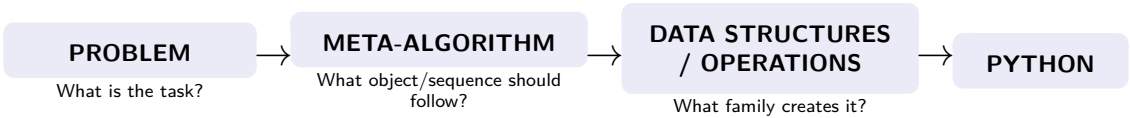
Python for Finance: In-class correction (Session 3)

Amedeo Andriollo
Finance Department (DRM)
Université Paris Dauphine - PSL

From problem to program



- We compare *plans* before code: before reaching the outputs determine the intermediate steps.
- A meta-algorithm traces how the tasks is decomposed

↪ Exact implementations come after translation: from your language to Python language.

Exercise A: objective, inputs, and required outputs

What you are asked to do

- Apply the reimbursement policy to each claim in the correct rule order.
- Store one result for every claim so that each individual decision can be inspected later.
- Keep the claim-level results in the same order as the original input.

Data provided

- Five aligned sequences containing claim ID, employee, category, amount, and receipt information.
- One lookup that gives the maximum reimbursable amount for each valid category.

What you should produce

- One ordered result record for each claim, including its status and reimbursable amount.
- The total reimbursement and number of rejected claims for each employee.
- The employee with the largest total reimbursement.
- An ordered list of the claim IDs that were capped or rejected and therefore require review.

Exercise A: the decision before the loop

1. Is the amount valid? If not, reject.
2. Is the category known? If not, reject.
3. Is a required receipt present? If not, reject.
4. If valid, does the category limit reduce the claim?
5. Return exactly one status and one reimbursement amount.

Why this order matters

Claim C106 has a negative amount. It is rejected by the first rule; later cap logic must never turn invalid input into a payable claim.

! Do not print inside the decision: later stages need values they can store, group and check.

↪ Decide one record correctly before repeating the decision nine times.

Exercise A: build once, summarise once

1. Check equal lengths and unique claim IDs.
2. Produce one four-field result record per claim, in input order.
3. Build employee totals and rejection counts.
4. Derive the top employee and review IDs from completed structures.
5. Reconcile claim-level and employee-level totals.

Route A: stream

Decide a claim, append its result, and update employee accumulators immediately.

or

Route B: stage

Normalise records, evaluate all records, group completed results, then summarise.

Exercise A: solution checks and common mistakes

Checks to confirm the solution is correct

- The number of processed results must equal the number of input claims.
- The final classifications must contain 3 approved, 3 capped, and 3 rejected claims.
- Total reimbursement across all claims must equal EUR 688.
- The sum of the claim-level reimbursements must equal the sum of the employee-level totals.
- Claims flagged for review must appear in the same order as in the original input.

Common mistakes to avoid

- Writing one long processing loop before clearly defining the rule for evaluating a single claim.
- Adding a claim to employee totals before deciding whether it is approved, capped, or rejected.
- Printing a result instead of storing it in a list or dictionary that can be used later.
- Using a set to store review claim IDs when their original order needs to be preserved.

Exercise B: objective, inputs, and required outputs

What you are asked to do

- Compare delivery reliability across regions and service types.
- Compute reliability using parcel counts: sum the on-time parcels and total recorded parcels within each group, then divide.
- For Express service, examine daily regional performance relative to a 92% reference level.

Data provided

- One 224-row CSV containing delivery observations and some unusable parcel-total values.
- One four-row lookup table that assigns each depot to a region.

What you should produce

- A four-row summary with one observation for each region–service combination.
- A 28×2 table of daily Express reliability, with dates as rows and regions as columns.
- The number of days each region falls below the 92% reference level.
- One figure showing the two daily regional Express reliability series.

$$\text{on-time rate} = \frac{\text{recorded parcels} - \text{late parcels}}{\text{recorded parcels}}$$

Exercise B: clean before calculating

1. Inspect shape, content and inferred types.
2. Recognise that `not_recorded` means the denominator is unavailable.
3. Parse dates and represent the marker as missing numeric data.
4. Exclude only rows whose parcel denominator is unavailable.

