FoxMIS CONNECT | INNOVATE

Fox School of Business
TEMPLE UNIVERSITY®

Python and Web Data Extraction: *Introduction*

Alvin Zuyin Zheng
zheng@temple.edu
http://community.mis.temple.edu/zuyinzheng/

About

- About Me: Alvin Zuyin Zheng , MIS
- Following up with the workshop in May
- Organizers: Sudipta Basu, Lalitha Naveen, Jing Gong
- This workshop is supported by the Office of Research and Doctoral Programs at Fox. Thanks Paul, Lindsay and everyone in the Office of Research!

About

- Student Assistants:
  - Shawn J Niederriter
  - Xue Guo
  - Zhe Deng
- Website: http://community.mis.temple.edu/zuyinzheng/pythonworkshop/

- Maintenance from 12:00 to 2:00pm!

Topics

1. Python Basics
2. Web Scraping
3. Introduction to Natural Language Processing

- *No prior programming experience needed*

Schedule

9:50 am	Welcome and Set Up
10:00 am	**Session 1–Python basics**
11:00 am	Coffee Break
11:20 am	**Session 2–Web Scraping (Part 1)**
12:20 pm	Lunch Break
1:20 pm	**Session 3–Web Scraping (Part 2)**
2:20 pm	Coffee Break
2:40 pm	**Session 4– Basic Intro to Natural Language Processing**
3:45 pm	Closing Remarks and Questions

FoxMIS CONNECT | INNOVATE

Fox School of Business
TEMPLE UNIVERSITY®

Python and Web Data Extraction: *Python Basics*

Jing Gong
gong@temple.edu
http://community.mis.temple.edu/gong

Prerequisites

- Before the workshop, your computer needs the following tools installed and working to participate.
  - A command-line interface to interact with your computer
  - A text editor to work with plain text files
  - Python 2.7
  - The `pip` package manager for Python
  - A browser that can view web source code like Chrome

(Please follow the set up guide posted here)

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

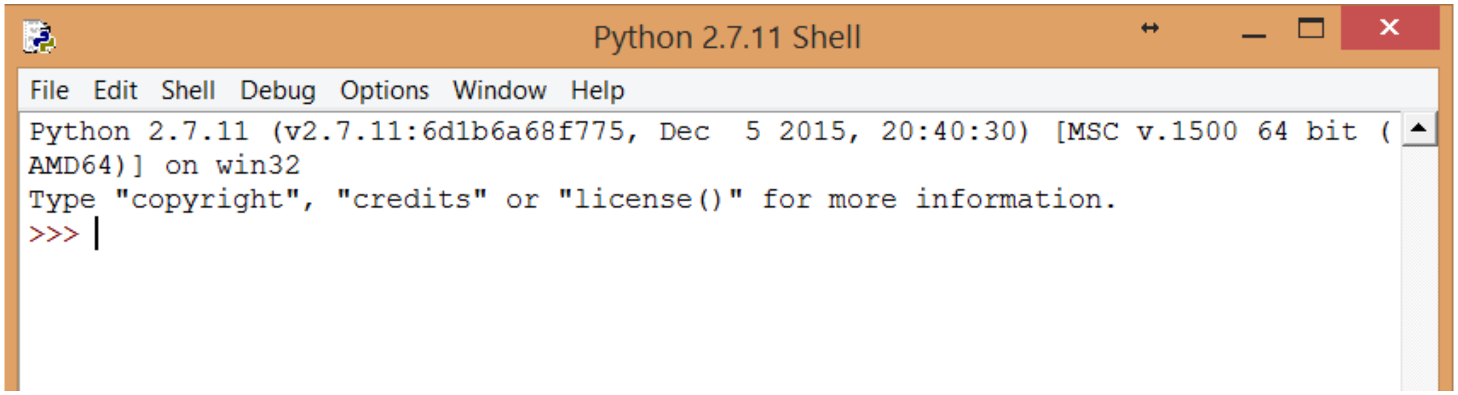
Why Python?

- Simple
- Easy to learn
- Free and open source
- Portable across platforms
- With extensive libraries

- Python 2 versus 3:
  - Very different
  - We will use the latest version of Python 2 (Latest version is Python 2.7.12)

Python IDLE (Interactive Shell)

- The Python IDLE provides an interactive environment to play with the language



```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

- Open Python IDLE
  - On Windows: tap the Windows key on your keyboard and type “idle” to open the “IDLE (Python GUI)”
  - On Mac: use Cmd+Space and type “idle” to select the “IDLE.”

Python IDLE (Interactive Shell)

- You can type commands directly into the interactive shell
- Results of expressions are printed on the screen

```
>>> 1+3
4
>>> workshop = "Python"
>>> workshop
'Python'
>>> print "hello"
hello
```

Variables are assigned using the “=“ sign

Indentation

- Python uses indentation (usually four spaces) to structure a block of codes
  - no curly braces {} to mark where the function code starts and stops

