Lecture 18: HTML and Web Scraping

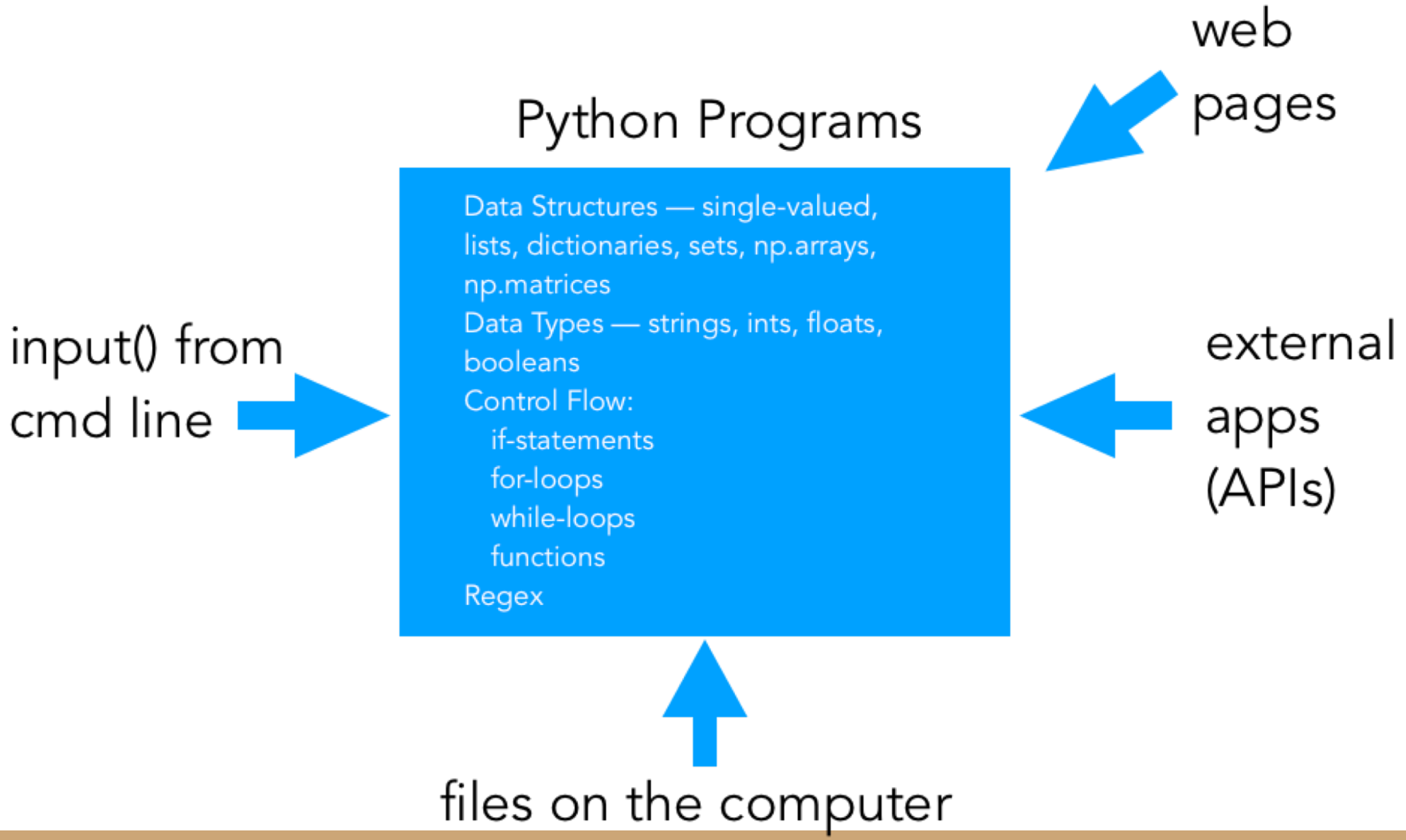
November 6, 2018

Reminders

- Project 2 extended until Thursday at midnight!
  - Turn in your python script and a .txt file
  - For extra credit, run your program on two .txt files and compare the sentiment analysis/bigram and unigram counts in a comment. Turn in both .txt files
- Final project released Thursday
  - You can do this with a partner if you want!
  - End goal is a presentation in front of the class in December on your results
  - Proposals will be due next Thursday

