# How can I calculate the Levenshtein Distance in Python?

Authored by

**stats writer**

April 23, 2024

The Levenshtein Distance is a measure of the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. In Python, the distance can be calculated using the built-in library function "editdistance". This function takes two strings as parameters and returns the Levenshtein Distance as an integer. The calculation is based on the dynamic programming approach and is efficient for large strings. It is commonly used in applications such as spell checkers, text similarity analysis, and DNA sequencing. By implementing the Levenshtein Distance in Python, users can easily compare and measure the similarity between two strings, providing a useful tool for various applications.

## Calculate Levenshtein Distance in Python

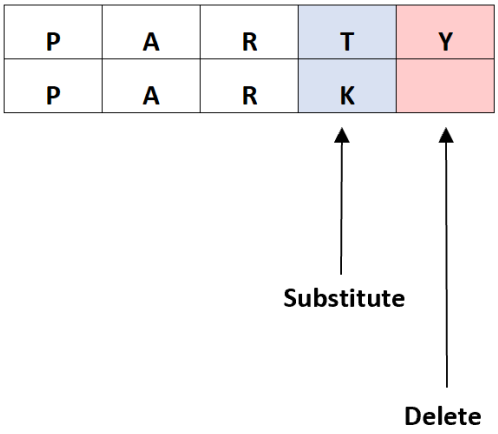
## The Levenshtein distance between two strings is the minimum number of single-character edits required to turn one word into the other.

## The word "edits" includes substitutions, insertions, and deletions.

## For example, suppose we have the following two words:

## PARTYPARK

## The Levenshtein distance between the two words (i.e. the number of edits we have to make to turn one word into the other) would be 2:






**In practice, the Levenshtein distance is used in many different applications including approximate string matching, spell-checking, and natural language processing.**

**This tutorial explains how to calculate the Levenshtein distance between strings in Python by using the python-Levenshtein module.**

**You can use the following syntax to install this module:**

**pip install python-Levenshtein**

**You can then load the function to calculate the Levenshtein distance:**

```
from Levenshtein import distance as lev
```

The following examples show how to use this function in practice.

### Example 1: Levenshtein Distance Between Two Strings

The following code shows how to calculate the Levenshtein distance between the two strings "party" and "park":

```
#calculate Levenshtein distance
lev('party', 'park')

2
```

The Levenshtein distance turns out to be 2.

### Example 2: Levenshtein Distance Between Two Arrays

The following code shows how to calculate the Levenshtein distance between every pairwise combination of strings in two different arrays:

```
#define arrays
a =
```

**b <-**

**#calculate Levenshtein distance between two arraysfor i,k in zip(a, b):**
**print(lev(i, k))**

**6**
**4**
**5**
**5**

**The way to interpret the output is as follows:**

**The Levenshtein distance between 'Mavs' and 'Rockets' is 6.The Levenshtein distance between 'Spurs' and 'Pacers' is 4.The Levenshtein distance between 'Lakers' and 'Warriors' is 5.The Levenshtein distance between 'Cavs' and 'Celtics' is 5.**