```
>>> x = 10
>>> if x>10:
...     print "x is larger than 10"
... else:
...     print "x is less than or equal to 10"
```

Returns:

```
x is less than or equal to 10
```

★

Comments

- Comments: Texts mainly useful as notes for the reader of the script.
- Are to the right of the # symbol

```
>>> print "hello" #this is a comment
Hello
>>> #this is a comment
>>>
```

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

Basic Data Types

- Numbers
  - Integers

```
>>> month = 3
```

  - Floats

```
>>> income = 100.2
```

  - Calculations

```
>>> 100/month
33
>>> income/month
33.4
```

Basic Data Types

- Strings
  - Specify strings using either single quotes or double quotes

```
>>> gender = "Male"
>>> name = 'Jack'
```

  - Use triple quotes for strings across multiple lines

```
>>> paragraph = """This is a long
paragraph with multiple lines."""
```

  - Concatenate strings using “+”

```
>>> name + gender
'JackMale'
```

List [ ]

- List: An ordered collection of data
  - You can have *anything* in a list:

```
>>> [0]
>>> [2.3, 4.5]
>>> [5, "Hello", "there", 9.8]
>>> range(4)
[0, 1, 2, 3]
```

  - Use len() to get the length of a list

```
>>> names=["Ben","Jack","Lee","Nick"]
>>> len(names)
4
```

Accessing and Updating Values in Lists

- Use [] to access values in the list

```
>>> names=["Ben","Jack","Lee","Nick"]
>>> names[0]
'Ben'
>>> names[1:3]
['Jack', 'Lee']
>>> names[1:]
['Jack', 'Lee', 'Nick']
>>> [names[i] for i in [1,3]]
['Jack', 'Nick']
```

[0]: the 1st value
[1]: the 2nd value
…

- Update values

```
>>> names[1]="Ann"
>>> names
['Ben', 'Ann', 'Lee', 'Nick']
```

To learn more about Python Lists: Visit here

Other Types

- Dictionaries {}
  - consist of key-value pairs.

```
>>> dict = {'name': 'Ann', 'age': 10}
```

- Tuples ()
  - Similar to list, but cannot be updated.

```
>>> tup = ('Ann', 10)
>>> tup[1]=12
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    tup[1]=12
TypeError: 'tuple' object does not support
item assignment
```

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

If

- The **`if`** statement checks a condition
- And can be combined with … `elif` … `else`

```
>>> age = 10
>>> if age>10:
...     print "age is greater than 10"
... elif age<10:
...     print "age is less than 10"
... else:
...     print "age is equal to 10"

age is equal to 10
```

For

- The **`for`** statement loops iterate over each value in a list

```
>>> for i in range(4):
...     print "i: ",i

i:  0
i:  1
i:  2
i:  3
```

Break

- **`break`** can be used to stop the for loop

```
>>> for i in range(4):
...     print "i: ",i
...     if i>1:
...         print "Exiting the for loop"
...         break

i:  0
i:  1
i:  2
Exiting the for loop
```

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

Packages (Modules)

- Python itself only has a limited set of functionalities.
- **Packages** provide additional functionalities.
- Use "`import`" to load a package

```
>>> import math
>>> import os
```

Install Packages Using `pip`

- For standard packages
  - You do not need to be installed manually

- For third-party packages
  - Use **`pip`** in your **command line interface** to install

```
pip install SomePackage
```

  - For example, to install the beautiful soup package:

```
pip install beautifulsoup4
```

See these instructions for how to open the command line interface.

- On Windows it is called “Command Prompt.”
- On Mac it is called “Terminal.”

Functions

- A *function* is a named sequence of statements that performs a desired operation

- Define a function:

```
>>> def f(x):
...     return x*2

>>> f(1)
2

>>> f(0.5)
1
```

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

Working Directory

- Working Directory: Think of it as the folder your Python is operating inside at the moment

- Get the current working directory:

```
>>> import os
>>> os.getcwd()
'C:\\Python27'
```

- Change the current working directory:

```
>>> os.chdir("C:/users/jing")
>>> os.getcwd()
'C:\\users\\jing'
```

Use forwardslash (/) or double backslash (\\) when specifying directories.

File Open/Close

- To use a file, you have to open it using the `open` function

  (Note: The following codes assumes that your files are in your current working directory)

```
>>> input_file = open("in.txt", "r")
>>> output_file = open("out.txt", "w")
>>> for line in input_file:
      output_file.write(line)
```

`"r"`	means reading
`"w"`	means writing
`"a"`	means appending

- When done, you have to close it using the `close` function

```
>>> input_file.close()
>>> output_file.close()
```

Outline

- Overview
- Data Types
- Control Flow
- Packages and Functions
- File Input/Output
- Regular Expression
- Tutorial 1. First Running the First Python Script

Regular Expressions (*RE*)

- *Regular Expressions* are a powerful **text** manipulation tool for
  - searching, replacing, and parsing text patterns

- You need to load the "**`re`**" package

