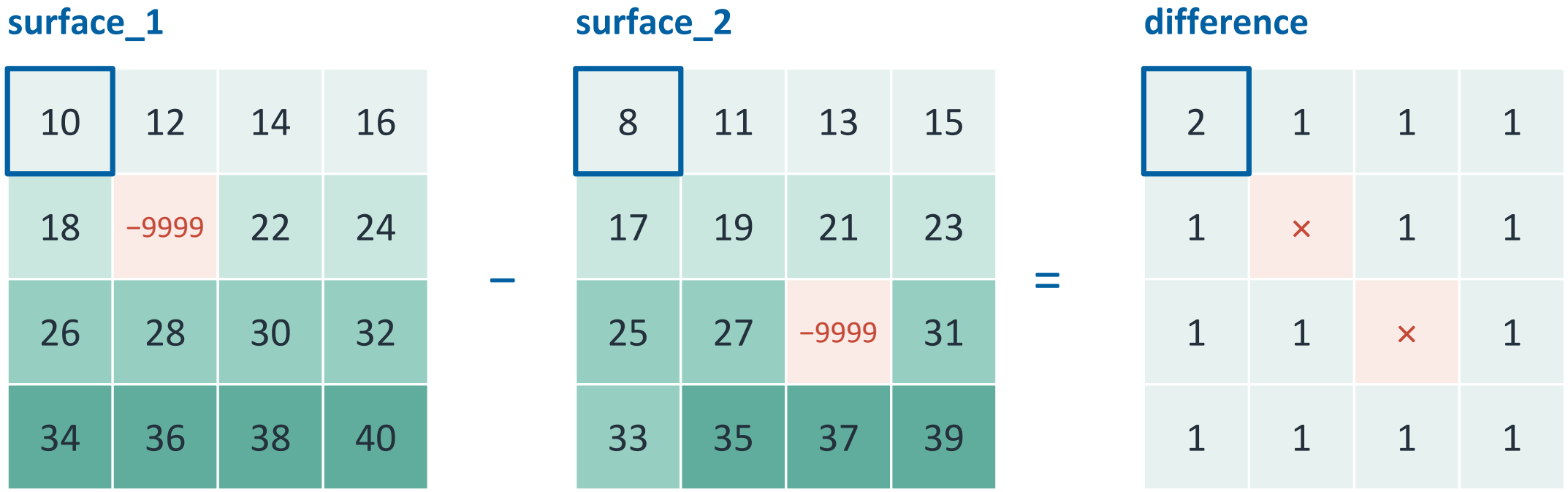


Which locations changed, and by how much?

Illustrative surfaces from the course deck; same extent and colour scale. Next: a small synthetic example.

Raster Difference: Compare the Same Cell

Each output cell is surface_1 minus surface_2 at one matching location.



row 0, col 0: 10 − 8 = 2

× means that at least one input is missing at that cell.

Check 1: Do the Raster Grids Match?

Before cell-by-cell subtraction, check that both rasters use the same spatial grid.

WHAT TO CHECK

CRS	Defined and matching CRS
Resolution	Same cell size
Shape	Same rows × columns
Transform	Same grid placement and orientation

For cell-by-cell subtraction, all four should match.

WITH BOTH RASTERS OPEN

```
src_1.crs == src_2.crs
```
True

```
np.allclose(src_1.res, src_2.res,
    rtol=0, atol=1e-6)
```
True

```
src_1.shape == src_2.shape
```
True

```
np.allclose(
    src_1.transform,
    src_2.transform,
    rtol=0, atol=1e-6)
```
True

transform checks grid placement and orientation.

If all checks pass, the two rasters use the same grid, so corresponding cells refer to the same location.

Check 2: Do the Values Mean the Same Thing?

A matching grid is necessary, but the measurements also need to be comparable.

VALID COMPARISON

NDVI (date 1) – NDVI (date 2)