Today's Goals

- Understand what Beautiful Soup is
- Have ability to:
  - download webpages
  - Print webpage titles
  - Print webpage paragraphs of text


web
pages
Python Programs
Data Structures — single-valued,
lists, dictionaries, sets, np.arrays,
np.matrices
Data Types — strings, ints, floats,
booleans
Control Flow:
if-statements
for-loops
while-loops
functions
Regex
input() from
cmd line
external
apps
(APIs)
files on the computer

HTML

- Hypertext Markup Language: the language that provides a template for web pages
- Made up of tags that represent different elements (links, text, photos, etc)
- See HTML when inspecting the source of a webpage

HTML Tags

- <html>, indicates the start of an html page
- <body>, contains the items on the actual webpage (text, links, images, etc)
- <p>, the paragraph tag. Can contain text and links
- <a>, the link tag. Contains a link url, and possibly a description of the link
- <input>, a form input tag. Used for text boxes, and other user input
- <form>, a form start tag, to indicate the start of a form
- <img>, an image tag containing the link to an image

Getting webpages online

- Similar to using an API like last time
- Uses a specific way of requesting, HTTP (Hypertext Transfer Protocol)
- HTTPS has an additional layer of security
- Sends a request to the site and downloads it
- HTTP/HTTPS have status codes to tell a program if the request was successful
- 2—, request was successful; 4— client error, often page not found; 5— server error, often that your request was incorrectly formed

Web Scraping

- Using programs to download or otherwise get data from online
- Often much faster than manually copying data!
- Makes the data into a form that is compatible with your code

Python and Webpages and Scraping, made easy

- Requests library: gets a webpage for you
- requests.get(url)

- BeautifulSoup library parses webpages (.html content) for you!
- Use BeautifulSoup to find all the text or all of the links on a page
- Documentation: https://www.crummy.com/software/BeautifulSoup/

Running the “three sisters” document through Beautiful Soup gives us a `BeautifulSoup` object, which represents the document as a nested data structure:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

print(soup.prettify())
# <html>
#  <head>
#   <title>
#    The Dormouse's story
#   </title>
#  </head>
#  <body>
#   <p class="title">
#    <b>
#     The Dormouse's story
#    </b>
#   </p>
#   <p class="story">
#    Once upon a time there were three little sisters; and their names were
#    <a class="sister" href="http://example.com/elsie" id="link1">
#     Elsie
#    </a>
#    ,
#    <a class="sister" href="http://example.com/lacie" id="link2">
#     Lacie
#    </a>
#    and
#    <a class="sister" href="http://example.com/tillie" id="link2">
#     Tillie
#    </a>
#    ; and they lived at the bottom of a well.
#   </p>
#   <p class="story">
#    ...
#   </p>
#  </body>
# </html>
```

Here are some simple ways to navigate that data structure:

```
soup.title
# <title>The Dormouse's story</title>

soup.title.name
# u'title'

soup.title.string
# u'The Dormouse's story'

soup.title.parent.name
# u'head'

soup.p
# <p class="title"><b>The Dormouse's story</b></p>

soup.p['class']
# u'title'

soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

One common task is extracting all the URLs found within a page's <a> tags:

```
for link in soup.find_all('a'):
    print(link.get('href'))
# http://example.com/elsie
# http://example.com/lacie
# http://example.com/tillie
```

Another common task is extracting all the text from a page:

```
print(soup.get_text())
# The Dormouse's story
#
# The Dormouse's story
#
# Once upon a time there were three little sisters; and their names w
# Elsie,
# Lacie and
# Tillie;
# and they lived at the bottom of a well.
#
# ...
```

Quick demo on getting webpages from server and parsing!

Advanced Topic

- Given this knowledge, how could we build a web crawler that downloads every webpage on the Internet?
- If we downloaded every webpage, what could we do with it?

Advanced Topic II

- Making requests to download a webpage is literally demanding resources (time and compute power) from the website's server.
- What if we make MANY requests? Is this allowed?
- What are the consequences if we made 100 simultaneous requests to CNN.com at the same time? What about 1,000? 1,000,000?

LAB TIME