Lesson 9: File & Dir Path, Modules, Basic Text Stats

Fundamentals of Text Processing for Linguists

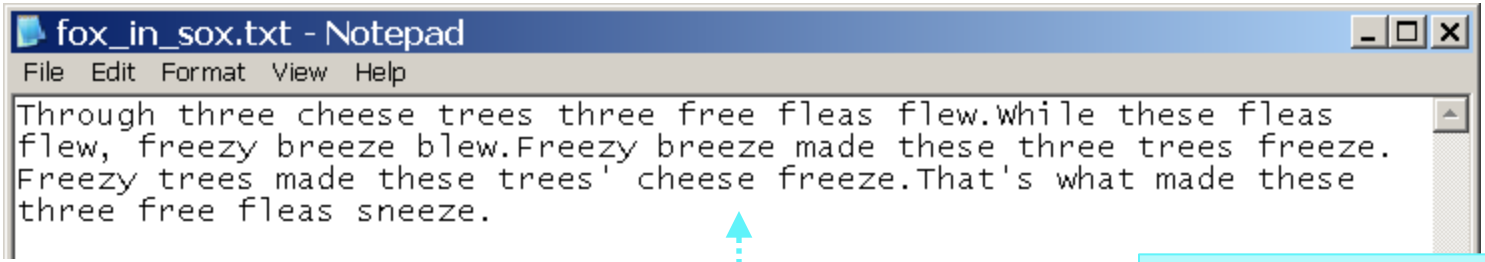
Na-Rae Han

Objectives

- File IO: continued
  - Line breaks
  - File and directory path
  - WD again
- Modules
  - Importing and using standard Python modules
  - Using your own custom modules
- Basic text stats

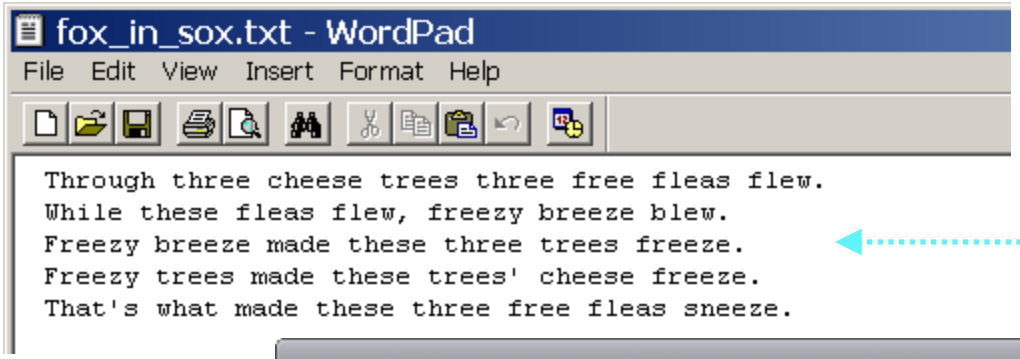
Line breaks

- Opening 'fox_in_sox.txt' in Notepad:



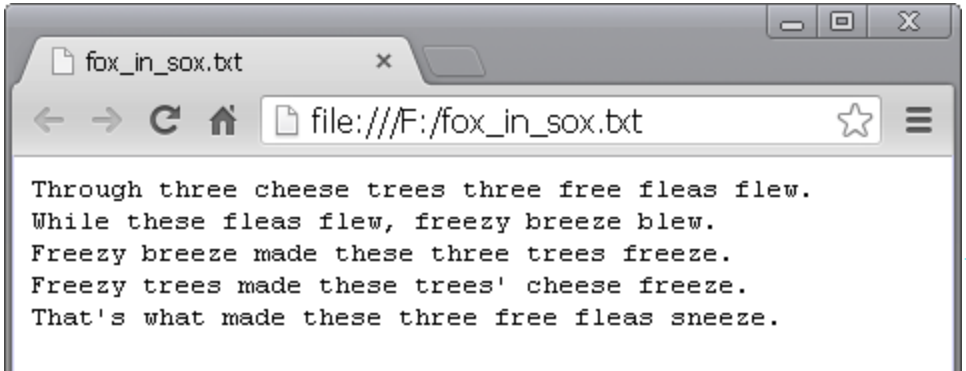
Line breaks are lost in Notepad

- WordPad:



Line breaks are displayed fine in WordPad and Chrome

- Chrome:



Line breaks and Python

- The end-of-line (EOL, also called 'line break') is encoded differently depending on the OS.
  - Unix, Linux, Mac OS X: \n ("line feed")
  - Older Mac OS up to 9: \r ("carriage return")
  - Windows: \r\n
- Python has ***universal newlines support***.
  - Within Python, all newline types are converted to \n.
  - When you write out to a file, Python will replace \n with the OS-appropriate newline marker.
  - If for some reason your Python does not convert the newline marker to \n, you can use the '**rU**' flag instead of 'r' when opening a file for reading.
  - For more information, read:

    http://docs.python.org/2/library/functions.html#open

Absolute file path: Windows

- A reference to a file can include a complete file path including the directory tree hierarchy: **absolute file path**
- Windows:
  - The delimiting character between directories is backlash "\".
  - `test.txt` in my Desktop is referred to as:

```
fname = 'C:\\Users\\naraehan\\Deaktop\\test.txt'
```

cf. `fname = 'C:\Users\naraehan\Deaktop\test.txt'` ✘

new line character '\n' tab character '\t'

  - OR: Use **r ('raw string')** prefix

```
fname = r'C:\Users\naraehan\Deaktop\test.txt'
```

✔

← **r'...'** forces the following string to '...' be interpreted as literal characters

Absolute file path: non-Windows

- A reference to a file can include a complete file path including the directory tree hierarchy: **absolute file path**

- OS-X, Linux, Unix:
  - The delimiting character is **slash "/"**.

    ```
    fname = '/Users/naraehan/Documents/test.txt'
    ```

  - ← Slashes are not special in Python. No need to prefix 'r' to make it a raw string. (It doesn't hurt.)

Relative file path, starting from WD

- A file's location can be specified **relative to WD**.
  - File is located in WD:

```
f = open('fox_in_sox.txt')
```

  - File is located in directory 'data' under WD:

```
f = open(r'data\fox_in_sox.txt')
```

Windows

```
f = open('data/fox_in_sox.txt')
```

OS-X, Linux, Unix

  - File is located "one directory up" from WD (WD's parent directory)

```
f = open(r'..\fox_in_sox.txt')
```

Windows

```
f = open('../fox_in_sox.txt')
```

OS-X, Linux, Unix

File path & WD, a summary

- Referencing a file with its absolute path: `'/Users/naraehan/Documents/fox_in_sox.txt'` always works.

- If referencing with file name only: `'fox_in_sox.txt'`, the file has to be in the current **WD (working directory)**.
  - In a **SCRIPT**:
    - WD is where the script is → file and script should be in the same dir.
    - After your script is executed in IDLE shell, shell's WD changes to the script's location.
  - In **IDLE shell**, your initial WD depends on your setting.
    - Find out WD and change it using the `os` module:
      - `os.getcwd(), os.chdir()`
    - Beware: after running a script, your shell's WD changes to the script's location.

math Module

```
>>> import math
>>> dir(math)
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh',
'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e',
'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
>>> math.sqrt(144)
12.0
>>> help(math.log)
Help on built-in function log in module math:

log(...)
    log(x[, base])

    Return the logarithm of x to the given base.
    If the base not specified, returns the natural logarithm (base e) of x.
>>> math.log(16, 2)
4.0
```

Importing a module

```
>>> import math
>>> math.sqrt(144)
12.0
>>> math.log(16, 2)
4.0
```

```
>>> import os
>>> os.getcwd()
'C:\\Python27'
>>> os.chdir(r'C:\Users\narae\Documents')
>>> os.getcwd()
'C:\\Users\\narae\\Documents'
```

- `math` and `os` are part of python's Standard Library.
- They are included with all Python distributions but are not loaded as built-in types.
- There are 3rd-party modules, which you have to download and install first

Importing a function from a module

```
>>> from math import sqrt
>>> sqrt(144)
12.0
>>> log(16, 2)
Traceback (most recent call last):
    ...
NameError: name 'log' is not defined
```

```
>>> from os import getcwd, chdir
>>> getcwd()
'D:\\Lab'
>>> chdir('text-processing')
>>> getcwd()
'D:\\Lab\\text-processing'
```

- **`from x import y`** statement imports y in the module x into the current namespace.
- If you will be using something from a particular module extensively, importing them this way eliminates the need for module name prefixing.

Surprise: *everything* is a module

- In python, every script can be imported and used as a module.
  - All **functions** and **top-level variable names** become accessible in the caller script.

plural.py

```
irr = {'goose':'geese', 'mouse':
'mice', 'child':'children'}

def getPlural(wd) :
    if wd in irr : return irr[wd]
    else : return wd+'s'
```

test.py

```
import plural

print plural.irr
print plural.getPlural('friend')
```

```
>>> ========== RESTART ==
>>>
{'goose': 'geese', 'mouse': 'mice',
'child': 'children'}
friends
>>>
```

Surprise: *everything* is a module

- In python, every script can be imported and used as a module.
  - EVERYTHING?
  - **A CAVEAT**: a module name can only contain **alphabet letters** and **underscore "_"**.