Same variable and same scale

NOT MEANINGFUL

10 °C – 8 m

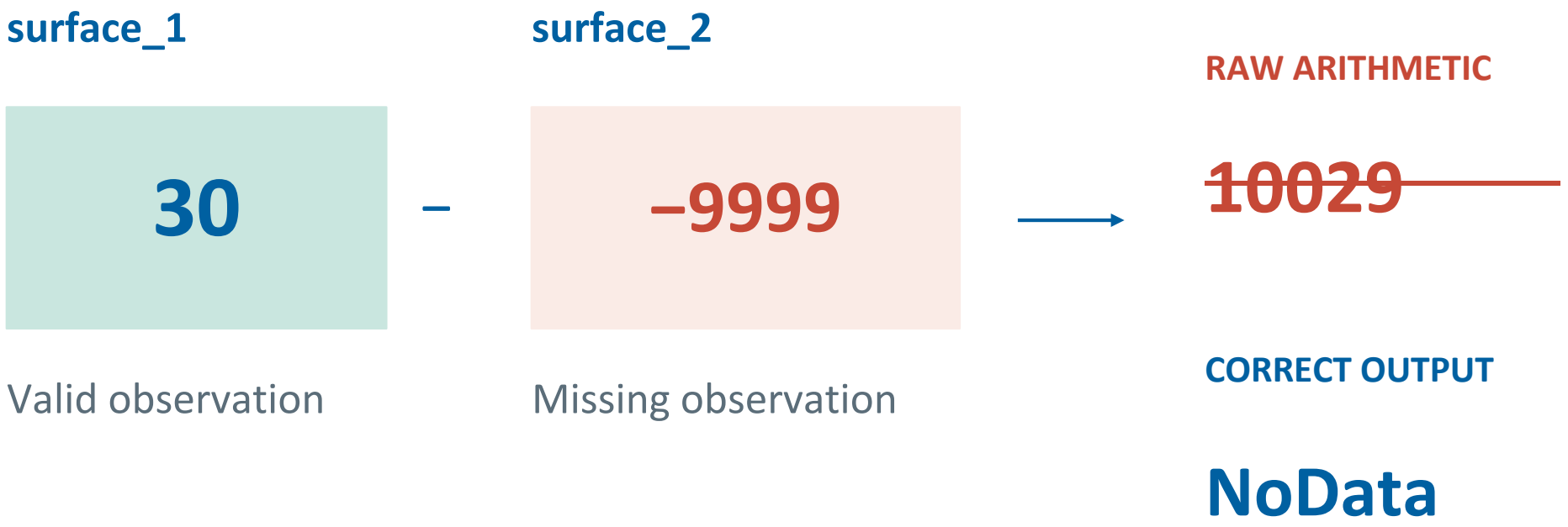
Different quantities / units

BEFORE SUBTRACTING, CONFIRM

- **Same variable**
  NDVI vs NDVI
- **Same units / scale**
  both use the same scale
- **Same interpretation**
  values have the same meaning

A matching grid is necessary, but Python cannot infer whether two measurements are scientifically comparable.

NoData Is Not a Measurement



One missing input → no valid difference

Exclude this position from the comparison, even though Python can subtract the stored numbers.

Select Cells That Are Valid in Both Inputs

A logical AND keeps a position only when both inputs are valid.

valid_1

1	1	1	1
1	0	1	1
1	1	1	1
1	1	1	1

AND

valid_2

1	1	1	1
1	1	1	1
1	1	0	1
1	1	1	1

→

valid

1	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

```
nodata = -9999.0
valid_1 = surface_1 != nodata
valid_2 = surface_2 != nodata
valid = valid_1 & valid_2
```

14 valid cells

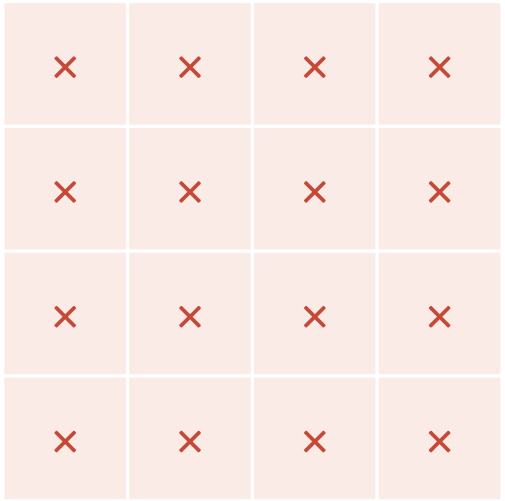
16 positions − 2 excluded

1 = keep; 0 = exclude. The outlined positions fail at least one input check.

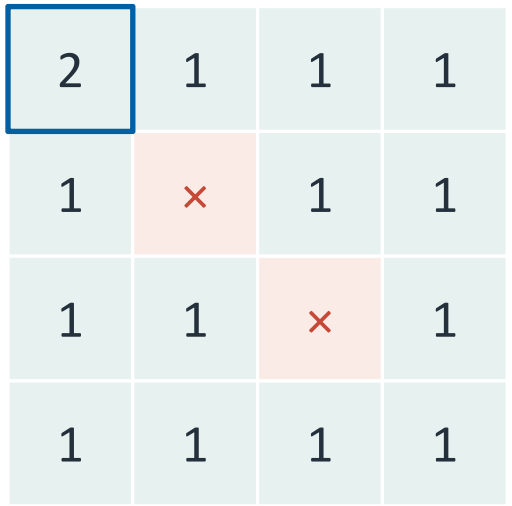
Fill Only the Valid Positions

Start with NoData everywhere, then update cells selected by the mask.

1 Create the output



2 Update valid cells

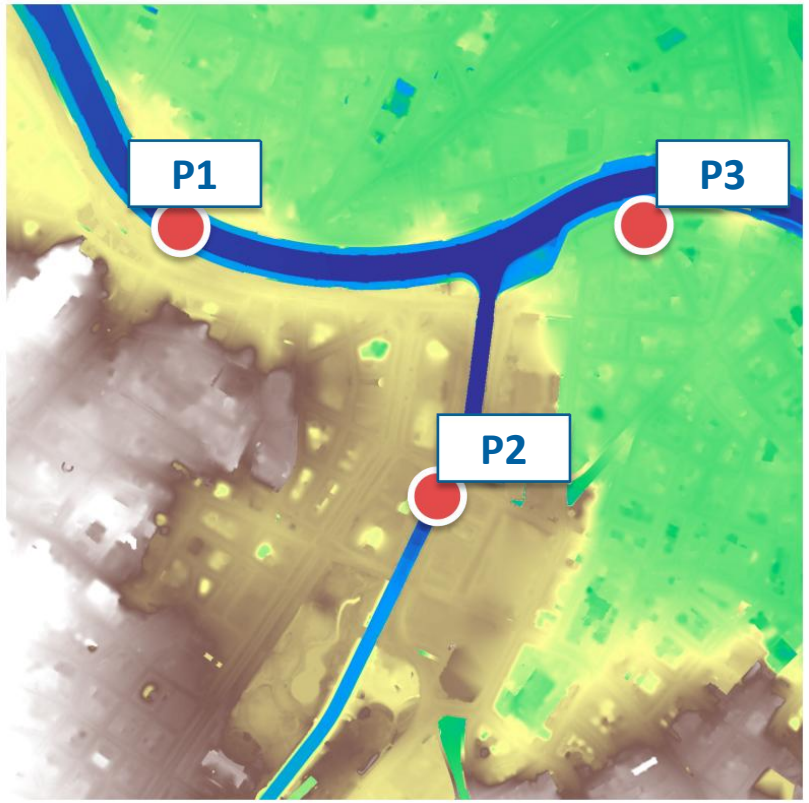



```
difference = np.full(
    surface_1.shape, nodata,
    dtype="float32")
```

```
difference[valid] = (
    surface_1[valid]
    - surface_2[valid])
```

How Can We Find Elevation at Survey Locations?

