Introduction to Web Scraping with Python

NaLette Brodnax

The Institute for Quantitative Social Science

Harvard University

January 26, 2018

workshop structure

1

intro

2

get the tools

3

review Python

4

scrape the web

my goals

- Review Python programming
- Introduce web scraping method
- Provide opportunities to practice

part 1: introduction

what is web scraping?

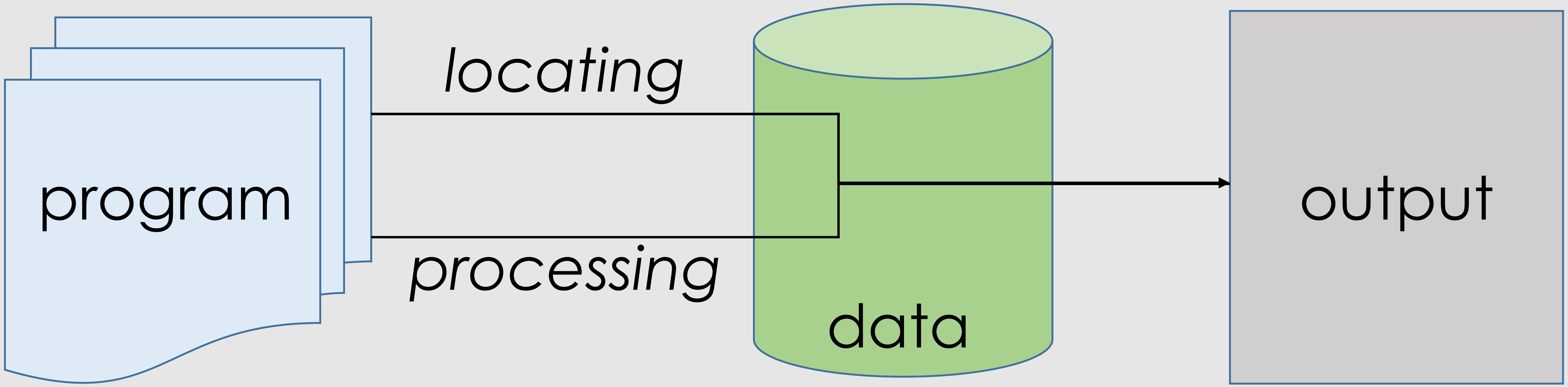
Web scraping is a set of techniques for extracting information from the web and transforming it into structured data that we can store and analyze.

when should you scrape the web?

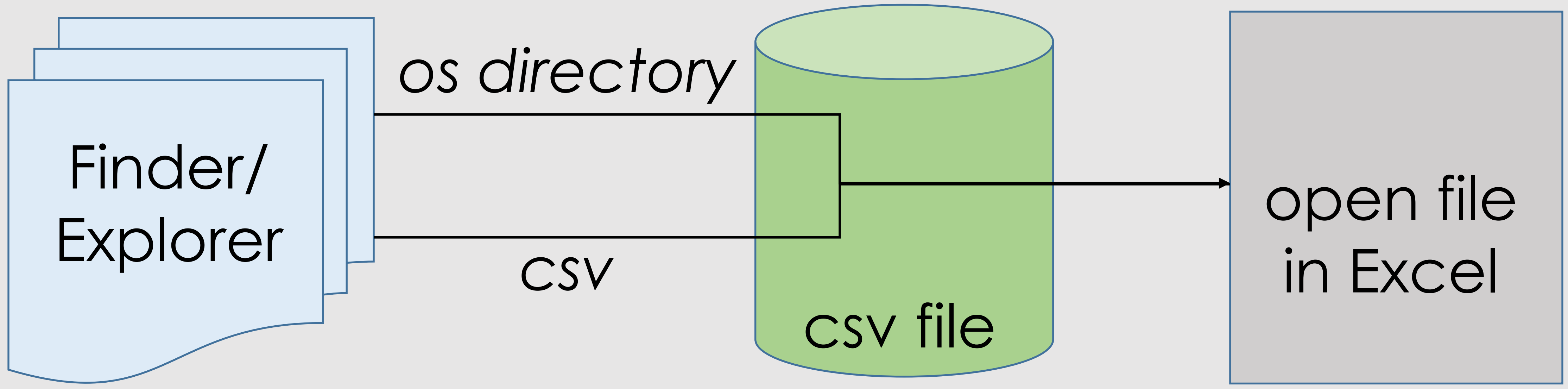
Web scraping should help you be more **efficient**

```
<html>
  <head>
    <meta name="TITLE" content
     <meta name="KEYWORDS" cont
      <meta name="DESCRIPTION"
       <link rel="stylesheet"
              language="ja
```

