Natural Language Processing

Lecture 5: Scripting with Python

Preliminaries

- How comfortable is everyone with writing code?
- How comfortable is everyone with Python?
- Python is a scripting language.

  Why is this good?

  Why is this bad?

Caveats

- There are many ways to do the same thing in Python
- We will NOT tell you everything
- We will also tell you things we find useful for text processing using Python
- There are lots of NLP tools to choose from
- There are lots of programming languages to choose from

Outline

- Python
  - File IO
  - Python data types – Lists, dictionaries, tuples
  - String operations
  - Regular expressions
  - Handling Unicode
  - Useful packages

Python Scripts

Hello, World!

```
# comments

print 'Hello, World!'
```

Control flows: for

```
# comments

for i in range(0,11):
  print i,

# tips: xrange is slightly faster
```

output: 0 1 2 3 4 5 6 7 8 9 10

no curly brackets, python uses indentation for code blocks

Reading files

```
# comments

inputfile=open('myfile.txt')

for line in inputfile:
  print line

inputfile.close()
```

Python modules

```
# a module is just a file ending in .py containing
# a set of functions

import os,sys

# comments
```

Python modules

```
# os.py file in the python “site” path
import os
# def walk(…) in os.py
os.walk

# selective import
from os.path import join
join(“/home”, “usrname”)

# rename for convenience
import os as cats
cats.walk(…)
```

Iterating a directory

```
import os,sys

# comments

for path,dirs,files in os.walk('/usr'):
  print path
  print dirs
  print files
```

```
/usr
['bin', 'include', 'lib', 'libexec', 'local', 'sbin', 'share', 'standalone', 'texbin', 'tibs', 'X11',
'X11R6']
[]
/usr/bin
[]
['2to3', '2to3-', ...
```

Writing files

```
# comments

inputfile=open('myfile.txt')
outputfile=open('myoutput.txt','w')

for line in inputfile:
  outputfile.write(line)

inputfile.close()
outputfile.close()
```

Control flows: if

```
# comments

for i in range(0,11):
  if i%3 == 0:
    print 3,
  elif i%2 == 0:
    print 2,
  else:
    print 1,
```

output: 3 1 2 3 2 1 3 1 2 3 2

False in Python

- Things that are False:
  1. None
  2. False (boolean value)
  3. Zero of any numeric type: 0, 0L, 0.0
  4. Any empty sequence: ‘ ’, ( ), [ ]
  5. Any empty mapping: { }

- Logical operators: `and` `or` `not` (like English)

String operations

```
# comments

i='natural'
j='language'
k='processing'

print i+' '+j+' '+k
```

Output: natural language processing

Type conversion

```
# comments

i='1'
j='2'
k='3'

print int(i)+int(j)+int(k)
```

output: 6

Important data types in Python

- file
- bool
- int
- float
- str / unicode (character string)
- list
- dict
- tuples

List

```
# comments

strings=['natural','language','processing']

for s in strings:
  print s,
print 'is fun'
```

output: natural language processing is fun

Processing characters

```
# comments

w='natural'

for i in range(0,len(w)):
  print w[i],

# alternatively
for i in w:
  print i,
```

output: n a t u r a l

Dictionary

```
# comments

dictionary = {}
dictionary['one'] = 1
dictionary['two'] = 2
dictionary['three'] = 3

for d in dictionary:
  print d+' '+str(dictionary[d]),
```

output:

three 3 two 2 one 1

Dictionary

```
# comments

dictionary = {}
dictionary['one'] = 1
dictionary['two'] = 2
dictionary['three'] = 3

word = 'two'
if word not in dictionary:
  dictionary[word] = 0
else:
  print dictionary[word]
```

output: 2

Python functions

```
def myfunction(a,b ='world'):
  print a+b
  return (a, b)

r = myfunction(1,2) # 3
r = myfunction('nat','lang') # natlang
r = myfunction('hello') # helloworld

x, y = myfunction(7, 8) # x=7, y=8

nlp_func = lambda a, b: a+b
nlp_func('hello', 'world')
```

Main “function”

```
# function definitions

# main usually comes at the end
# after the function definitions
if __name__=='__main__':

  a = 10
  . . .
```

Quick style guide

- Use parentheses sparingly
- Indent code with 4 spaces
  - Never mix tabs and spaces
- Imports should be on separate lines
- Naming:-

GLOBAL_CONST, global_var, function_name, module_name, ClassName

Ref: http://google-styleguide.googlecode.com/svn/trunk/pyguide.html
http://legacy.python.org/dev/peps/pep-0008/

PYTHON FOR TEXT

How do I?