Use each point coordinate to query the cell underneath it.

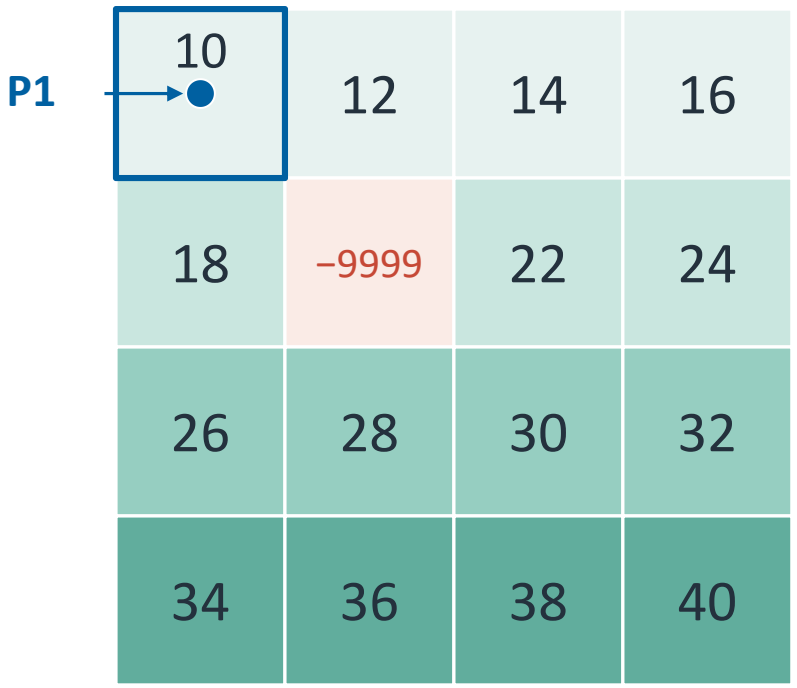


Point	Raster value
P1	?
P2	?
P3	?

Coordinates → cell → value

Point Sampling: Coordinates Choose a Cell

P1 falls inside the upper-left cell of the synthetic raster.



1 **Point coordinate (x, y)**

```
(600050, 5339950)
```

2 **Containing cell**

row 0, column 0

3 **Read the value**

```
surface_1[0, 0]  # 10
```

Sampling returns the value of the containing cell; no interpolation is used here.

From Coordinates to a Column

Use the x offset from the raster origin to find the containing column.

POINT COORDINATE

`P2 = (600250, 5339750)`

1 Find the eastward offset

600250 − 600000 = 250 m

2 Convert distance to cell widths

250 / 100 = 2.5 cells

→ inside column 2

COLUMN INDICES

	0	1	**2**	3
0	10	12	14	16
1	18	−9999	22	24
2	26	28	30	32
3	34	36	38	40

250 m east

Column indices start at 0. An offset of 2.5 cell widths lies inside column 2.

From Coordinates to a Row — Then Look Up the Cell

The array index is [row, column], so calculate y first and combine the two indices.

1 **Find the southward offset**

5340000 − 5339750 = 250 m

250 / 100 = 2.5 cells

→ inside row 2

2 **Combine row and column**

```
surface_1[2, 2]  # 30
```

ROW 2 + COLUMN 2

	0	1	**2**	3
0	10	12	14	16
1	18	−9999	22	24
2	26	28	30	32
3	34	36	38	40

P2 is at the centre of cell [2, 2].

Map coordinates: (x, y). Array lookup: [row, column].

Sample Multiple Points with Rasterio

Once the point coordinates are ready, Rasterio queries the corresponding cells in the open raster.

1 Collect one coordinate pair per point

```
coords = [(p.x, p.y) for p in points.geometry]
```

2 Sample while the raster is open

```
with rasterio.open(raster_path) as src:
    samples = list(src.sample(coords))
```

ONE SAMPLE PER COORDINATE

P1 → sample 1

P2 → sample 2

P3 → sample 3

same order in
→ same order out

Rasterio returns one sample for each coordinate, in the same order.

Store Sampled Values as Point Attributes

Add one raster value to each existing point feature.

BEFORE: POINTS

point_id	geometry
P1	Point(...)
P2	Point(...)
P3	Point(...)



AFTER: point_results

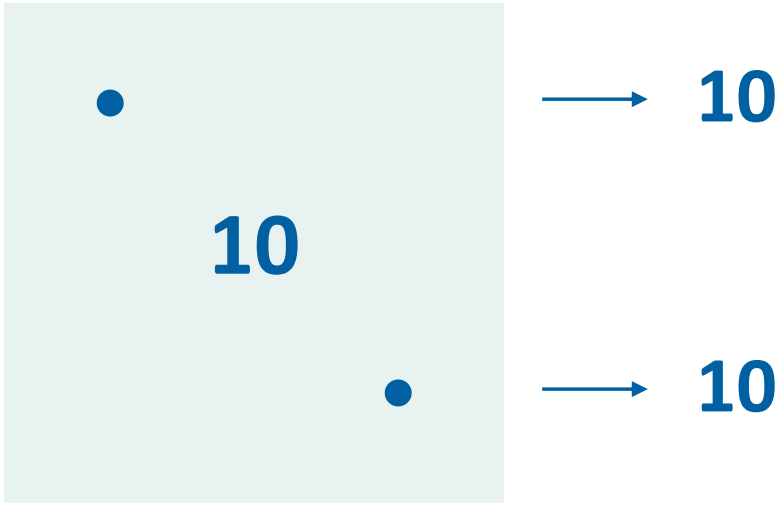
point_id	raster_value	geometry
P1	**10**	Point(...)
P2	**30**	Point(...)
P3	**40**	Point(...)

```
point_results = points.copy()
point_results["raster_value"] = [v[0] for v in samples]
```

What Does a Sampled Point Value Mean?