a simple model: locate and process data

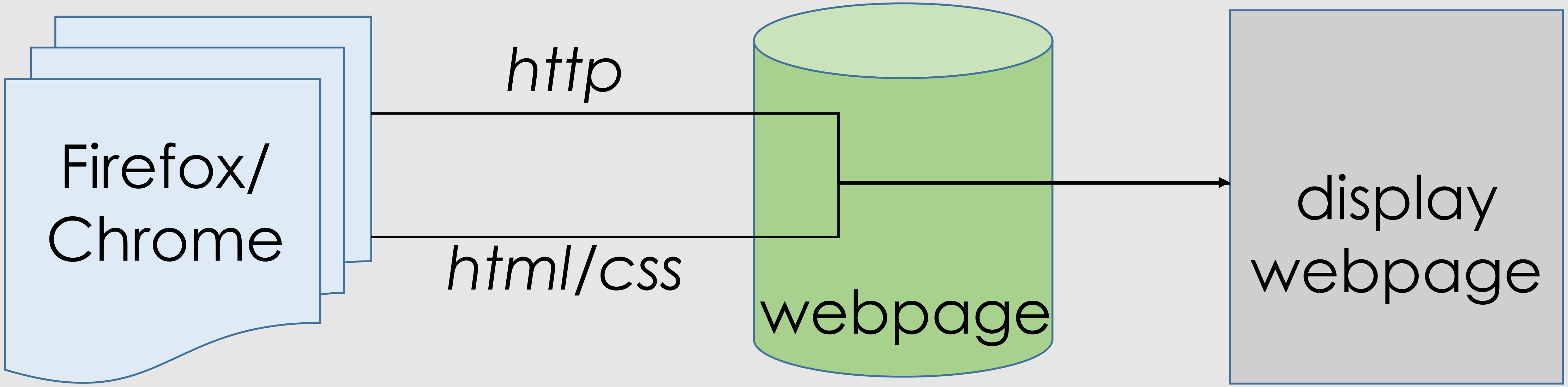


example 1: browse and open csv file



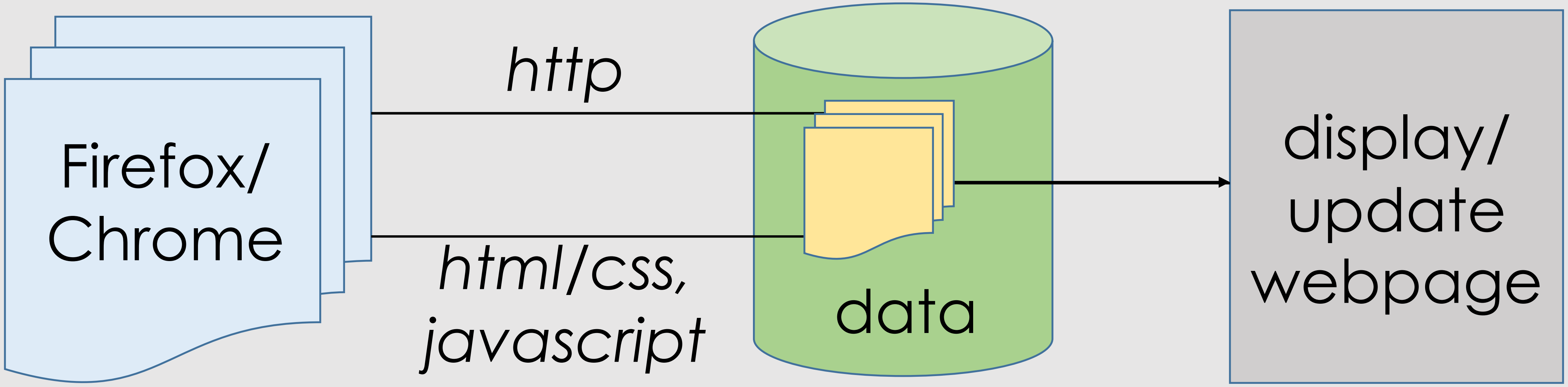
The program and data are all located on one computer

