Report on String Similarity Matching Program

Introduction

The goal of this project was to develop a Python program that can compare two strings of 6-10 characters, calculate the percentage similarity, and generate a detailed match report. The program also performs alignment to improve the comparison between the strings.

Objectives

- Accept two strings of 6-10 characters from the user.
- Calculate similarity percentage using multiple approaches:
  - Levenshtein distance
  - Alignment-based matching
- Generate a match report showing matches, mismatches, and gaps.
- Provide a visual alignment of the strings.

Methodology

Input Validation: Ensures strings are between 6-10 characters.

Similarity Calculation:

- Levenshtein Distance: Calculates edits needed to transform one string to another; similarity (%) = (1 - distance/max_length) * 100.
- Alignment-Based Matching: Uses a global alignment approach (match=1, mismatch=0, gap=-1) to generate aligned strings and matches.

Match Report Generation: Shows aligned strings, matches, mismatches, gaps, and match visualization.

Optional: difflib.SequenceMatcher ratio provided as a reference.

Implementation Details

- Language: Python 3.x
- Dependencies: Python standard library only (difflib)
- Platform: Windows, Linux, macOS compatible
- Algorithm: Dynamic programming for Levenshtein and global alignment
- File structure:
  string_similarity.py - Main script
  README.md - Instructions
  Report_String_Similarity.pdf - This report

Sample Run

```
Input:
  String A: kitten
  String B: sitting
Output (excerpt):
  Aligned strings:
    kitten-
    ||| |
    sitting
  Matches: 4
  Alignment length: 7
  Similarity (matches/max_len): 57.14%
  Levenshtein similarity: 57.14% (distance = 3)
```

Assumptions

- Strings must be 6-10 characters
- Case-insensitive comparison by default
- Alignment scoring: match=1, mismatch=0, gap=-1
- Optimized for short strings

Conclusion

The program calculates string similarity with multiple methods, provides a detailed match report, and demonstrates fundamental string matching algorithms using dynamic programming.