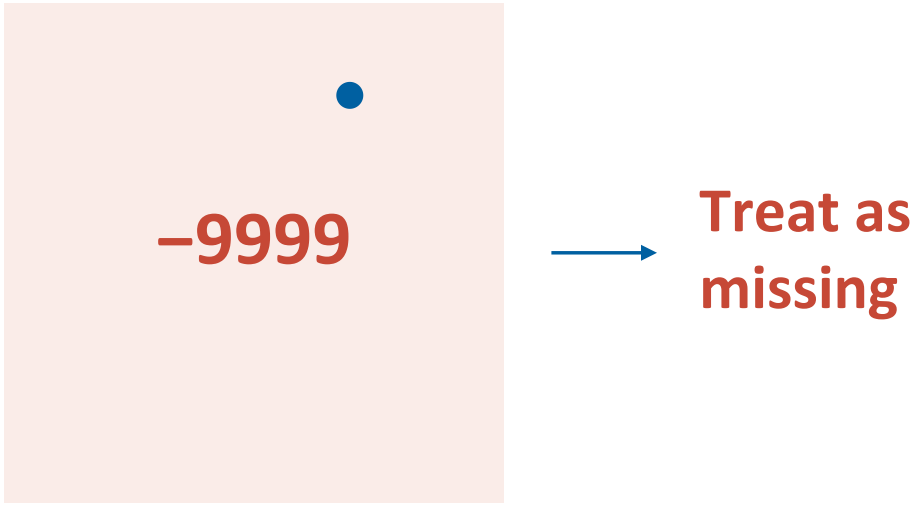
The raster cell, not the coordinate precision, limits the sampled detail.

TWO POINTS, ONE CELL



One cell → one stored value

ONE POINT IN A NODATA CELL



The marker is not an observation.

More precise point coordinates do not create finer raster detail or fill NoData gaps.

How Can We Compare Mean Elevation Across Districts?

A district contains many raster cells, but we want one result per district.



ONE ROW PER DISTRICT

District	Mean elevation
1	?
2	?
3	?

Many cells → one area summary

Zonal Statistics: Select First, Then Summarize

The polygon chooses the cells. The statistic summarizes their valid values.

RASTER + ZONE Z1

10	12	14	16
18	–9999	22	24
26	28	30	32
34	36	38	40



1 Select valid cells

```
[10, 12, 18]
```



2 Summarize the zone

Z1 mean ≈ 13.33

The output describes the area selected by the polygon.

Rasterio mask(): Select the Zone

The polygon selects the cells; crop=True only makes the output rectangle smaller.

RASTER + ZONE Z1

10	12	14	16
18	–9999	22	24
26	28	30	32
34	36	38	40

selected cells

```
clipped, _ = mask(
    src, [zone.geometry],
    crop=True, filled=False)
```

crop=True

→ smaller bounding rectangle

filled=False

→ keep outside cells masked

Crop changes extent; the mask controls which values are usable.

What Does mask() Return?

The result still has a band dimension and keeps the excluded cell masked.

RETURNED ARRAY

```
clipped.shape  # (1, 2, 2)
```

bands × rows × columns

10	12
18	×

The × cell is masked, not a measurement.

1 Select the first 2D grid

```
band = clipped[0]
```

1 band → first 2D grid

2 Keep the unmasked values

```
values = band.compressed()
```

→ [10, 12, 18]

Next: compute the mean from these three valid values.

Repeat the Same Operation for Every Polygon

The loop does not introduce a new spatial operation; it repeats the one-zone workflow.

REPEAT THE ONE-ZONE WORKFLOW

```
rows = []
for _, zone in zones.iterrows():
    clipped, _ = mask(
        src, [zone.geometry],
        crop=True, filled=False)
    values = clipped[0].compressed()
    rows.append({
        "zone_id": zone.zone_id,
        "count": values.size,
        "mean": values.mean()})
```

1. **Select zone**
2. **Select valid pixels**
3. **Append one row**

Z1 → 3 valid cells → 13.33

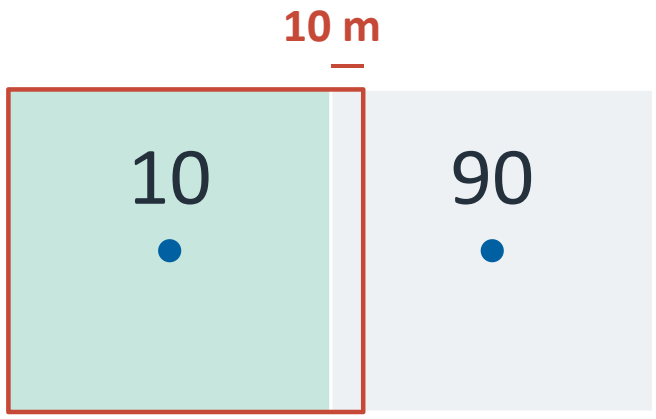
Z2 → 4 valid cells → 35

What If a Polygon Covers Only Part of a Cell?

The polygon reaches 10 m into the second cell; the selection rule changes the result.

CELL CENTRE INSIDE

```
all_touched=False
```

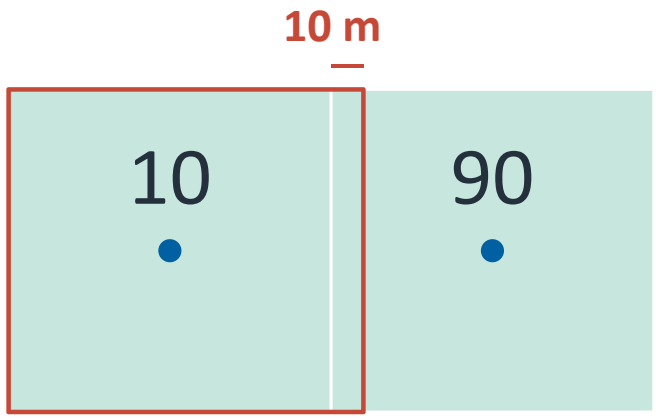


100 m per cell

[10] → mean = 10

EVERY TOUCHED CELL

```
all_touched=True
```



100 m per cell

[10, 90] → mean = 50

Same polygon and raster; different inclusion rule. Red outline = polygon; dots = cell centres.