example 2: view a static web page



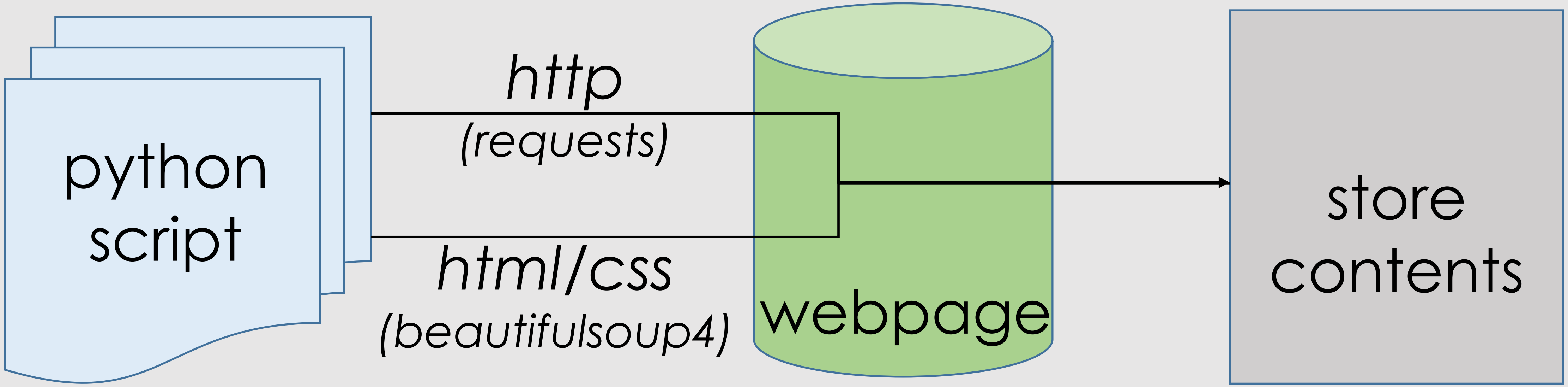
The program and data are located on different computers

example 3: view a dynamic web page



The browser encounters instructions for how to display the webpage based on the user

goal: scrape a static page



Use the **requests** library for http protocols and the **beautifulsoup4** library for html processing

part 2: getting set up

getting the tools

Interpreter → Output

Text Editor + Interpreter → Output

$\boxed{\text{Command Line + Text Editor + Interpreter}}$ → Output

↓

Integrated Development Environment (IDE) → Output

installing the tools on your machine

- Install Anaconda Python 3.6 from Continuum https://www.anaconda.com/download
    - **Spyder** IDE
    - Navigator graphical interface
    - Conda command line utility
    - Includes popular data science packages

not using anaconda?

Download Python 3.6
https://www.python.org/downloads/
→Includes IDLE, an IDE with a text editor and interpreter
→Includes pip, Python's standard package manager

Install the necessary libraries **from the command line**:

```
$ pip3 install --upgrade pip
$ pip3 install requests
$ pip3 install beautifulsoup4
```

Exercise 2.1

create a new script: *review.py*

```
print(“Hello, world.”)
```

Save the program/script.
Run the program in the IDE: **F5** or **Run** or ▶

part 3: Python review

programming in Python

1. sequences
2. control structures
3. writing functions
4. using modules and packages

data types: sequences

String—ordered sequence of characters

```
‘happy’
```

List—ordered sequence of items

```
[‘Leia’, ‘Rey’, ‘Maz’]
```

Dictionary—unordered sequence of key-value pairs

```
{‘name’:‘Kylo’, ‘side’:‘dark’}
```

Exercise 3.1

working with sequences

- Sequences bound by different characters
  - string ‘’
  - list **[]**
  - dictionary **{}**
- Reference items in an **ordered** sequence by number, *starting from 0 or ending at -1*
- Reference dictionary items by key

Exercise 3.2

control structures: loops

for loop

while loop

```
name = ‘Grace Hopper’

i = 0
for letter in name:
    if letter in [‘a’,‘e’,‘i’,‘o’,‘u’]:
        i = i + 1
print(name + ‘ has ’ + str(i) + ‘ vowels.’)

i = 0
vowel_count = 0
while i < len(name):
    if name[i] in [‘a’,‘e’,‘i’,‘o’,‘u’]:
        vowel_count = vowel_count + 1
    i = i + 1
print(name + ‘ has’ + str(vowel_count) + ‘ vowels.’)
```

Exercise 3.3

functions

```
def function_name(argument1, argument2, ...):
    first command
    second command
    return output
```

```
def say_hello(name_string):
    print(‘Hello, ’ + str(name_string) + ‘!’)
    return None

say_hello(‘NaLette’)
```

