Basic Mutation and Tree Construction
Relevant Scripts: tree1.py, tree2.py

Objectives:

- Understand how mutations are modeled.
- Observe how string ancestry is recorded in a tree.
- Begin identifying the idea of a "leading error" (Leitfehler).

Tasks:

1.Run tree1.py and examine the tree output.
    - What is the role of the error_rate? Try with 0.0, 0.1, 0.5.
    - How does the mutation affect string diversity?
    - Are repeated errors traceable in the tree?
2. Compare with tree2.py, which introduces elimination.
    - What is different when elimination_rate > 0?
    - Does the tree still preserve full ancestry?
    - What happens to branches that die early?

Reflection Questions:

- Can you find a sequence that occurs in multiple branches? What does this suggest?
- What would a "Leitfehler" look like in such a tree?

Selection by Fitness
Relevant Script: tree3.py

Objectives:

- Understand how fitness is calculated from string diversity.
- Explore the effect of fitness-based elimination.

Tasks:

1.Inspect the fitness() function in TreeNode.
    - What makes a sequence "fit"?
    - Which strings are likely to survive?
2. Run simulations with different error_rate and elimination_rate values.
    - What happens when mutations are rare? Frequent?
    - How does string variety change over generations?

Reflection Questions:

- How does selection influence which mutations survive?
- Could a "Leitfehler" survive elimination if it increases fitness?

Population Limits and Uniqueness
Relevant Scripts: tree4.py, tree5.py

Objectives:

- Explore the impact of population limits.
- Understand the concept of enforcing unique sequences.

Tasks:

1. Run tree4.py with a low max_population.
    - What kind of nodes are eliminated?
    - What happens if all survivors are very similar?
2. Run tree5.py and analyze the caching mechanism.
    - What is the role of the sequence cache?
    - Try to trigger a cache conflict (hint: low error_rate).

Reflection Questions:

- Does preventing duplicate sequences alter the notion of a leading error?
- Is this similar to manuscript contamination (copying from multiple sources)?

Redefining Fitness
Relevant Scripts: tree6.py, tree7.py, tree7a.py

Objectives:

- Examine different fitness functions.
- Consider how selection pressure can be designed.

Tasks:

1.Look at the fitness() functions in each script.
    - tree6.py: Focus on count of 'b'.
    - tree7.py: Sum of letter pair differences.
    - tree7a.py: Weighted distance and penalties.
2. Run simulations with identical parameters in each script.
    - Which strings tend to dominate?
    - How does the fitness definition change the outcome?

Reflection Questions:

- Which mutations are reinforced by each fitness strategy?
- How do these compare to historical transmission of errors?
- Do all fitness advantages correspond to meaningful features (in texts)?