Choose a Statistic That Matches the Values

Measurements can be averaged. Category labels should be counted.

CONTINUOUS MEASUREMENTS

10	12
18	20

(10 + 12 + 18 + 20) / 4

Mean = 15

CATEGORICAL LABELS

1	1
2	2

1 = class A
2 = class B

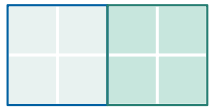
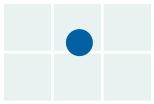
SHARE OF THE FOUR CELLS

A: 50%	B: 50%

Count category labels. An average code of 1.5 does not identify a meaningful class.

Four Spatial Relationships, Four Verified Outputs

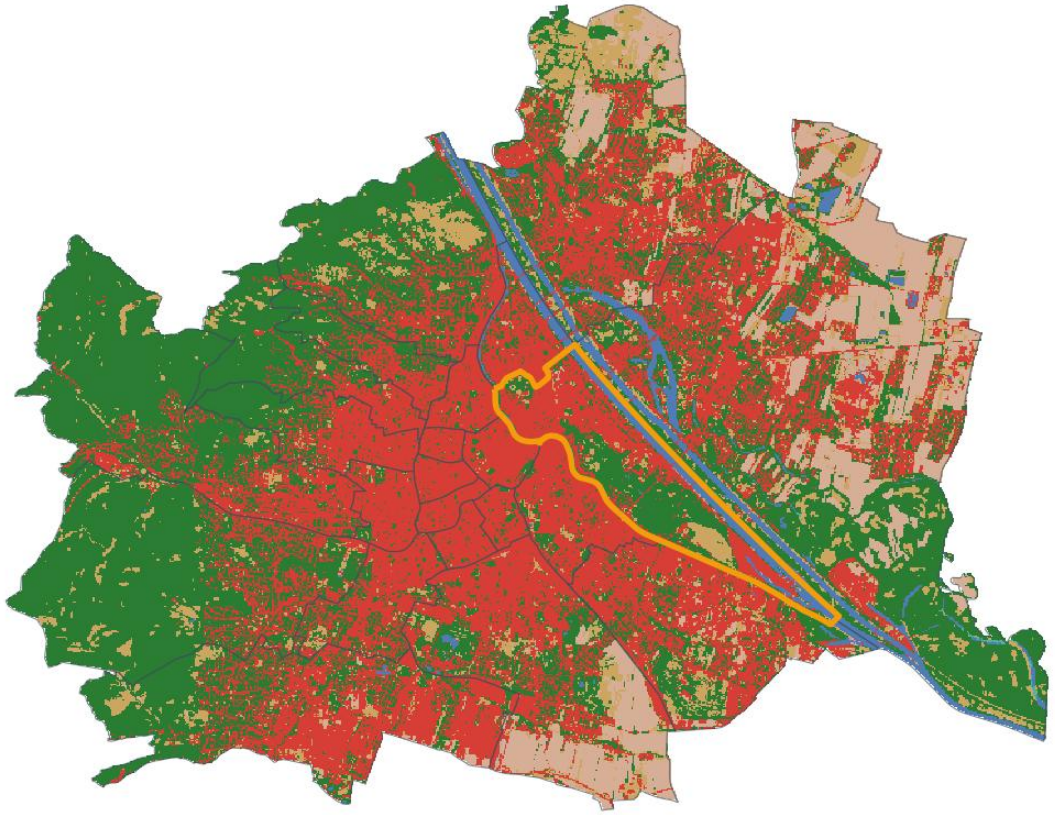
Identify the relationship first, then choose the operation and its checks.

	RELATIONSHIP / OPERATION	VERIFIED OUTPUT	CHECK BEFORE COMPUTING
	**Mosaic** Adjacent raster tiles	**2 × 4 array**	Edges + grid alignment
	**Difference** Matching raster cells	**14 valid cells**	Same grid + valid inputs
	**Sampling** Point ↔ raster	**P2 = 30**	CRS + containing cell
	**Zonal summary** Polygon ↔ raster	**Z1 mean ≈ 13.33**	Selected cells + validity

Check compatibility → Compute → Verify → Interpret

How Much of Leopoldstadt Is Classified as Built-up?

Use the same polygon-raster workflow with real categorical data.



LEOPOLDSTADT / DISTRICT 2

50 = Built-up

COUNT THE TARGET CLASS

$$\frac{\text{Built-up pixels}}{\text{All valid district pixels}}$$

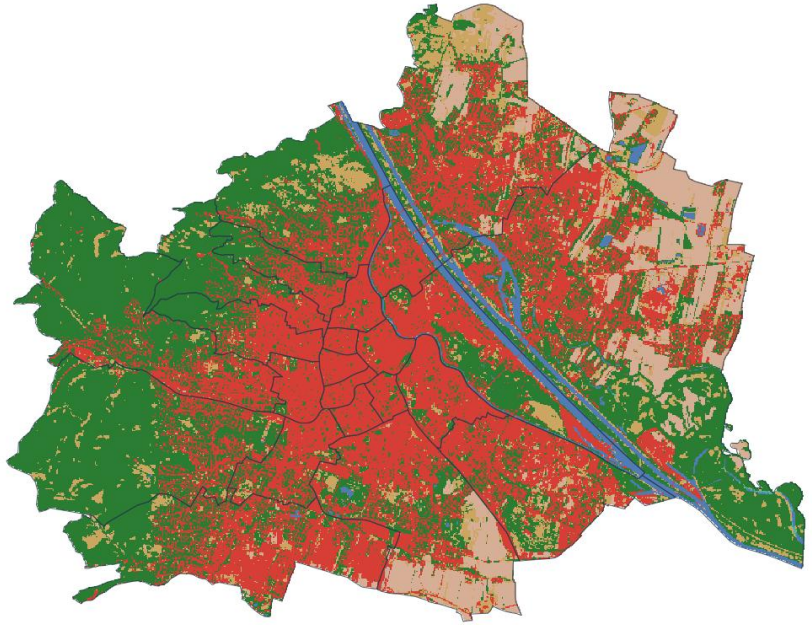
Then express this ratio as a percentage.

