
You are currently looking at **version 1.0** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the [Jupyter Notebook FAQ \(https://www.coursera.org/learn/python-text-mining/resources/d9pwm\)](https://www.coursera.org/learn/python-text-mining/resources/d9pwm) course resource.

Working With Text

```
In [1]: text1 = "Ethics are built right into the ideals and objectives of the United Nations "  
len(text1) # The length of text1
```

Out[1]: 76

```
In [2]: text2 = text1.split(' ') # Return a list of the words in text2, separating by ' '.  
len(text2)
```

Out[2]: 14

```
In [3]: text2
```

```
Out[3]: ['Ethics',  
        'are',  
        'built',  
        'right',  
        'into',  
        'the',  
        'ideals',  
        'and',  
        'objectives',  
        'of',  
        'the',  
        'United',  
        'Nations',  
        '']
```

List comprehension allows us to find specific words:

```
In [4]: [w for w in text2 if len(w) > 3] # Words that are greater than 3 letters long in text2
```

```
Out[4]: ['Ethics',  
        'built',  
        'right',  
        'into',  
        'ideals',  
        'objectives',  
        'United',  
        'Nations']
```

```
In [5]: [w for w in text2 if w.istitle()] # Capitalized words in text2
```

```
Out[5]: ['Ethics', 'United', 'Nations']
```

```
In [6]: [w for w in text2 if w.endswith('s')] # Words in text2 that end in 's'
```

```
Out[6]: ['Ethics', 'ideals', 'objectives', 'Nations']
```

We can find unique words using `set()`.

```
In [7]: text3 = 'To be or not to be'
        text4 = text3.split(' ')

        len(text4)
```

```
Out[7]: 6
```

```
In [8]: len(set(text4))
```

```
Out[8]: 5
```

```
In [9]: set(text4)
```

```
Out[9]: {'To', 'be', 'not', 'or', 'to'}
```

```
In [10]: len(set([w.lower() for w in text4])) # .lower converts the string to lowercase.
```

```
Out[10]: 4
```

```
In [11]: set([w.lower() for w in text4])
```

```
Out[11]: {'be', 'not', 'or', 'to'}
```

Processing free-text

```
In [12]: text5 = '"Ethics are built right into the ideals and objectives of the United Nations" \
        #UNSG @ NY Society for Ethical Culture bit.ly/2guVelr'
        text6 = text5.split(' ')

        text6
```

```
Out[12]: ['"Ethics',
          'are',
          'built',
          'right',
          'into',
          'the',
          'ideals',
          'and',
          'objectives',
          'of',
          'the',
          'United',
          'Nations"',
          '#UNSG',
          '@',
          'NY',
          'Society',
          'for',
          'Ethical',
          'Culture',
          'bit.ly/2guVelr']
```

Finding hastags:

```
In [13]: [w for w in text6 if w.startswith('#')]
```

```
Out[13]: ['#UNSG']
```

Finding callouts:

```
In [14]: [w for w in text6 if w.startswith('@')]
```

```
Out[14]: ['@']
```

```
In [15]: text7 = '@UN @UN_Women "Ethics are built right into the ideals and objectives of the United Nations  
#UNSG @ NY Society for Ethical Culture bit.ly/2guVelr'  
text8 = text7.split(' ')
```

We can use regular expressions to help us with more complex parsing.

For example '@[A-Za-z0-9_]+' will return all words that:

- start with '@' and are followed by at least one:
- capital letter ('A-Z')
- lowercase letter ('a-z')
- number ('0-9')
- or underscore ('_')

```
In [16]: import re # import re - a module that provides support for regular expressions  
[w for w in text8 if re.search('@[A-Za-z0-9_]+', w)]
```

```
Out[16]: ['@UN', '@UN_Women']
```

```
In [ ]:
```

```
In [ ]:
```