Task Description

The programming task is to compute from sales data how often product combinations of any two products are bought within the same basket. The sales data consists of a CSV file with multiple baskets, where each basket contains at least one product that was bought. Details about the data format can be found in the README file for the example data.

An example of a file with 3 baskets:

basket_id	product_id
123	1
123	2
456	1
456	2
456	3
789	1

Which should result in the following output of the implemented algorithm:

product_1	product_2	# baskets
1	2	2
2	3	1
1	3	1

I.e. both products 1 and 2 were bought in the two baskets 123 and 456, which is why the result for this combination is two. The product combination 1 and 3, as well as 2 and 3, appear only in basket 456, and therefore their result is one. The basket 789 contains only one product and therefore does not change the result at all.

Some further notes about the task:

- the order of the rows in the result output doesn't matter.
- the results for product combinations like (1, 2) and (2, 1) are always identical. For the solution, it does not matter if both are included, or only one of them.
- a product will appear at most once in each basket.
- each basket will only have a small number of products, and all products of a basket will appear within consecutive lines of the input file.

Task Restrictions

The main challenge will be that we work with the assumption that **neither the input file nor the output file fit into memory**. Therefore the calculation will imply some intermediate steps:

- somehow split the dataset into multiple smaller datasets.
- compute intermediate results for each of the smaller datasets.
- combine the intermediate results to create the final result.

Technical notes:

- to simplify the setup and development here we will only simulate the memory constraints by running the program with very little allocated memory. Therefore we can test also with rather small datasets.
- test data can be generated with the generate_dataset.py script, which will create a random dataset for a given scale.
- the task is not about distributing the computation so the intermediate results should be calculated in a sequential loop (instead of parallelizing the computation).

General notes:

- Please implement the solution for the task in Python.
- Please do not use any libraries that already solve the problem, but implement the algorithm yourself. It's fine to e.g. use libraries that help reading/writing csv files though.

Task Details

The following subtasks are meant as a guide for how the task could be solved. You do not have to follow them step by step if you find a different way to solve the task.

Subtask 1: Find a good way of splitting the data into multiple datasets.

- generate a dataset with scale 1 using the Python script from the data.tar.gz archive (python generate_dataset.py --scale 1) that can be used to test the implementation (alternatively you can also work with the data_example.csv.gz file).
- find a good way to structure the data into intermediate datasets, considering the computation task.
- write a program that splits the dataset into multiple smaller files.
- for the documentation: explain the chosen approach for splitting the dataset.

Subtask 2: Calculate the co-occurrences for products.

- implement the algorithm that computes the product co-occurrences from the intermediate files created in subtask 1.

Subtask 3: Documentation.

- write a short explanation of how to run the program.
- explain why it computes the correct result.
- explain why it works within the memory constraints.
- explain how you would productionize your solution.

Bonus

Bonus tasks:

- generate datasets with different scales, and measure the runtime of your algorithm. Does it behave as expected?
- the baskets in the example data are generated randomly, and therefore each product combination appears in approximately the same number of baskets. Would the algorithm still work if this wasn't the case, and some product combinations would occur much more often than others?
- how would the problem change if we are also interested in the occurrence of combinations of three products within a basket?