Source: ESA WorldCover 2021 and Vienna district boundaries, retained from the course materials.

Two Datasets with Two Different Roles

The raster supplies the class. The polygon defines the area to summarize.

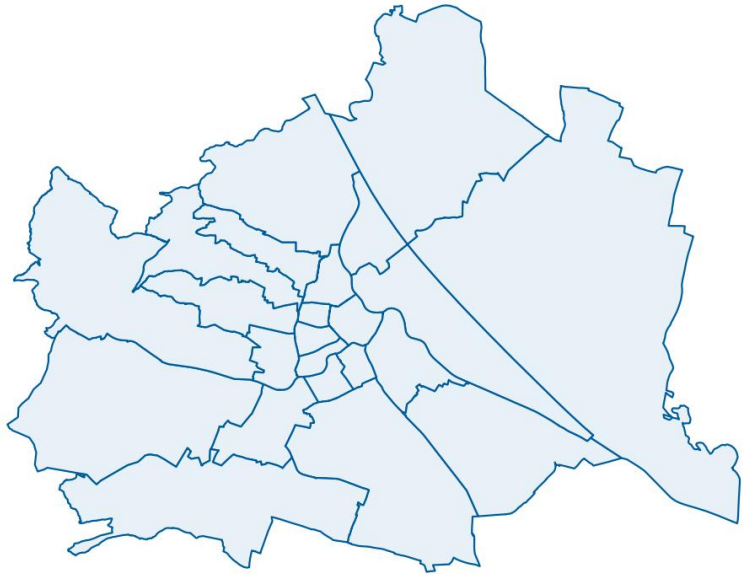
RASTER: WHAT IS IN EACH CELL?



W05_Vienna_WorldCover2021_10m.tif

+

VECTOR: WHICH AREA IS OURS?



W05_Vienna_Districts_23.geojson

We count raster categories only within the selected district polygon.

Inspect the Raster Before Analysis

Read the file properties instead of assuming the synthetic-example settings.



Blue = valid raster footprint

Property	Prepared course raster
CRS	EPSG:31256
Cell size	10 × 10 m
Bands	1
NoData	**0**

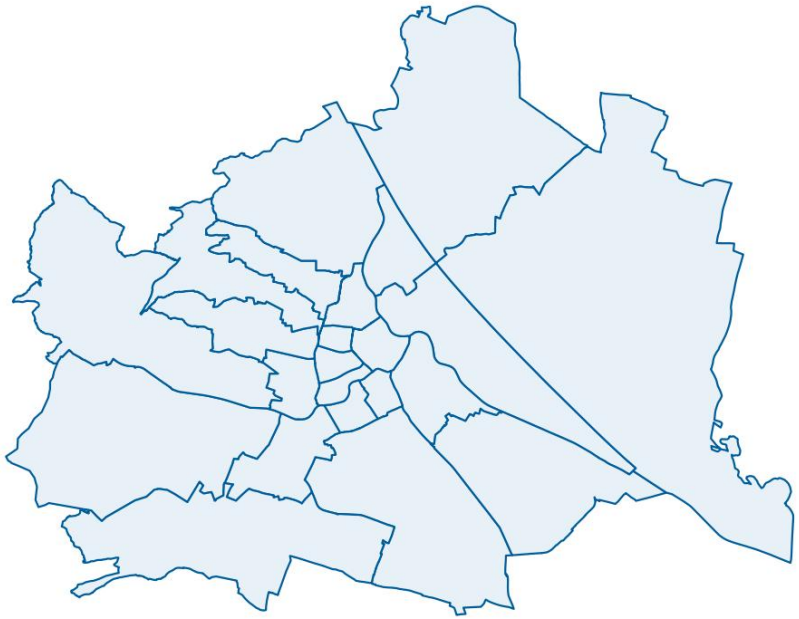
Use this file's NoData value: 0.

```
with rasterio.open(raster_path) as src:
    print(src.crs, src.res, src.nodata)
```

Metadata and valid footprint retained from the prepared course WorldCover raster.

Inspect the District Layer Before Selecting a Zone

One polygon feature corresponds to one row in the attribute table.



district_id	district_name
1	Innere Stadt
2	**Leopoldstadt**
3	Landstraße
...	...
23	Liesing

23 polygon features

Source CRS: EPSG:4326

Use the district ID to choose a feature; do not assume its row position identifies it.

Vienna district attributes and boundaries retained from the course deck.

Transform the Districts into the Raster CRS

Change the polygon coordinates; keep the raster values unchanged.