```
>>> import re
```

Exact match: An example

- Suppose we have a text string, and want to know if the string has the word "`cat`" in it…

String:
```
" A fat cat doesn't eat oat
but a rat eats bats. "
```

- `re.search(pattern, string[, flags])`
  - find the **first** location where the *pattern* produces a match

```
>>> import re
>>> teststring = """ A fat cat doesn't eat oat but a
rat eats bats. """
>>> match = re.search("cat",teststring)
>>> print match.group()
cat
```

Extracting Texts

- How to **extract** everything between "`cat`" and "`rat`"?

String:
```
"A fat cat doesn't eat oat but
a rat eats bats."
```

- We can define a *pattern* `"cat(.*?)rat"`
  - `(.*?)` represents everything in between "cat" and "rat"

```
>>> match = re.search("cat(.*?)rat",teststring)
>>> print match.group(0)
cat doesn't eat oat but a rat
>>> print match.group(1)
 doesn't eat oat but a
```

Find All Matched Subtrings

- `re.findall(pattern, string[, flags])`
  - Find all matched substrings with a given pattern

- `".at"` is a pattern that matches any of 'fat', 'cat', 'eat', 'oat', 'rat', 'eat' (or '1at', '2at', ‘aat', ‘bat‘ ...)

```
>>> import re
>>> teststring = """ A fat cat doesn't eat oat but
a rat eats bats. """
>>> match = re.findall(".at",teststring)
>>> print match
['fat', 'cat', 'eat', 'oat', 'rat', 'eat']
```

Escape Character

- Special characters often cause problems because they are used to define patterns
- If you want a special character to just behave normally (most of the time) you prefix it with backslash (\ )
-

Escape	Meaning
\\	\
\ '	'
\ "	"
\n	newline
\t	tab
\r	return
\.	.

More about Regular Expression

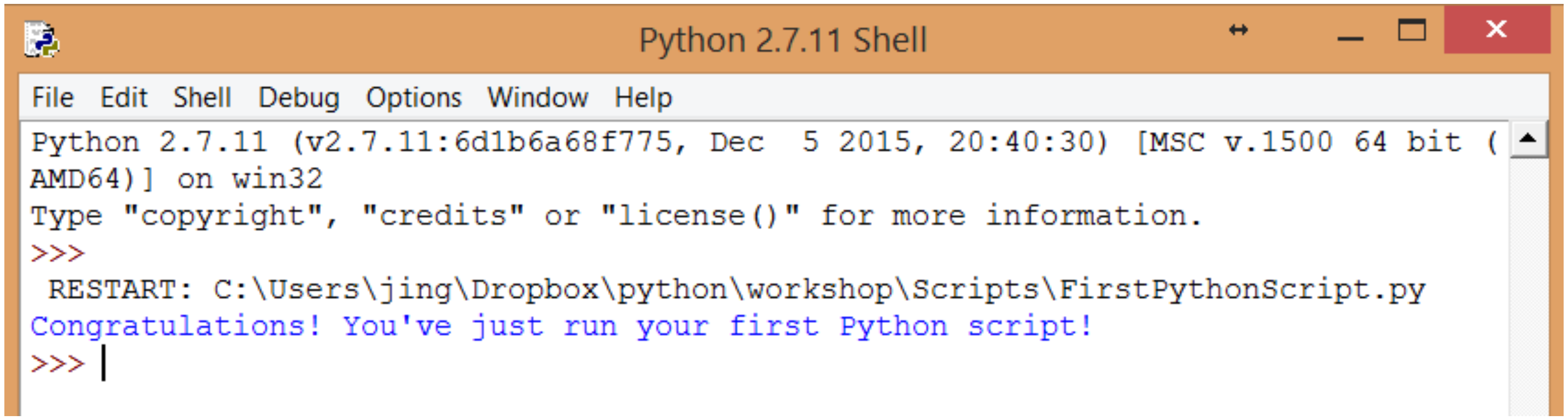
- Python's official Regular Expression HOWTO:
  - https://docs.python.org/2/howto/regex.html#regex-howto
- Google's Python Regular Expression Tutorial:
  - https://developers.google.com/edu/python/regular-expressions
- Test your regular expression:
  - https://regex101.com/#python

Outline

Tutorial 1: Running Your First Python Script

- Download the FirstPythonScript.py file from the website and try to run it

- Steps
  1. Locate the .py file you'd like to run in your folder
  2. Open the the .py file with IDLE
  3. Click the "Run" menu and choose "Run Module"



```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
 RESTART: C:\Users\jing\Dropbox\python\workshop\Scripts\FirstPythonScript.py
Congratulations! You've just run your first Python script!
>>> |
```

Online Recourses for Python

- Python's BeginnersGuide listed many online books and tutorials: https://wiki.python.org/moin/BeginnersGuide/Programmers
- Python's Official Tutorial: https://docs.python.org/2/tutorial/
- Python Basic Tutorial by TutorialsPoint: http://www.tutorialspoint.com/python/
- learnpython.org: http://www.learnpython.org/
- Google's Python Class: https://developers.google.com/edu/python/