GET THE WORDS

String operations

```
# comments

w='natural language processing\n'
tokens=w.strip().split()

for t in tokens:
  print t
```

output: natural
language
processing

String operations

```
# comments

wlist =['natural','language','processing']

print ' '.join(wlist)
```

output: natural language processing

Regular expressions

```
# matching single character

a # matches ‘a’
abc # matches `abc’
[abc] # matches ‘a’,‘b’ or ‘c’
[a-z] # matches ‘a’, ‘b’, …, or ‘z’
[a-z0-9] # matches any lowercase alphanumerics
[^a-z] # matches anything not in [a-z]
\d # matches digits
\D # matches non digits, i.e [^0-9]
\s # matches whitespaces
\b # matches empty string at beginning or end of word
\w # equivalent to [a-zA-Z0-9]

. # matches any single character
```

Regular expressions

```
# matching patterns

fo* # matches ‘f’, ‘fo’, ‘foo’, …
fo+ # matches ‘fo’, ‘foo’, ‘fooo’, …
fo? # matches ‘f’ or ‘fo’

foo|bar # matches ‘foo’ or ‘bar’
fo(o|b)ar # matches ‘fooar’ or ‘fobar’
(foo)+ # matches ‘foo’, ‘foofoo’, …
^foo # matches ‘foo’ at the beginning only
foo$ # matches ‘foo’ at the end only
fo+? # overloaded ?, non-greedy wildcard matching
```

Regular expressions

```
print "Words, words, words.".split()
# ['Words,', 'words,', 'words.']

\W = all characters not in [0-9a-zA-z]

print re.split(r'\W+', 'Words, words, words.')
# ['Words', 'words', 'words', '']


Keep all delimiters too:

print re.split(r'(\W+)', 'Words, words, words.')
# ['Words', ', ', 'words', ', ', 'words', '.', '']
```

How do I?

FIND ALL THE EMAILS

(or URLS/DATES, PRESIDENTS, ETC.)

Regular expressions

```
# Let's find all the presidents in text

line = """President Barack Obama said that
First Lady Michelle Obama said ...
... French President Francois Holland said ...
"""

presRe=re.compile(r'(President( [A-Z][\S]*)+)')

print pres.findall(line)

[('President Barack Obama', ' Obama'), ('President
Francois Holland', ' Holland')]
```

Regular expressions

```
# regex groups with parentheses

re_email = re.compile(r'([0-9a-z][\w_\.-]*)\@([0-9a-z]
[\w_\.-]*)\.([a-z]{2,4})$')


m = re_email.match('johnsmith@cs.cmu.edu')
print m.group() # johnsmith@cs.cmu.edu
print m.group(1) # johnsmith
print m.group(2) # cs.cmu
print m.group(3) # edu

m = re_email.search('my email is johnsmith@cs.cmu.edu')
print m.span() # (12, 32)
print m.span(1) # (12, 21)
```

Regular expressions

```
# substitutions

print re.sub(r'\s+', ' ', 'a   line with     space')
# a line with space


re_email = re.compile(r'([0-9a-z][\w_\.-]*)\@([0-9a-z]
[\w_\.-]*)\.([a-z]{2,4})$')

print re_email.sub(r'shomir@\2.\3', 'johnsmith@cs.cmu.edu')
# shomir@cs.cmu.edu
```

BUT THE WORDS ARE ALL MESSED UP

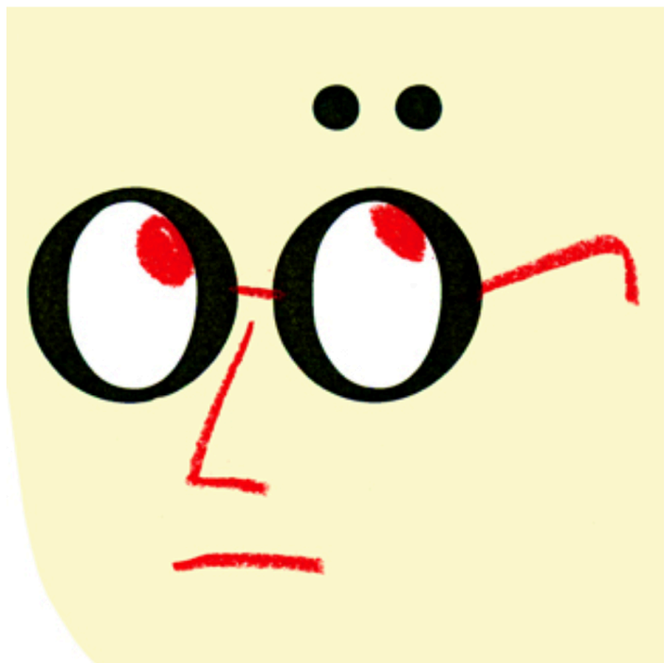
'ascii' codec can't encode character u'\u2019' in position 16: ordinal not in range(128)

Handling Unicode in Python

```
# processing a utf-8 encoded file

contents = open('wiki-article.txt', 'r').read()
u_content = contents.decode('utf-8')


# alternatively
import codecs
contents = codecs.open('wiki-article.txt', 'r',
'utf-8').read()

# common Unicode symbols: ‘, ’, “, ”, –
# diacritics: á, é, í, ñ
a_content = u_content.encode('ascii', errors='ignore')
# errors = ['strict', 'ignore', 'replace']
```

The special tool we use here at *The New Yorker* for punching out the two dots that we then center carefully over the second vowel in such words as "naïve" and "Laocoön" will be getting a workout this year, as the Democrats coöperate to reëlect the President.

