Introduction to Python: Worksheet

Trevor Cohn

Euromasters Summer School 2005

Python is installed on the DICE workstations (i.e. those found in Appleton Tower). Login using your allocated account and password, then launch a Terminal. You can either run Python by either typing `python2.3` or `idle2.3` at the command prompt. The former launches Python within the current terminal, while the later will open a new window and also allows editing of files and debugging.

You should use either a standard Unix editor (such as emacs, vim or pico) or use idle in order to create and edit source files.

If you wish to install Python and/or NLTK on your own machine, you will find distributions online. Currently, the latest release of Python is 2.4.1, and is available from `http://www.python.org`. You can also retrieve NLTK from `http://nltk.sourceforge.net` – installation instructions are given at `http://nltk.sourceforge.net/install.html`.

Basic Python

Introductory material on Python is available at `http://www.python.org/doc/Intros.html`. Specifically, the *Python Tutorial* is well worth working through. It can be found at `http://docs.python.org/tut/tut.html`.

The following examples concern lists and strings. Please try them out, and experiment with variations. You will need to do this from inside IDLE:

```
>>> a = ['colourless', 'green', 'ideas']
>>> a
['colourless', 'green', 'ideas']
>>> len(a)
3
>>> a[1]
'green'
>>> a[1:]
['green', 'ideas']
>>> b = a + ['sleep', 'furiously']
>>> b
['colourless', 'green', 'ideas', 'sleep', 'furiously']
>>> b[-1]
'furiously'
>>> b.sort()
>>> b
['colourless', 'furiously', 'green', 'ideas', 'sleep']
>>> b.reverse()
>>> b
['sleep', 'ideas', 'green', 'furiously', 'colourless']
>>> b[2] + b[1]
'greenideas'
>>> for w in b:
...     print w[0]
...
's'
'i'
```

```
'g'
'f'
'c'
>>> b[2][1]
'r'
>>> b.index('green')
2
>>> b[5]
IndexError: list index out of range
>>> b[0] * 3
'sleepsleepsleep'
>>> c = ' '.join(b)
>>> c
'sleep ideas green furiously colourless'
>>> c.split('r')
['sleep ideas g', 'een fu', 'iously colou', 'less']
>>> map(lambda x: len(x), b)
[5, 5, 5, 9, 10]
>>> [(x, len(x)) for x in b]
[('sleep', 5), ('ideas', 5), ('green', 5), ('furiously', 9), ('colourless', 10)]
```

Experiment further with lists and strings until you are comfortable with using them. Next we'll take a look at Python *dictionaries* (or associative arrays).

```
>>> d = {}
>>> d['colourless'] = 'adj'
>>> d['furiously'] = 'adv'
>>> d['ideas'] = 'n'
>>> d.keys()
['colourless', 'furiously', 'ideas']
>>> d.values()
['adv', 'adj', 'n']
>>> d
{'furiously': 'adv', 'colourless': 'adj', 'ideas': 'n'}
>>> d.has_key('ideas')
1
>>> for w in d.keys():
...     print "%s [%s]," % (w, d[w]),
...
furiously [adv], colourless [adj], ideas [n],
>>>
```

Finally, try out Python's regular expression module, for substituting and searching within strings.

```
>>> import re
>>> s = "colourless green ideas sleep furiously"
>>> re.sub('l', 's', s)
'cosoursess green ideas sseep furioussy'
>>> re.sub('green', 'red', s)
'colourless red ideas sleep furiously'
>>> re.findall('[^aeiou][aeiou]', s)
['co', 'lo', 'le', 're', ' i', 'de', 'le', 'fu', 'ri']
>>> re.findall('([^aeiou])([aeiou])', s)
[('c', 'o'), ('l', 'o'), ('l', 'e'), ('r', 'e'), (' ', 'i'), ('d', 'e'),
('l', 'e'), ('f', 'u'), ('r', 'i')]
>>> re.findall('(.).\bs 1', s)
['o', 's']
>>>
```

Please see the Python documentation, for further information on regular expressions:
`http://www.python.org/doc/current/lib/module-re.html`.

Text processing

A small sample from the Penn Treebank is included in the NLTK corpora. This can be found in `/use/share/nltk-data/treebank`. We will look at tokenising these data using Python.

We will use the raw text (in the `raw` subdirectory) and the tagged test (in the `tagged` subdirectory).

1. Write a function to tokenise the the raw text. It should open a raw file (eg. `raw/wsj_0065`) and print out all the tokens contained in the text, one per line. Special care should be taken to deal with punctuation such as commas, periods and dollar symbols. You can start with the following naive implementation which just separates the tokens using whitespace.

   ```
   infile = open('/usr/share/nltk-data/treebank/raw/wsj_0065')
   for line in infile.readlines():
           for token in line.split():
                   print token
   infile.close()
   ```

2. Next, write a function to count the number of occurrences of each word in all of raw files (01 - 99). Display the words in decreasing frequency order. You will probably need to use a dictionary to count the words, and the `sort` and `reverse` methods to reorder the results. The following snippet gives the names of all the raw files:

   ```
   ['/usr/share/nltk-data/treebank/raw/wsj_00%02d' % i for i in range(1, 100)]
   ```

3. Now write a function to tokenise the tagged text, found in the `tagged` subdirectory. This text contains additional markup for part-of-speech and noun-phrase chunking. Here's a snippet from `wsj_0001`:

   ```
   [ Pierre/NNP Vinken/NNP ]
   ,/,
   [ 61/CD years/NNS ]
   old/JJ ,/, will/MD join/VB
   ...
   ```

   The forward slash separates each token from its part-of-speech, and the brackets surround noun-phrase chunks. Your function should ignore the bracketing, printing each token and part of speech on subsequent lines.

4. Finally, use this to determine the frequencies of each POS tag. Which words and parts-of-speech are found more frequently inside a noun-phrase than outside. Does this tally with your intuition? You may want to filter out those words or POS tags that occur very infrequently in the data set, as these will add noise to your results.