Python String Processing Cheatsheet

Visit KDnuggets.com for more cheatsheets and additional Data Science, Machine Learning, AI & Analytics learning resources

Stripping Whitespace

Strip leading whitespace with `lstrip()`, trailing whitespace with `rstrip()`, both with `strip()`.

```
s = '   A sentence with whitespace.   \n'

>>> print('{}'.format(s.lstrip()))

>>> print('{}'.format(s.rstrip()))

>>> print('{}'.format(s.strip()))
```

To strip characters other than whitespace, pass in the character(s) you want stripped.

```
>>> print('{}'.format(s.rstrip('A')))
```

Splitting Strings

Split strings into lists of smaller substrings with `split()`.

```
s = 'KDnuggets is a fantastic resource'

>>> print(s.split())
['KDnuggets', 'is', 'a', 'fantastic', 'resource']
```

Other character(s) sequences can be passed.

```
s = 'these,words,are,separated,by,comma'

>>> print('{}'.format(s.split(',')))
['these', 'words', 'are', 'separated', 'by', 'comma']
```

Joining List Elements Into a String

Join list element strings into single string in Python using `join()`.

```
s = ['KDnuggets', 'is', 'a', 'fantastic', 'resource']

>>> print(' '.join(s))
KDnuggets is a fantastic resource
```

Join list elements with something other than whitespace in between ("and" in this example).

```
s = ['Eleven', 'Mike', 'Dustin', 'Lucas', 'Will']

>>> print(' and '.join(s))
Eleven and Mike and Dustin and Lucas and Will
```

Reversing a String

Strings can be sliced like lists. Reversing one can be done in the same way as a list's elements.

```
s = 'KDnuggets'

>>> print('The reverse of KDnuggets is {}'.format(s[::-1]))
The reverse of KDnuggets is: steggunDK
```

Converting Uppercase and Lowercase

Converting between cases can be done with `upper()`, `lower()`, and `swapcase()`.

```
s = 'KDnuggets'

>>> print('{}'.format(s.upper()))
KDNUGGETS

>>> print('{}'.format(s.lower()))
kdnuggets

>>> print('{}'.format(s.swapcase()))
kdNUGGETS
```

Checking for String Membership

Check for string membership using the `in` operator.

```
s1 = 'perpendicular'
s2 = 'pen'
s3 = 'pep'

>>> print('\'pen\' in \'perpendicular\': {}'.format(s2 in s1))
'pen' in 'perpendicular': True

>>> print('\'pep\' in \'perpendicular\': {}'.format(s3 in s1))
'pep' in 'perpendicular': False
```

Find the location of a substring with `find()` (-1 means not present).

```
s = 'Does this string contain a substring?'

>>> print('\'string\' location -> {}'.format(s.find('string')))
'string' location -> 10

>>> print('\'spring\' location -> {}'.format(s.find('spring')))
'spring' location -> -1
```

Replacing Substrings

Replace substrings with replace().

```
s1 = 'The theory of data science is of the utmost importance.'
s2 = 'practice'

>>> print('{}'.format(s1.replace('theory', s2)))
The practice of data science is of the utmost importance.
```

Combining the Output of Multiple Lists

Combine multiple lists in an element-wise fashion with `zip()`.

```
countries = ['USA', 'Canada', 'UK', 'Australia']
cities = ['Washington', 'Ottawa', 'London', 'Canberra']

>>> for x, y in zip(countries, cities):
>>>   print('The capital of {} is {}.'.format(x, y))
The capital of USA is Washington.
The capital of Canada is Ottawa.
...
```

Checking for Anagrams

Check for anagrams by counting, comparing letter occurrences.

```
from collections import Counter
def is_anagram(s1, s2):
  return Counter(s1) == Counter(s2)

>>> print('listen is an anagram of silent ->
   {}'.format(is_anagram('listen', 'silent')))
listen is an anagram of silent -> True
```

Checking for Palindromes

Check for palindromes by reversing a word and then using ==.

```
def is_palindrome(s):
   reverse = s[::-1]
   if (s == reverse):
      return True
   return False

>>> print('racecar is a palindrome ->
   {}'.format(is_palindrome('racecar')))
racecar is a palindrome -> True
```

Matthew Mayo, 2022