*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) 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 [7]:

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

Out[7]:

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

In [4]:

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

Out[4]:

```
['Ethics', 'United', 'Nations']
```

In [5]:

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

Out[5]:

```
['Ethics', 'ideals', 'objectives', 'Nations']
```

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

In [6]:

```
text3 = 'To be or not to be'
text4 = text3.split(' ')

len(text4)
```

Out[6]:

```
6
```

In [7]:

```
len(set(text4))
```

Out[7]:

```
5
```

In [8]:

```
set(text4)
```

Out[8]:

```
{'To', 'be', 'not', 'or', 'to'}
```

In [9]:

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

Out[9]:

```
4
```

In [10]:

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

Out[10]:

```
{'be', 'not', 'or', 'to'}
```

In [11]:

```
text = 'hello there\
how are you!'
len(text)
```

Out[11]:

```
23
```

In [12]:

```
text.upper()
```

Out[12]:

```
'HELLO THEREHOW ARE YOU!'
```

In [13]:

```
'hello' in text
```

Out[13]:

```
True
```

In [17]:

```
'h' in 'hello'
```

Out[17]:

```
True
```

In [14]:

```
text = '"hello"'
len(text)
```

Out[14]:

```
7
```

In [15]:

```
s = 'Hello There'
print(s.isalpha())  # comprise of alphabets only (False because of space).
print(s.istitle())
s = 'qwerty'; str = 'qwerty1234'
print(s.isalpha())
print(str.isdigit())
print(str.isalnum())  # alphanumeric
```

```
False
True
True
False
True
```

In [16]:

```
s = '-';
seq = ('a', 'b', 'c');    # This is sequence of strings.
print(s.join(seq))
```

```
a-b-c
```

In [17]:

```
s = '    There you are. ';
s.strip()
```

Out[17]:

```
'There you are.'
```

In [18]:

```
s.rstrip()     # removes whitespaces from end.
```

Out[18]:

```
'    There you are.'
```

In [19]:

```
s = 'abracaadabra'
print(s.find('bra'))    # finds first match from the start of the string.
print(s.rfind('bra'))   # finds first match from the end of the string.
```

```
1
9
```

In [20]:

```
s.replace('aa','a')
```

Out[20]:

```
'abracadabra'
```

In [21]:

```
s = 'ouagadougou'
text = s.split('ou')
text
```

Out[21]:

```
['', 'agad', 'g', '']
```

In [22]:

```
'ou'.join(text)
```

Out[22]:

```
'ouagadougou'
```

Getting characters out of words.

In [23]:

```
list(s)
```

Out[23]:

```
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

In [24]:

```
[c for c in s]
```

Out[24]:

```
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

In [25]:

```
f = open('dates.txt','r')
f.seek(0)
text = f.readline();   # reads one line.
text
```

Out[25]:

```
'03/25/93 Total time of visit (in minutes):\n'
```

In [26]:

```
text.rstrip()     # to get rid of '\n' which is a whitespace.
```

Out[26]:

```
'03/25/93 Total time of visit (in minutes):'
```

In [27]:

```
f.seek(0)         # f.seek(n) in general.
text = f.read();  # f.read(n) reads n characters.
print(text)
len(text)         # no. of characters in file.
```

```
03/25/93 Total time of visit (in minutes):
6/18/85 Primary Care Doctor:
sshe plans to move as of 7/8/71 In-Home Services: None
7 on 9/27/75 Audit C Score Current:
2/6/96 sleep studyPain Treatment Pain Level (Numeric Scale): 7
.Per 7/06/79 Movement D/O note:
4, 5/18/78 Patient's thoughts about current substance abuse:
10/24/89 CPT Code: 90801 - Psychiatric Diagnosis Interview
3/7/86 SOS-10 Total Score:
(4/10/71)Score-1Audit C Score Current:
(5/11/85) Crt-1.96, BUN-26; AST/ALT-16/22; WBC_12.6Activities of Daily Liv
ing (ADL) Bathing: Independent
4/09/75 SOS-10 Total Score:
8/01/98 Communication with referring physician?: Done
1/26/72 Communication with referring physician?: Not Done
5/24/1990 CPT Code: 90792: With medical services
1/25/2011 CPT Code: 90792: With medical services
4/12/82 Total time of visit (in minutes):
1; 10/13/1976 Audit C Score, Highest/Date:
4, 4/24/08 Relevant Drug History:
```

In [28]:

```
f.seek(9)
f.read(5)
```

Out[28]:

```
'Total'
```

In [29]:

```
text1 = text.splitlines()
print(len(text1))
print(text1[0])
```

```
500
03/25/93 Total time of visit (in minutes):
```

In [30]:

```
f.close()
```

In [31]:

```
f.closed
```

Out[31]:

```
True
```

Processing free-text

In [32]:

```
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[32]:

```
['"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 [33]:

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

Out[33]:

```
['#UNSG']
```

Finding callouts:

In [34]:

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

Out[34]:

```
['@']
```

In [35]:

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

In [36]:

```
text8
```

Out[36]:

```
['@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']
```

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 [37]:

```
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)]     # it will also match abc@xyz as th
```

Out[37]:

```
['@UN', '@UN_Women']
```

In [38]:

```
[w for w in text8 if re.search(r'@[A-Za-z0-9_]+', w)]
```

Out[38]:

```
['@UN', '@UN_Women']
```

In [39]:

```
text = 'ouagadougou'
re.findall(r'[aeiou]',text)
```

Out[39]:

```
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']
```

In [40]:

```
re.findall(r'[^aeiou]',text)
```

Out[40]:

```
['g', 'd', 'g']
```

In [41]:

```
datestr = '23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 Oct 2002\n\
23 October 2002\nOct 23, 2002\nOctober 23, 2002\n'
datestr
```

Out[41]:

```
'23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 Oct 2002\n23 October 2002
\nOct 23, 2002\nOctober 23, 2002\n'
```

In [42]:

```
dates = datestr.splitlines()
dates
```

Out[42]:

```
['23-10-2002',
 '23/10/2002',
 '23/10/02',
 '10/23/2002',
 '23 Oct 2002',
 '23 October 2002',
 'Oct 23, 2002',
 'October 23, 2002']
```

In [43]:

```
date = '23-10-2002\n'      # '\n' is a one-caracter string containing a newline.
len(date)
```

Out[43]:

```
11
```

In [44]:

```
re.findall(r'\d{1,2}[-/]\d{1,2}[-/]\d{2,4}',datestr)
```

Out[44]:

```
['23-10-2002', '23/10/2002', '23/10/02', '10/23/2002']
```