LSA 325
Intro to Computational Lingusistics

Steven Bird

Ewan Klein

Edward Loper

For Help with Programming...

Online Python Tutorials:

How to think like a computer scientist
http://www.ibiblio.org/obp/thinkCSpy/

Non-Programmer's Tutorial for Python
http://en.wikibooks.org/wiki/NonProgrammer%27s_Tutorial_for_Python

Values

- Objects that programs manipulate (I.e., data).
- Examples:
  - `2`
  - `"hello"`
  - `["up", "on", "the", "hill"]`
- Each value has a type.
  - Determines what you can do with the value.
  - Can permit modification (mutable type) or forbid modification (immutable type).

Variables

- Named locations for values.
- Examples: word, words, flargbor
- Variables can have (almost) any name
  - The choice of name doesn't affect the program.
  - Choose meaningful names.
- Variables do *not* have types.
- Assignment statements (x=4) put a new value in the variable.

Frequency Distributions

- A FreqDist is just a histogram.
  - Count how many times each value occurs.
- Construct one of two ways:
  - Incrementally, with a loop:

```
>>> fd = FreqDist()  # Empty!
>>> for word in list_of_words:
...     fd.inc(word)
```

  - Directly from a list of values (e.g., a corpus):

```
>>> fd = FreqDist(list_of_words)
```

Probability Distributions

- NLTK provides tools to convert frequency distributions to probability distributions.

http://nltk.org/doc/api/nltk.probability-module.html

```
>>> nltk.probability.demo()
6 samples (1-6); 500 outcomes were sampled for each FreqDist
=========================================================================
      FreqDist MLEProbD Lidstone HeldoutP HeldoutP CrossVal |  Actual
-------------------------------------------------------------------------
  1   0.102000 0.102000 0.102386 0.116000 0.102000 0.102000 | 0.083333
  2   0.164000 0.164000 0.164016 0.178000 0.164000 0.156667 | 0.166667
  3   0.238000 0.238000 0.237575 0.248000 0.238000 0.248000 | 0.250000
  4   0.240000 0.240000 0.239563 0.220000 0.240000 0.242667 | 0.250000
  5   0.168000 0.168000 0.167992 0.166000 0.168000 0.169333 | 0.166667
  6   0.088000 0.088000 0.088469 0.072000 0.088000 0.081333 | 0.083333
-------------------------------------------------------------------------
Total 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 | 1.000000
=========================================================================
```

Assignment 1 - Example Solution

```
from nltk.corpus import inaugural
from nltk import FreqDist

for item in inaugural.items:
    fd = FreqDist()
    for word in inaugural.tokenized(item):
        if word.lower() in ['man', 'men', 'he']:
            fd.inc('male')
        elif word.lower() in ['woman', 'women', 'she']:
            fd.inc('female')
    print fd['male'], fd['female'], item
```

Functions

- Functions are fixed pieces of code that…
  - Take zero or more values as inputs (*parameters* or *arguments*)
  - Do something with those values
  - Return a value
- We've seen how to use functions:

```
>>> len(words)
23413
>>> word_freqs = FreqDist(words)
>>> word_freqs.max()
'the'
```

Defining New Functions

```
def NAME(ARGUMENTS...):
    STATEMENTS
```

- NAME can be any name you like.
- ARGUMENTS is a list of variable names.
  - The values that are passed to the function will be placed in these variables.
  - These variables are local -- they disappear as soon as the function completes.
- STATEMENTS is a list of statements.
  - A “return” statement can be used to exit the function, and return a given value.

Defining New Functions -- Example

```
def double(something):
    return something+something
```

- Statements inside the function won't get called until the function is called.
- When the function is called:
  - A new variable (something) is created
  - The function's STATEMENTS are executed
  - When a return statement is executed, Python jumps back to the code that called the function.

Defining New Functions -- Stemming

```
SUFFIXES = ['ing', 'es', 's', 'ed']

def stem(word):
    for suffix in SUFFIXES:
        if word.endswith(suffix):
            word = word[:-len(suffix)]
    return word
```

List Comprehensions

- Useful for…
  - Transforming:
    - Do something to every element in a list.
  - Filtering:
    - Keep only the elements that satisfy some condition.

```
[expr for var in sequence]
[expr for var in sequence if test]
```

Assignment 1 - Using List Comprehensions & Functions

```
male_words = ['man', 'men', 'he']
female_words = ['woman', 'women', 'she']

def male_female_ratio(document):
    fd = FreqDist([word.lower() for word in document])
    male = sum([fd[word] for word in male_words])
    female = sum([fd[word] for word in female_words])
    print '%20s %4d %4d' % (item, male, female)

for item in inaugural.items:
    male_female_ratio(inaugural.tokenized(item))
```

- Write a function that finds the average of a list of numbers
- Find the average length of words that start with vowels in Brown corpus section a

```
from nltk.corpus import brown
brown.read('a')
```