districts

EPSG:4326

longitude / latitude

to_crs()

districts_aligned

EPSG:31256

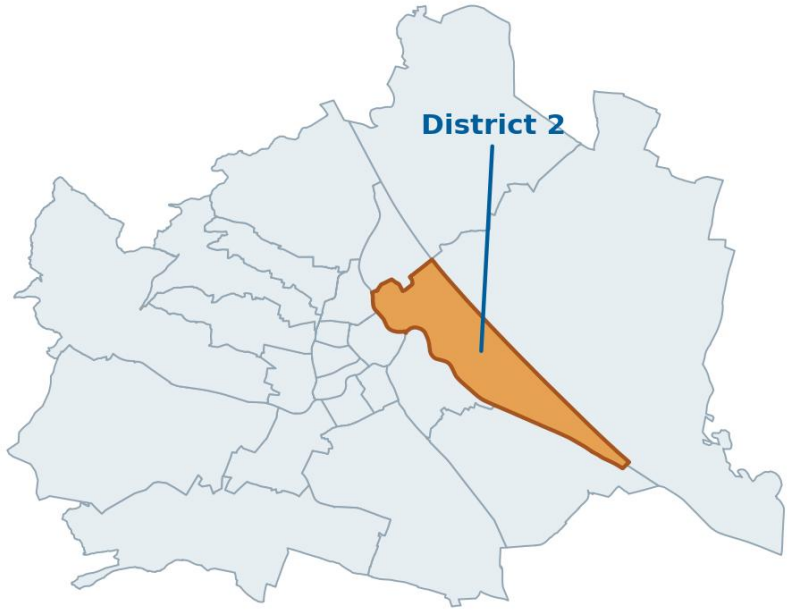
Now matches the raster CRS

```
with rasterio.open(raster_path) as src:
    raster_crs = src.crs
districts_aligned = districts.to_crs(raster_crs)
```

Use districts_aligned in all later polygon selections and raster overlays.

Select Leopoldstadt by Its District ID

Filter the aligned layer first; then take the selected feature.



1 Filter district_id == 2

district_id	district_name
2	Leopoldstadt

2 Take the selected feature

```
chosen.iloc[0]
```

One selected row, not row 0 of all districts.

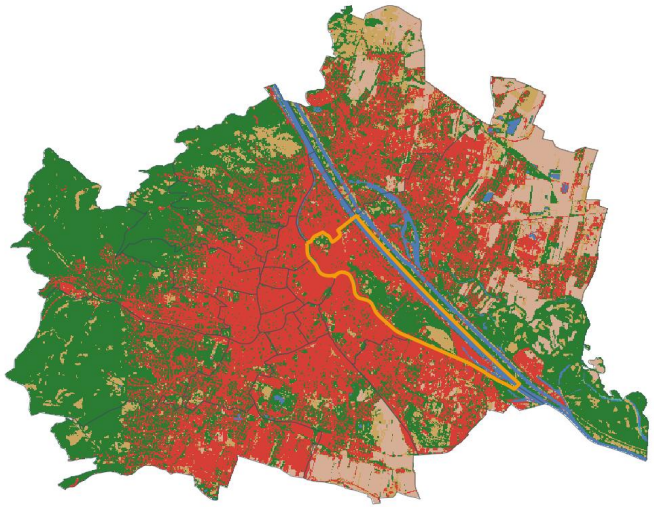
```
is_district_2 = districts_aligned["district_id"] == 2
chosen = districts_aligned[is_district_2]
guided_district = chosen.iloc[0]
```

Selected district boundary retained from the course deck.

Select the District Pixels, Then Crop the Extent

The output is rectangular, but cells outside the polygon remain masked.

CITY RASTER + SELECTED DISTRICT



CROPPED RECTANGLE



Outside the polygon stays masked

```
with rasterio.open(raster_path) as src:
    guided_clip, guided_transform = mask(
        src, [guided_district.geometry],
        crop=True, filled=False)
```

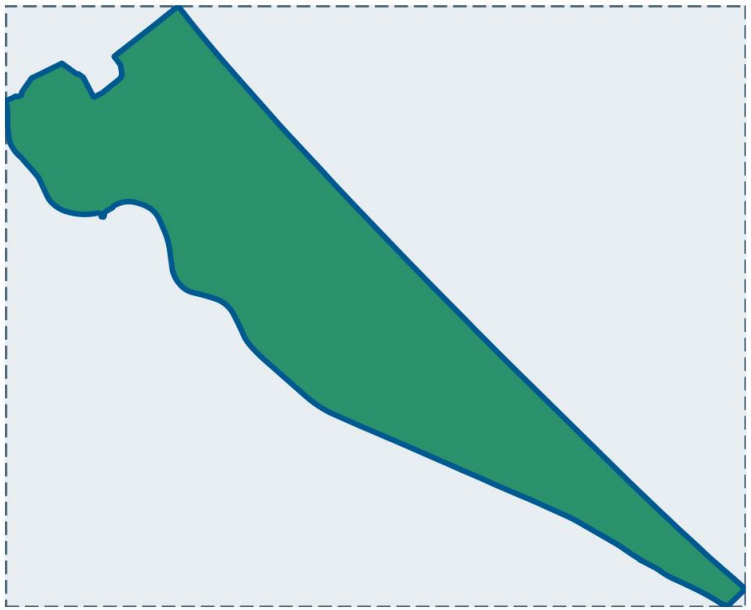
ESA WorldCover 2021 and the selected Vienna district, retained from the course materials.

A Cropped Rectangle Is Not the District

Cropping defines the output extent; the polygon defines which pixels belong to the district.