  ← `get_plural.py` is a good module name.

  ← `Ex5.palindrome.narae.py` is not. It cannot be imported as a module *unless you change the file name*. The "`import …`" statement won't work with the name.

  - By convention, a module should have a **short**, **all-lowercase name**.

Try it out

3 minutes

```
>>> import math
>>> math.sqrt(144)
12.0
>>> math.log(16, 2)
4.0
```

plural.py

```
irr = {'goose':'geese', 'mouse':
'mice', 'child':'children'}

def getPlural(wd) :
    if wd in irr : return irr[wd]
    else : return wd+'s'
```

test.py

```
import plural

print plural.irr
print plural.getPlural('friend')
```

```
>>> ========== RESTART ==
>>>
{'goose': 'geese', 'mouse': 'mice',
'child': 'children'}
friends
>>>
```

Calling a function from a module

pal.py

```
def isPalindrome(wd) :
    # ... (function body clipped)
    if wd == rwd : verdict = True
    else : verdict = False
    return verdict
```

test.py

```
import pal

if pal.isPalindrome('level') :
    print 'level is a palindrome'
```

- `pal.py` is imported as a **module**.

Module name is prefixed to the function name.

Module vs. caller script

pal.py

```
def isPalindrome(wd) :
    # ... (function body clipped)
    return verdict

if isPalindrome('mom') :
    print 'mom is a palindrome'
```

- But if a module script contains its own main process...

test.py

```
import pal

if pal.isPalindrome('level') :
    print 'level is a palindrome'
```

Module vs. caller script

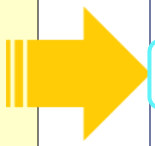
pal.py

```
def isPalindrome(wd) :
    # ... (function body clipped)
    return verdict

if isPalindrome('mom') :
    print 'mom is a palindrome'
```

... importing a module will end up executing the main process as well!

test.py

```
import pal

if pal.isPalindrome('level') :
    print 'level is a palindrome'
```



```
>>>
mom is a palindrome
level is a palindrome
>>>
```

Module vs. caller script: solution (1)

pal.py

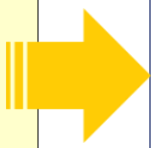
```
def isPalindrome(wd) :
    # ... (function body clipped)
    return verdict

# only function definitions exist
# in this module script
```

- One way to get around this is to keep a module script free of its own main processes.

  ← Not practical.

test.py

```
import pal

if pal.isPalindrome('level') :
    print 'level is a palindrome'
```



```
>>>
level is a palindrome
>>>
```

Module vs. caller script: solution (2)

pal.py

```
def isPalindrome(wd) :
    # ... (function body clipped)
    return verdict

def main() :
    if isPalindrome('mom') :
        print 'mom is a palindrome'
```

- Another is to encapsulate the main part of the module in its own **special function** named **`main()`** …

Module vs. caller script: solution (2)

pal.py

```
def isPalindrome(wd) :
    # ... (function body clipped)
    return verdict

def main() :
    if isPalindrome('mom') :
        print 'mom is a palindrome'

if __name__ == '__main__':
    main()
```

- Another is to encapsulate the main part of the module in its own **special function** named `main()` ...
- And call `main()` only when this script is executed on its own (i.e., not imported as a module from another script).

Often, `main()` contains a process that demonstrates the function usages of the module.

Try it out

5 minutes

- Convert your "past-tense generator script" (HW#2) to be module-ready.
  1. Save the file with an all-lowercase file name: `pastgen.py`.
  2. Neatly package the main process of the script into `main()`
  3. Add the main function routine at the end:

```
if __name__ == '__main__':
    main()
```

  4. Run the module on its own and verify that `main()` executes

- And then, from a different script, try importing the module and calling the functions:
  - `isLikeTap(), getPastTense()`

* If your script has a problem, download and work with Shameek's:
  - http://www.pitt.edu/~naraehan/ling1901/pastgen.py

Congratulations!

- **You have now learned all of essential Python.**

- From this point on, we will focus on:
  - Applying the knowledge to real-world problems
    - How to process text for basic statistics
    - How to work with a corpus
    - How to search through text data
  - Learn a few additional tricks
    - List comprehension
    - Unicode
    - Pickling
    - Running python on command-line

It is time to REVIEW

- We have learned a WHOLE LOT so far.
- First of all, you must ***KNOW*** WHAT WE LEARNED.
  - This is a GOOD time to REVIEW and make sure you have a good command of Python basics.
- Secondly, you must BE ABLE TO ***SYNTHESIZE*** from what you know.
  - As you review the slides, you will find yourself (hopefully) nodding along.
  - That kind of passive knowledge is NOT ENOUGH: you should be able to write the scripts on your own (TRY IT!)
  - And that is NOT ENOUGH: you should be able to apply your coding skills to NOVEL PROBLEMS (TRY IT!)

