In [1]:

```
# Inter-language translation is a challenging problem in natural language processing and
# artificial intelligence. With advances in machine learning, artificial intelligence ar
# language processing, services like Google Translate (100+ languages) and Microsoft Bi
# Translator (60+ languages) can translate between languages instantly.

# Inter-language translation also is great for people traveling to foreign countries. Th
# can use translation apps to translate menus, road signs and more. There are even effor
# live speech translation so that you'll be able to converse in real time with people wh
# not know your natural language.11,12 Some smartphones, can now work together with inea
# headphones to provide near-live translation of many languages.

# The TextBlob library uses Google Translate to detect a text's language and translate
# TextBlobs, Sentences and Words into other languages.16 Let's use detect_language
# method to detect the language of the text we're manipulating ('en' is English):

import textblob
from textblob import TextBlob
text = 'Today is a beautiful day. Tomorrow looks like bad weather.'
blob = TextBlob(text)
blob
```

Out[1]:

```
TextBlob("Today is a beautiful day. Tomorrow looks like bad weather.")
```

In [2]:

```
# Next, let's use the translate method to translate the text to Spanish ('es') then
# detect the language on the result. The to keyword argument specifies the target langua

spanish = blob.translate(to='es')
```

In [3]:

```
spanish
```

Out[3]:

```
TextBlob("Hoy es un hermoso dia. Mañana parece mal tiempo.")
```

In [5]:

```
# Next, let's translate our TextBlob to simplified Chinese (specified as 'zh' or 'zh-CN
# then detect the language on the result:

chinese = blob.translate(to='zh')
```

In [6]:

```
chinese
```

Out[6]:

```
TextBlob("今天是美好的一天。明天看起来像恶劣天气。")
```

In [7]:

```
# In each of the preceding cases, Google Translate automatically detects the source lang
# You can specify a source language explicitly by passing the from_lang keyword
# argument to the translate method, as in:

chinese = blob.translate(from_lang='en', to='zh')
```

In [8]:

```
# Calling translate without arguments translates from the detected source language to
# English:

spanish.translate()
```

Out[8]:

```
TextBlob("Today is a beautiful day. Tomorrow looks bad weather.")
```

In [9]:

```
chinese.translate()
```

Out[9]:

```
TextBlob("Today is a beautiful day. It looks like bad weather tomorrow.")
```

In [10]:

```
# Note the slight difference in the English results.
```

In [11]:

```
# Inflections are different forms of the same words, such as singular and plural (like '
# and “people”) and different verb tenses (like “run” and “ran”). When you’re calculati
# word frequencies, you might first want to convert all inflected words to the same for
# more accurate word frequencies. Words and WordLists each support converting words to
# their singular or plural forms. Let’s pluralize and singularize a couple of Word obje

from textblob import Word
index = Word('index')
```

In [12]:

```
index.pluralize()
```

Out[12]:

```
'indices'
```

In [14]:

```
cacti = Word('cacti')
cacti.singularize()
```

Out[14]:

```
'cactus'
```

In [15]:

```
# Pluralizing and singularizing are sophisticated tasks which, as you can see above, are
# not as simple as adding or removing an “s” or “es” at the end of a word.
# You can do the same with a WordList:

from textblob import TextBlob
animals = TextBlob('dog cat fish bird').words
```

In [16]:

```
animals.pluralize()
```

Out[16]:

```
WordList(['dogs', 'cats', 'fish', 'birds'])
```

In [17]:

```
# For natural language processing tasks, it’s important that the text be free of spellin
# Software packages for writing and editing text, like Microsoft Word, Google Docs and
# others automatically check your spelling as you type and typically display a red line
# misspelled words. Other tools enable you to manually invoke a spelling checker.
# You can check a Word’s spelling with its spellcheck method, which returns a list of
# tuples containing possible correct spellings and a confidence value. Let’s assume we n
# to type the word “they” but we misspelled it as “theyr.” The spell checking results sh
# two possible corrections with the word 'they' having the highest confidence value:

from textblob import Word
word = Word('theyr')
%precision 2
```

Out[17]:

```
'%.2f'
```

In [18]:

```
word.spellcheck()
```

Out[18]:

```
[('they', 0.57), ('their', 0.43)]
```

In [19]:

```
# Note that the word with the highest confidence value might not be the correct word for
# the given context.
# TextBlobs, Sentences and Words all have a correct method that you can call to correct
# spelling. Calling correct on a Word returns the correctly spelled word that has the
# highest confidence value (as returned by spellcheck):

word.correct() # chooses word with the highest confidence value
```

Out[19]:

```
'they'
```

In [20]:

```
# Calling correct on a TextBlob or Sentence checks the spelling of each word. For each
# incorrect word, correct replaces it with the correctly spelled one that has the highes
# value:

from textblob import Word
sentence = TextBlob('Ths sentense has missplled wrds.')
sentence.correct()
```

Out[20]:

```
TextBlob("The sentence has misspelled words.")
```