CROP

rectangular output extent



POLYGON

valid district pixels

KEEP

inside polygon + valid class

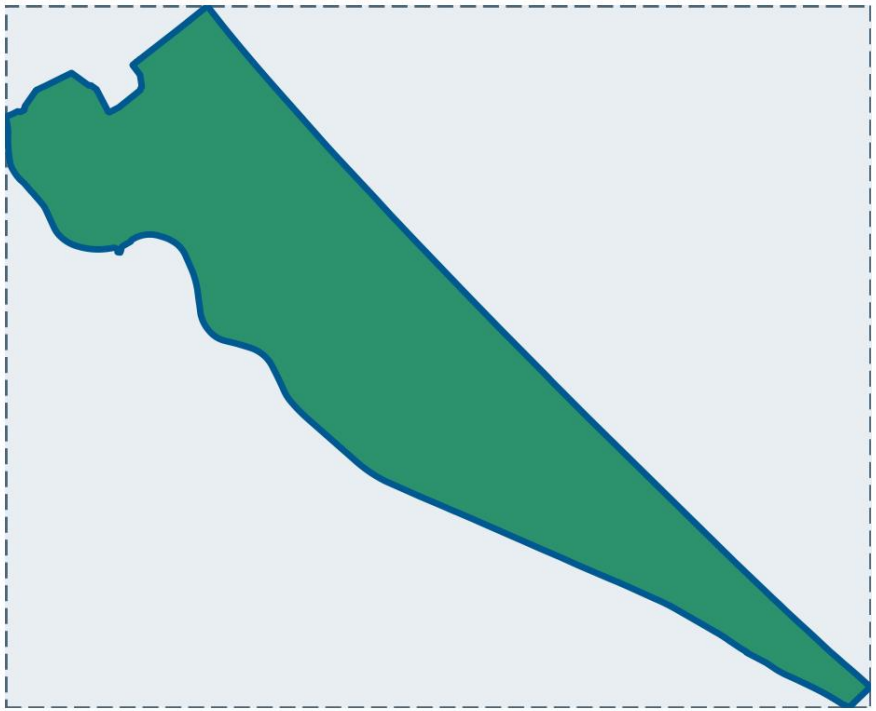
EXCLUDE

outside polygon or NoData

Denominator = valid classified pixels inside the polygon.

Which Pixels Belong in the Denominator?

Count valid classified cells inside the district, not all cells in the crop.



EXCLUDE

Outside the polygon or NoData

KEEP

Valid classified district pixels

192,427

This is the denominator.

```
guided_values = guided_clip[0].compressed()
valid_count = guided_values.size
```

Valid-pixel footprint and count retained from the prepared course datasets.

Calculate Leopoldstadt's Built-up Share

Divide the class 50 count by the valid-pixel count, then express it as a percentage.

```
valid_count = guided_values.size
builtup_count = np.count_nonzero(guided_values == 50)
builtup_pct = 100 * builtup_count / valid_count
```

76,935

Built-up pixels (class 50)

192,427

All valid district pixels

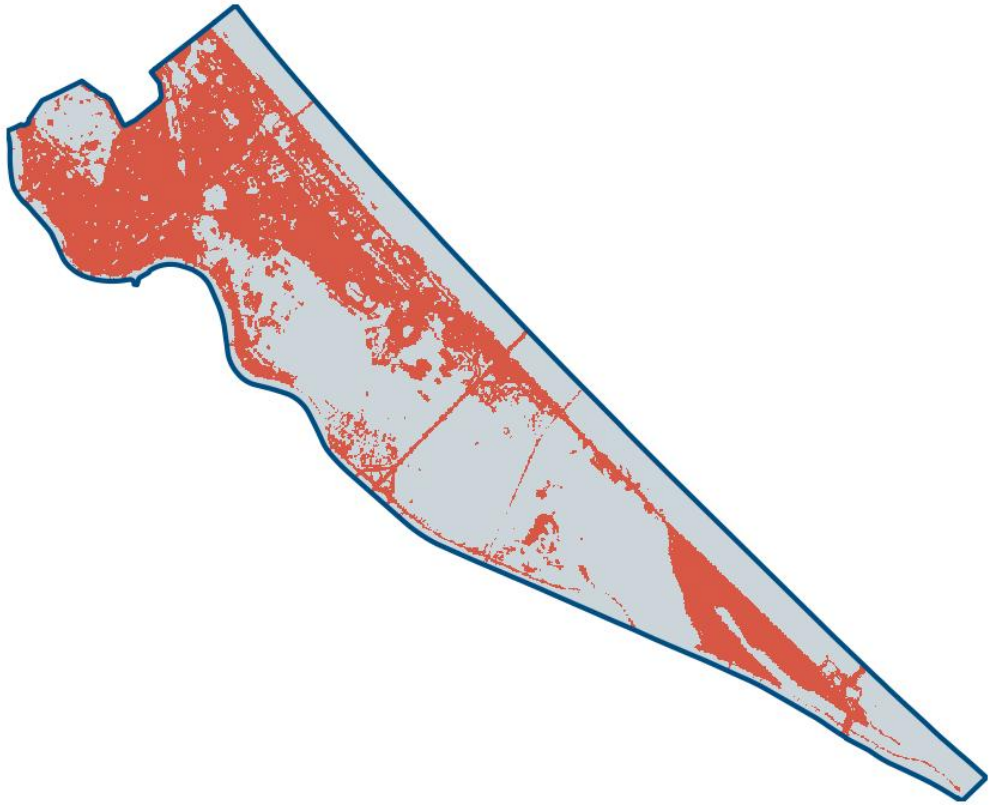
≈ **0.3998** → **39.98%**

as a percentage

39.98% of valid Leopoldstadt pixels are classified as Built-up in WorldCover 2021.

Read the Map Behind the Percentage

The percentage tells us how much; the map shows where.



39.98%

of valid district pixels

- Class 50: Built-up
- Other valid classes
- NoData / outside

Where are Built-up pixels concentrated within the district?

This is a land-cover class share, not the exact building footprint area.

Prepared Leopoldstadt map from the course deck. ESA WorldCover 2021 and Vienna district boundary.

Independent Checkpoint: Tree Cover Across 23 Districts

Change the district set and class code; keep the same workflow.

	Guided example	Independent checkpoint
Districts	Leopoldstadt (District 2)	**All 23 districts**
Target class	50: Built-up	**10: Tree cover**

1 Calculate

One result row per district

2 Verify

Recalculate at least one district

3 Interpret

Explain high and low tree-cover shares

Plotting is optional; calculation, verification and interpretation are the required parts.

Midterm: Format and Rules

Weeks 1–5. Paper exam.

50	×	2	=	100
single-answer MCQs		points per question		points in total

2 hours

40% of the course grade

Early hand-in is allowed.

No phones or other electronic devices.

PRACTICE QUIZ

20 MINUTES

10 MULTIPLE-CHOICE QUESTIONS

UNGRADED · FOR PRACTICE ONLY

LAB TIME