An Introduction to Python for Text Analysis

Alison Craig

Department of Political Science

November 18, 2015

The Ohio State University

How to Use Python

- Why Python?
- Python 2.7 vs Python 3.0
- Basic command line exercises
- Python scripts for more complex problems

Syntax

- Types of values:
  - boolean: True, False
  - integer: 7, 100
  - float: 7.3, 100.0
  - string: 'hello world', 'congress'
  - list: [5, 10, 'congress', 3.5]
  - dictionary: {'aaron':1, 'alex':2}
  - tuple: ('a', 'b', 'c')

- Variable names:
  - begin with lowercase letter
  - letters and numbers allowed
  - underscores allowed
  - names are case sensitive
  - no reserved words
  -
```
name = "python"
type(name)
<type 'str'>
```

Reserved Words

```
and as assert break class continue def del elif else
except exec finally for from global if import in is
lambda not or pass print raise return try while with
yield
```

Operators and Conditionals

- Operators
  - mathematical: `+, -, *, /, **, %`
  - string: `+` to concatenate strings
  - comparison: `==, !=, >, <, >=, <=, is, is not`
  - logical: `and, or, not`
- Conditional statements
  - ```
    if x > 0 :
        print 'x is positive'
    ```
  - ```
    if x < y :
        print 'x is less than y'
    elif x > y :
        print 'x is greater than y'
    else :
        print 'x and y are equal'
    ```

Exceptions

- Try and Except to catch exceptions
  - ```
    inp = raw_input('Enter Fahrenheit Temperature:')
    try :
        fahr = float(inp)
        cel = (fahr - 32.0) * 5.0 / 9.0
        print cel
    except :
        print 'Please enter a number'
    ```
    ```

Functions and Iterations

- Functions can be built-in, imported, or user-defined
  - `max, min, len, int, float, str, help`
  - `random.random(), random.randint(x,y), math.sqrt(x)`
  - ```
    def tempconvert(fahr) :
        cel = (fahr - 32.0) * 5.0 / 9.0
        print cel
    tempconvert(100)
    ```
- Iterations use while and for loops
  - ```
    n = 5
    while n > 0 :
        print n
        n = n-1
    ```
  - ```
    total = 0
    for iter in [3, 41, 12, 9, 74, 15] :
        total = total + iter
    print total
    ```
    ```

Strings and Lists

- Characters in strings are accessed with the bracket operator
  - ```
    fruit = 'banana'
    letter = fruit[0]
    half = fruit[:len(fruit)/2]
    bigfruit = fruit.upper()
    index = fruit.find('na', 3)
    ```
- Lists are sequences of elements enclosed in brackets
  - ```
    cheeses = ['Cheddar', 'Edam', 'Gouda']
    cheeses[2] = 'Swiss'
    middle = cheeses[1:2]
    for i in range(len(cheeses)) :
        cheeses[i] = cheeses[i].upper()
    cheeses.append('SWISS')
    ```
- Break strings into lists with the `list` and `split` functions

Dictionaries and Tuples

- Dictionaries map keys to a set of values, giving a set of unordered key-value pairs
  - `eng2sp = dict()`
    `eng2sp = {'one':'uno', 'two':'dos', 'three':'tres'}`
- Tuples are basically immutable lists
  - `t = ('a', 'b', 'c', 'd', 'e')`
  - `m = ['have', 'fun'] x, y = m`
  - Can be used as key-value pairs in dictionaries

Reading and Writing Files

- Opening a file returns a file handle, not the actual data. Short files can be read into a single string, otherwise read the file in chunks using a `for` or `while` loop.
  - ```
    fhand = open('data.txt')
    for line in fhand :
        line = line.rstrip()
        if line.startswith('From:'):
            print line
    ```
- To write to a text file, open a new file with mode 'w' and use the write command to add new data to the end until you close the file.
  - ```
    fout = open('output.txt', 'w')
    fout.write(line)
    fout.close()
    ```

Modules

- Modules and packages are a collection of functions. Some come with python, others need to be installed.
- Pre-installed modules are loaded with `import modulename`
- For other packages, you can download and install packages from their source or use an installer like `pip`
  - download get-pip.py from https://pip.pypa.io
  - `sudo python get-pip.py`
  - `sudo pip install packagename`
  - `import` command within python interpreter

Useful Modules and Libraries

- `csv` - built in module for reading and writing csv files
- `re` - regular expression library
- `urllib2` - build in module for reading web pages from a URL
- `nltk` - package for pre-processing and classification
- `BeautifulSoup` - package for parsing HTML texts

Regular Expressions

- Regular expressions allow you to use special characters to more precisely search for and extract data
- Character matching:
  - `if re.search('^F..m:', line)`
  - `if re.search('^From:.+@', line)`
- Extracting data:
  - `lst = re.findall('\S+@\S', data)`
  - `[a-zA-Z0-9]\S*@\S*[a-zA-Z]`
- Basic search strings:
  - ^ Matches the beginning of the line
  - . Matches any character
  - \s Matches a whitespace character
  - \S Matches a non-whitespace character
  - [aeiou] Matches a character in the set
  - [^A-Za-z] Matches a character not in the set

Additional Resources

- Coursera Python for Everybody Specialization
  - Programming for Everybody
  - Python Data Structures
  - Using Python to Access Web Data
  - Using Databases with Python
- Udacity Intro to Computer Science course
- learnpythonthehardway.org
- textminingonline.com