Mary Norris, "The Curse of the Diaeresis," New Yorker. April 26, 2012



Handling Unicode in Python

```
# ñ (\u00f1) is also n (\u006e) + ~ (\u0303)
# we need normalization!

# http://en.wikipedia.org/wiki/Unicode_equivalence#Normalization

# coding: utf-8
import unicodedata
n = unicodedata.normalize('NFKD', u'ñ') # u'n\u0303'

# ignore diacritics
unicodedata.normalize('NFKD', u'André').encode('ascii',
errors='ignore')
# Andre

__RE_HYPHENS = regex.compile(ur'[\p{Pd}\p{Pc}]+', re.U)
```

Handling Unicode in Python

```
# more un-latin like alphabets?
# arabic? cyrillic? devanagari scripts? chinese,
japanese, korean?

import unidecode # http://pypi.python.org/pypi/Unidecode
import unihandecode # http://pypi.python.org/pypi/Unihandecode

print (u"\u5317\u4EB0")
# 北京

print unidecode(u"\u5317\u4EB0")
# Bei Jing

# transliteration!
# it is an open problem: FSTs? Machine learning?
```

How do I?

COUNT STUFF

Python "Counter"

```
# Counter
from collections import Counter

cnt = Counter()
for word in ['red', 'blue', 'red', 'green', 'blue']:
  cnt[word] += 1
print cnt
# Counter({'blue': 2, 'red': 2, 'green': 1})

print cnt + cnt
# Counter({'blue': 4, 'red': 4, 'green': 2})

words = re.findall('\w+ly', open('alice.txt').read().lower())
print Counter(words).most_common(3)
# [('only', 52), ('hastily', 16), ('certainly', 14)]
```

How do I?

COUNT MORE THAN WORDS

Useful Python modules

```
import sys, os, datetime, math, string, random
import subprocess
import urllib, BeautifulSoup, json, pickle

import re

import collections

import numpy, scipy, pandas, matplotlib, nltk,
gensim, networkx
```

Python for NLP

```
import nltk
# http://nltk.org/ and http://nltk.org/book/
# natural language toolkit

sentence = "At eight o'clock on Thursday morning Arthur
didn't feel very good."
tokens = nltk.word_tokenize(sentence)
print tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']

tagged = nltk.pos_tag(tokens)
print tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN')]

# tokenization, tagging, parsing, chunking, etc.
```

Python for NLP

```
from nltk.corpus import wordnet

for syn in wordnet.synsets('bank', 'n'):
    print syn.name(), syn.definition()
```

bank.n.01 sloping land (especially the slope beside a body of water)
depository_financial_institution.n.01 a financial institution that accepts deposits and channels the money into lending activities
bank.n.03 a long ridge or pile
bank.n.04 an arrangement of similar objects in a row or in tiers
bank.n.05 a supply or stock held in reserve for future use (especially in emergencies)
...

Python for NLP

```
from nltk.corpus import wordnet

for syn in wordnet.synsets('bank', 'n'):
    print syn.hypernyms()
```

```
[Synset('slope.n.01')]
[Synset('financial_institution.n.01')]
[Synset('ridge.n.01')]
[Synset('array.n.01')]
[Synset('reserve.n.02')]
[Synset('funds.n.01')]
[Synset('slope.n.01')]
...
```

Python for NLP

```
from nltk.corpus import wordnet

for syn in wordnet.synsets('bass', 'n'):
    print syn.hyponyms()
```

```
[]
[Synset('figured_bass.n.01'), Synset('ground_bass.n.01')]
[]
[Synset('striped_bass.n.01')]
[Synset('smallmouth_bass.n.01'), Synset('largemouth_bass.n.01')]
[Synset('basso_profundo.n.01')]
[Synset('bass_horn.n.01'), Synset('bass_guitar.n.01'), Synset('bass_fiddle.n.01')
[Synset('freshwater_bass.n.02')]
```

How do I?

LEARN

Python for Machine Learning

```
import numpy as np, scipy, matplotlib

# http://www.numpy.org/ <- handle numbers
# http://www.scipy.org/ <- scientific functions
# http://matplotlib.org/ <- plotting
# http://scikit-learn.org/stable/ <- machine learning

a = np.array([1,2,3])
print a[1:3] # matlab style indexing
print np.dot(a, a) # linear algebra functions

# many more implementations for common mathematical
# functions like root finding, FFTs, etc

# matlab-like plotting with matplotlib
```

Python for Machine Learning

```
from sklearn import linear_model, datasets
iris = datasets.load_iris()
X=iris.data
Y=iris.target
logreg = linear_model.LogisticRegression()
logreg.fit(X, Y)

# learned parameters:
logreg.coef_
logreg.intercept_


# classification, regression, clustering, dimensionality
reduction
```

IN PRACTICE

Putting it all together

Example python script to extract features from documents and learn a movie review sentiment classifier, with:

- nltk, re
- numpy
- scikit-learn
- scipy

http://bit.ly/1ncNt85