A COMMON MISCONCEPTION

TRUTH: Nobody, even seasoned programmers, programs solely from memory.

- Rather than trying to commit everything to memory, you should:
  - have a good overall knowledge of programming structure and practices
  - be willing to explore
  - **have your references handy – know how to look stuff up!**

Basic text statistics: type, token

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.

- **Token**:
  - Each word instance in text
  - Symbols are typically separated out through a process called *tokenization*
  - Text size is typically measured in # of tokens
- **Type**:
  - Unique words in text

← Word type *of* has 6 tokens

← Text has 76 tokens and 50 types

Type-token ratio (TTR)

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.

- Text has 76 tokens and 50 types
- TTR (Type-Token Ratio): 50/76 = 0.6578

Type-Token Ratio (**TTR**) is often used as a metric for **vocabulary diversity**.

- CAUTION: TTR figures are only directly comparable across similarly sized texts.
- As a text gets larger, TTR will get naturally lower as more words will get repeated.

Raw vs. relative frequency

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.

Frequency:

- Count of the number of linguistic element (tokens) found in a text
- = raw frequency, raw count

← *of* has a frequency of 6

Relative frequency:

- Frequency count relative to the text size (= total tokens)
- raw freq / text size

← *of* has a relative frequency of 6/76 = 0.07894

Ordered frequency lists

Bible

rank	count	word
1	70573	,
2	64023	the
3	51696	and
4	43808	:
5	34670	of
6	26203	.
7	13580	to
8	12912	that
9	12667	in
10	10419	he
11	10139	;
12	9838	shall
13	8997	unto
14	8971	for
15	8854	I
16	8473	his
17	8177	a
18	7964	lord
19	7376	they
20	7013	be

Alice in Wonderland

rank	count	word
1	2871	'
2	2418	,
3	1642	the
4	988	.
5	872	and
6	729	to
7	632	a
8	595	it
9	553	she
10	545	I
11	514	of
12	462	said
13	450	!
14	411	you
15	398	Alice
16	369	in
17	357	was
18	315	that
19	264	--
20	263	as

- Comparing frequency rankings across texts can highlight differences in authorial style, register and subject field.

HW#3: Two Inaugural Speeches

Washington (1789)

Fellow-Citizens of the Senate and of the House of Representatives:

Among the vicissitudes incident to life no event could have filled me with greater anxieties than that of which the notification was transmitted by your order, and received on the 14th day of the present month. On the one hand, I was summoned by my Country, whose voice I can never hear but with veneration and love, from a retreat which I had chosen with the fondest predilection, and, in my flattering hopes, with an immutable decision, as the asylum of my declining years -- a retreat which was rendered every day more necessary as well as more dear to me by the addition of habit to inclination, and of frequent interruptions in my health to the gradual waste committed on it by time. …

Obama (2009)

My fellow citizens:

I stand here today humbled by the task before us, grateful for the trust you have bestowed, mindful of the sacrifices borne by our ancestors. I thank President Bush for his service to our nation, as well as the generosity and cooperation he has shown throughout this transition.

Forty-four Americans have now taken the presidential oath. The words have been spoken during rising tides of prosperity and the still waters of peace. Yet, every so often the oath is taken amidst gathering clouds and raging storms. At these moments, America has carried on not simply because of the skill or vision of those in high office, but because We the People have remained faithful to the ideals of our forbearers, …

Looking ahead

- **List comprehension**: `[x for x in list]`
  - Filter out elements that does not meet a certain condition
  - Transform each element in list

```
>>> mary = 'Mary had a little lamb'.split()
>>> mary
['Mary', 'had', 'a', 'little', 'lamb']

>>> [w for w in mary if len(w) > 3]
['Mary', 'little', 'lamb']
>>> [w for w in mary if 'a' in w]
['Mary', 'had', 'a', 'lamb']

>>> [w.upper() for w in mary]
['MARY', 'HAD', 'A', 'LITTLE', 'LAMB']
>>> [len(w) for w in mary]
[4, 3, 1, 6, 4]
```

Wrap-up

- **Next class**
  - List comprehension, large text files
- **Homework #3**
  - Two inaugural speeches: Washington vs. Obama
  - http://www.pitt.edu/~naraehan/ling1901/HW3.html
  - 50 points total (previous HWs were 40 points)
  - Due Tuesday midnight after spring break
  - **This one is fairly complex – start early!**
- Unicode poll
  - Chinese, Arabic, Spanish, and what else?