Python Workshop

gaurav sood
http://gsood.com
twitter | github

April 10, 2015

Why Python?

- Readable, easy
- Automatic memory management
- Very High Level Language with High level data types built in
- Python is extensible.
  NumPy, SciPy, MatPlotLib etc.
- Python is an interpreted language. No need to compile.

More on Python

- Key developer Guido van Rossum
- Name comes from Monty Python. Guido was a fan.
- Object oriented (based) but also functional
- Dynamic Typing
- Get started:

```
Unix:
    add /usr/local/bin to the search path
Windows:
    set path=%path%;C:\python27
```

Some things to know

- Statement grouping done by indentation
  (instead of beginning and ending brackets)
- No variable or argument declarations are necessary.
- Index begins at 0
- How to get help
  help(), help(tuple)
- Some useful functions

  dir(str), type()

Installing Packages

- pip or easy_install
- easy_install

```
wget https://bootstrap.pypa.io/ez_setup.py -O - |python
easy_install nytimesarticle
```

- pip

```
wget https://bootstrap.pypa.io/get-pip.py
or python get-pip.py
pip install nytimesarticle
pip install --upgrade nytimesarticle
pip install -r requirements.txt
pip install nytimesarticle
```

Data Structures: Lists

Look up pandas for R like data structures

- List:

  `['apple', 1, 'o', True]`

- Indexing:
  - `['apple', 1][1]` or `['apple', 1][-1]`
  - `['apple', 1, 'o', True][1:3]`
  - `['apple', 1, 'o'][:1] + ['apple', 1, 'o'][1:]`
- Adding Stuff:
  - Append:

    `['apple', 1, 'o'].append('juice')`
  - Insert:

    `a =['apple', 1, 'o']; a.insert(2, 'juice')`
  - Extend:

    `a =['apple', 1, 'o']; a.extend([2,2])`

Data Structures: Lists

- Searching:
  - `a =['apple', 1, 'o']; a.index('apple'); a.index(8)`
  - `'apple' in a`
- Deleting Stuff:
  - Remove:

    `a.remove('apple')`
  - Pop:

    `a.pop()`
- Operators:
  - `a += ['juice']`
  - `[1, 2, 3]*3`

Data Structures: Tuples

- Tuple:

  `('apple', 1, 'o', True)`
- No `append`, `remove`, `index`

Data Structures: Dictionary

- `a = {"fruit":"apple", "color":"red"}`
- `a["fruit"]`
- `a["fruit"]='orange'`
- `a["shape"]='sphere'`
- `del a["color"]; a.clear()`

Scraping

Scraping

- To analyze data, we typically need structure.
  For instance, same number of rows for each column.
- But found data often with human readable structure.
- Copy and paste, type, to a dataset.
- But error prone, and not scalable.
- Idea: Find the less accessible structure, automate based on it.

Collecting Found Digital Data

- Software
  - R - Not the best but will do.
  - Python, Ruby, Perl, Java, ...
  - 30 Digits, 80 Legs, Grepsr ...
- Some things to keep in mind
  - Check if there is an API, or if data are available for download
  - Play Nice:
    Scraper may be disallowed in 'robots.txt'
    Build lag between requests. Make lags random.
    Scrape during off-peak hours

Paper

- Create digital images of paper
- Identify colored pixels as characters (OCR)
- Software
  - Adobe Pro., etc.
  - Best in class commercial: Abbyy FineReader
    Now has an API
  - Best in class open-source: Tesseract (Google, command line tool)
- Scrape off recognized characters: pyPdf etc.
- Post-processing

Scraping one HTML page in Python

Shakespeare's Twelfth Night
Using Beautiful Soup

```
from BeautifulSoup import BeautifulSoup

from urllib import urlopen


url  = urlopen(`http://bit.ly/1D7wKcH').read()

soup = BeautifulSoup(url)

text = soup.p.contents

print text
```

Getting text from one pdf in Python

A Political Ad
Using PyPdf

```
import pyPdf


pdf = pyPdf.PdfFileReader(file('path to pdf', `rb'))

content = pdf.getPage(0).extractText()

print content
```

Scraping many urls/files to structured data

- Loop, exploiting structure of the urls/file paths
  e.g. ESPN URL
- Handle errors, if files or urls don't open, what do you do?
- To harvest structured data, exploit structure within text
- Trigger words, html tags, ...

Exception(al) Handling

```
try:
     pdf = pyPdf.PdfFileReader(file(pdfFile, 'rb'))
except Exception, e:
     return `Cannot Open: %s with error: %s' %
(pdfFile, str(e))
```

Inside the page

- Chrome Developer Tools
- Quick Tour of HTML
  - Tags begin with < and end with >
  - Tags usually occur in pairs. Some don't (see img). And can be nested.
  - Mozilla HTML elements
  - <p> is for paragraph
  - <a> is for a link
  - , <ul> is for ordered, unordered list; is a bullet
  - tags can have attributes.
  - DOM, hierarchical, parent, child:

```
<html>
    <body>
        <p></p>
    </body>
</html>
```

Find Things

Navigate by HTML tags:

```
soup.title, soup.body, soup.body.contents
```

Search HTML tags:

```
soup.find_all('a'), soup.find(id="nav1")
```

So to get all the urls in a page:

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

Beautiful Soup Documentation

Text Processing

Text as Data

- Bag of words assumption
  Lose word order

- Remove stop words:
  If, and, but, who, what, the, they, their, a, or, ...
  Be careful: one person's stopword is another's key term.

- (Same) Word: Stemming and Lemmatization
  Taxing, taxes, taxation, taxable $\rightsquigarrow$ tax

- Remove rare words
  $\sim$ .5% to 15%, depending on application

- Convert to lowercase, drop numbers, punctuation, etc.

How?

Using Natural Language Toolkit (`nltk`)

- Lowercase:

```
text = text.lower()
```

- Remove stop words:

```
swords = stopwords.words('english')
words  = wordpunct_tokenize(text)
words  = [w for w in words if w not in swords]
text = ' '.join(words)
```

- Stemming:

```
st = EnglishStemmer()
words = wordpunct_tokenize(text)
words = [st.stem(w) for w in words]
text = ' '.join(words)
```

To Matrices

- n-grams

```
from nltk import bigrams, trigrams, ngrams
text = word tokenize(text)
text_bi  = bigrams(text)
```

Resources:

- Python Tutor
- Code Academy
- Learn Python The Hard Way