Aim: Using the inbuild python libraries for string matching using fuzzy logic.

In [1]:
```
#!pip install fuzzywuzzy
```

```
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
```

In [2]:
```
#!pip install python-levenshtein
```

```
Collecting python-levenshtein
  Downloading python_Levenshtein-0.20.9-py3-none-any.whl (9.4 kB)
Collecting Levenshtein==0.20.9
  Downloading Levenshtein-0.20.9-cp39-cp39-win_amd64.whl (101 kB)
     ---------------------------------------- 101.3/101.3 kB ? eta 0:00:00
Collecting rapidfuzz<3.0.0,>=2.3.0
  Downloading rapidfuzz-2.13.7-cp39-cp39-win_amd64.whl (1.0 MB)
     ---------------------------------------- 1.0/1.0 MB 64.1 MB/s eta 0:00:0
0
Installing collected packages: rapidfuzz, Levenshtein, python-levenshtein
Successfully installed Levenshtein-0.20.9 python-levenshtein-0.20.9 rapidfuzz
-2.13.7
```

In [4]:
```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
```

In [13]:
```
strA = 'ChandanGupta is a lifesaver'
strB = 'chandan gupta is a LIFE SAVER'
```

In [14]:
```
ratio = fuzz.ratio(strA.lower(),strB.lower())
print(ratio)
```

```
96
```

In [15]:
```
ratio2 = fuzz.ratio(strA,strB)
print(ratio2)
```

```
57
```

Partial Ratio:

In [26]:
```
strA = 'Mumbai, Maharashtra'
strB = 'Mumbai'
pratio = fuzz.partial_ratio(strA,strB)
ratio3 = fuzz.ratio(strA,strB)
print(pratio)
print(ratio3)
```

```
100
48
```

In [25]:
```
strA = 'Neil Nitin Mukesh'
strB = 'Nitin'
pratio = fuzz.partial_ratio(strA,strB)
ratio4 = fuzz.ratio(strA,strB)
print(pratio)
print(ratio4)
```

```
100
45
```

In [24]:
```
strA = 'Lalu Prasad Yadav'
strB = 'Prasad Lalu Yadav'
tsratio = fuzz.token_sort_ratio(strA,strB)
ratio5 = fuzz.ratio(strA,strB)
print(tsratio)
print(ratio5)
```

```
100
71
```

Fuzzywuzzy library provides fuzz API which is very useful in doing string matching for applications like NLP, it provides us with variety of string matching functions that can be used in different scenarios

1. Ratio_Function: It finds out the similarity ratio between the two strings using Levenshtein distance formula.
2. Partial_Ratio_Functio: It is used to perform sub-string matching, this function is typically useful while matching people's names.
3. Token_Sort_Ratio_Function: It sorts the strings alphabetically and then do matching on the alphabetically sorted version of the strings.

Compare the following strings using ratio, partial_ratio and token_sort_ratio functions.

In [45]:
```
strA = 'truth or dare'
strB = 'truth dare'
ratio = fuzz.ratio(strA.lower(),strB.lower())
ratio6 = fuzz.ratio(strA,strB)
print(ratio)
print(ratio6)
```

```
87
87
```

In [44]:
```
strA = 'Truth Or Dare'
strB = 'Truthdare'
ratio = fuzz.ratio(strA,strB)
ratio9 = fuzz.ratio(strA,strB)
print(ratio)
print(ratio9)
```

```
73
73
```

In [31]:
```
strA = 'truth or dare'
strB = 'truth dare'
pratio = fuzz.partial_ratio(strA,strB)
ratio10 = fuzz.ratio(strA,strB)
print(pratio)
print(ratio10)
```

```
70
87
```

In [32]:
```
strA = 'Raj Kapoor'
strB = 'Ranbir Kapoor'
pratio = fuzz.partial_ratio(strA,strB)
ratio11 = fuzz.ratio(strA,strB)
print(pratio)
print(ratio11)
```

```
70
78
```

In [33]:
```
strA = 'truth or dare'
strB = 'truth or dare!'
pratio = fuzz.partial_ratio(strA,strB)
ratio7 = fuzz.ratio(strA,strB)
print(pratio)
print(ratio7)
```

```
100
96
```

In [29]:
```
strA = 'truth or dare'
strB = 'dare or truth'
tsratio = fuzz.token_sort_ratio(strA,strB)
ratio8 = fuzz.ratio(strA,strB)
print(tsratio)
print(ratio8)
```

```
100
46
```

In [34]:
```
strA = 'truth or dare'
strB = 'or dare truth'
tsratio = fuzz.token_sort_ratio(strA,strB)
ratio12 = fuzz.ratio(strA,strB)
print(tsratio)
print(ratio12)
```

```
100
54
```

Token Set Ratio

In [48]:
```
S1 = 'truth or dare'
S2 = 'truth or or dare'
tsratio1 = fuzz.token_sort_ratio(S1,S2)
tsratio2 = fuzz.token_set_ratio(S1,S2)
print(tsratio1)
print(tsratio2)
```

```
90
100
```

In [49]:
```
strA = 'Neil Nitin'
strB = 'Neil Nitin Nitin'
tsratio1 = fuzz.token_sort_ratio(strA,strB)
tsratio2 = fuzz.token_set_ratio(strA,strB)
print(tsratio1)
print(tsratio2)
```

```
77
100
```

In [52]:
```
str1 = 'I Love My Car'
str2 = 'I am loving my car'
WRatio = fuzz.WRatio(str1,str2)
print(WRatio)
```

```
77
```

In [53]:
```
str1 = 'truth or dare'
str2 = 'Truth Or Dare'
WRatio = fuzz.WRatio(str1,str2)
print(WRatio)
```

```
100
```

In [54]:
```
str1 = 'truth or dare!!!'
str2 = 'truth or dare'
WRatio = fuzz.WRatio(str1,str2)
print(WRatio)
```

```
100
```

In [55]:
```
str1 = 'truth or dare!!!'
str2 = 'truth or dares'
WRatio = fuzz.WRatio(str1,str2)
print(WRatio)
```

```
96
```

WRatio is a string matching function that has the ability to ignore uppercase, lowercase and also alphNumeric characters.

In [62]:
```
query = 'artificial intelligence'
choices = ['artificial intelligence','Artificial Intelligence', 'arts intellig
print('List of ratios:')
print(process.extract(query,choices),'\n')
print('Best among the above list:', process.extractOne(query,choices))
```

```
List of ratios:
[('artificial intelligence', 100), ('Artificial Intelligence', 100), ('a inte
lligence', 86), ('arts intelligence', 80)]

Best among the above list: ('artificial intelligence', 100)
```

In [ ]:
```
```