Evolving Pipelines:

Understand how **genetic algorithms** can be used to construct transformation pipelines that solve a target-driven text task.

Recap: pipeline9.py

- Before today, you worked twice with a script (pipeline9.py) that applies a **fixed sequence of transformations**.
  - What was the goal of that script? *The pipeline tried to...*
- Was there any evolution or optimization in pipeline9.py?
  ☐ Yes ☐ No → If yes, how?

Understanding pipelineOptimizer.py

This script tries to **evolve** a transformation pipeline that produces a **target string** (e.g., "mutter").

- What are the components of an Individual in this script?
  ☐ A string ☐ A list of functions ☐ A dictionary of parameters
- What operations simulate "evolution"?
  ☐ Sorting ☐ Mutation and crossover ☐ Manual selection
- What role does the fitness function play here?

Comparison and Reflection

- What is the main **difference** between pipelineOptimizer.py and pipelineOptimizer_deap.py in terms of structure?

  The first script implements evolution by...
  The second script...
- Do you think DEAP makes it easier or harder to understand what's going on? Why?

  I think...

- Think back to the use of **stemmata** (tree-like structures for modeling textual transmission).
  Could **evolutionary methods** (as in these scripts) also apply to cultural transmission?

Maybe in cases where versions blend or "cross"...