Row-count evidence

224 raw rows − 6 unavailable denominators = 218 analysis rows.

! Replacing an unavailable denominator by zero does not repair information: it creates an undefined rate and changes the meaning of the record.

- `driver_hours` is inspected as context but is not needed for this question.

Exercise B: merge the region and check the result

218 delivery observations with depot IDs → **4-row table depot ID → region** → **218 delivery observations with region added**

- Many delivery observations can belong to same depot, but each depot should map to one region only.
- After the merge, check that 1) the number of observations is still 218 and 2) that every observation has a region.
- Avoid manually assigning regions with separate conditions for D01, D02, D03, and D04.
  The table keeps the mapping in one place and is easier to check and update.
- A merge that runs without an error is not enough:
  ↪ Verify that rows were not unexpectedly duplicated and that every depot found a matching region.

Exercise B: sum parcel counts before the rates

How to calculate the reliability rate

1. Within each region–service or date–region group, sum the total number of recorded parcels.
2. Sum the number of late parcels, or equivalently compute and sum the number of on-time parcels.
3. Calculate one reliability rate from these grouped totals.

Error: averaging rates/percentage

Suppose one row contains 10 parcels with 100% delivered on time, while another contains 90 parcels with 80% delivered on time.

$$\text{avg. two row rates} = \frac{100\% + 80\%}{2} = 90\%,$$

$$\text{reliability rate} = \frac{10 \times 1 + 90 \times 0.8}{10 + 90} = 82\%.$$

↪ The two rows represent very different numbers of parcels, so they should not receive equal weight.

- Averaging row-level percentages would incorrectly treat the 10-parcel row and the 90-parcel row as equally important.
- The correct procedure is therefore: **sum the parcel counts first, then the rate**.

Exercise B: prepare the daily Express series

1. Start from the cleaned dataset, but keep only rows where service == `"express"`.
2. For each date and region, sum the total recorded parcels and the on-time parcels.
3. Use these grouped tots to calculate, for each date-region combination, the reliability rate (=tot on-time parcels/tot recorded parcels).
4. Reshape the results so that dates are in the rows and regions are in separate columns (28x2 table).
5. Plot one labelled reliability line for each region and add the horizontal 92% reference line.

Exercise B: solution checks and common mistakes

Checks to confirm the solution is correct

- The final daily Express table should have 28 dates and 2 regional columns, with no missing or infinite values.
- Every reliability rate between 0 and 1.
- The number of days below the 92% reference level should be 0 for North and 10 for South.
- The numbers of observations across the region–service groups should add up to the 218 rows in the cleaned analysis dataset.
- The final figure should contain two regional reliability lines and one horizontal 92% reference line.

Common mistakes to avoid

- Averaging row-level percentages instead of summing parcel counts first (see previous slide).
- Plotting the original depot-level observations instead of one aggregated reliability value for each date and region.
- Calculating rates before identifying and removing rows with an unavailable parcel denominator.
- Assuming the depot-to-region merge is correct without checking.
- Counting individual depot observations below 92% instead of counting aggregated region-days below the reference level.

Exercise B: two valid ways

Route A: keep the data in long format

Clean the row-level dataset, attach the region information, group the observations, calculate the reliability rate, and reshape the data only when building the final daily table.

- Individual observations remain easy to inspect throughout the workflow.
- The steps follow the natural sequence: clean, merge, group, calculate, then reshape.

Route B: build separate date-by-region tables

Create one date-by-region table for total parcels and another for late parcels, then use the two aligned tables to calculate daily reliability.

- The parcel totals used in the daily calculation are visible directly in the two tables.
- The separate region-by-service summary still needs to be produced with its own grouping step.

Both routes are valid.

↪ Open `Session3_InClass_Solutions.ipynb`.