Exercise 3.4

modules

import statements allow you to add functions

```
import csv

with open("workshop.csv", 'w') as csvfile, \
      open(filename, 'r') as txtfile:
     writer = csv.writer(csvfile)
```

use module name to call functions

part 4: scraping the web

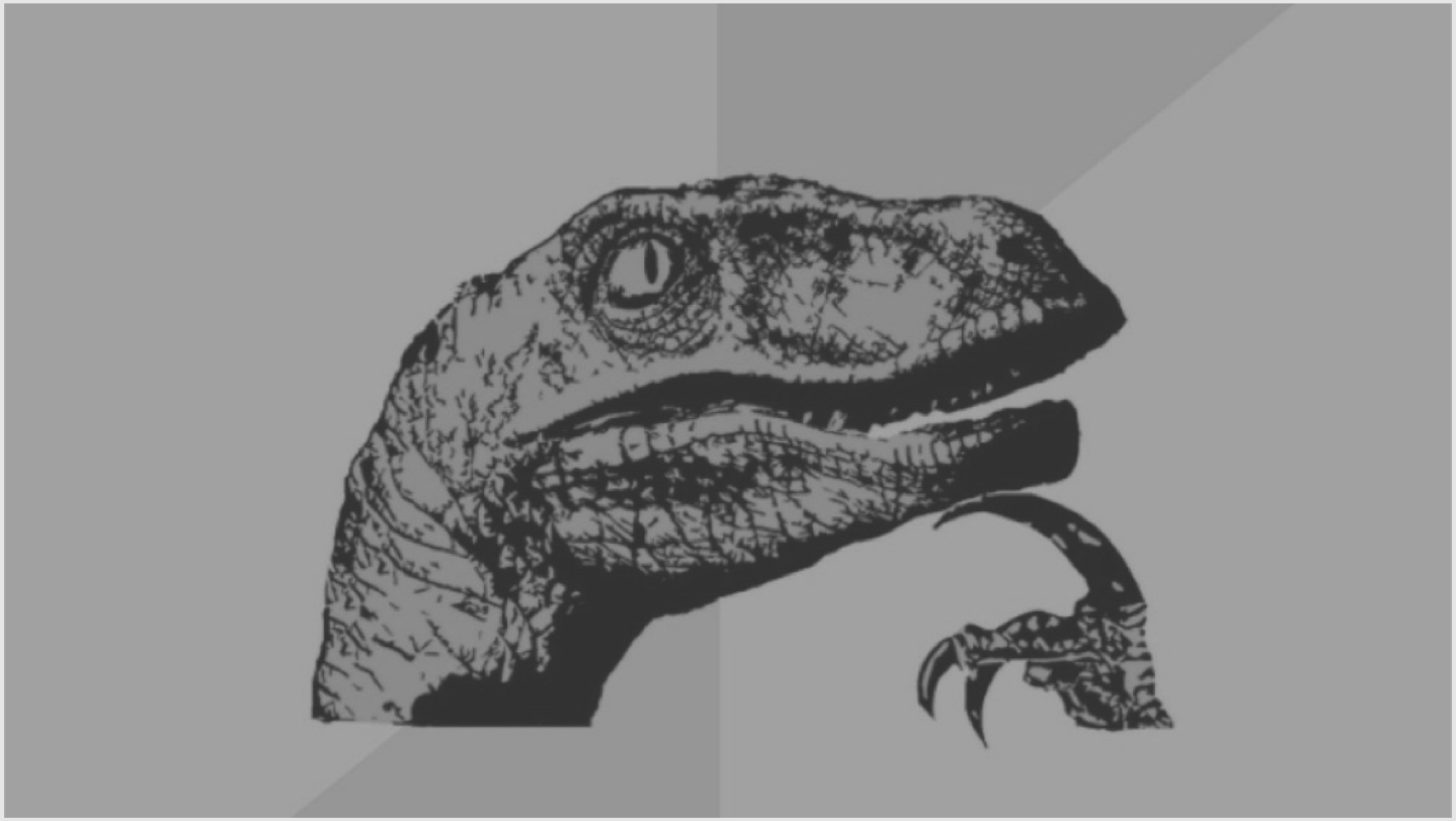
datasets for statistics class

Please open the following in Firefox or Chrome:

http://ww2.amstat.org/publications/jse/jse_data_archive.htm

Notice:

- No option to download all datasets
- No index to see webpage contents

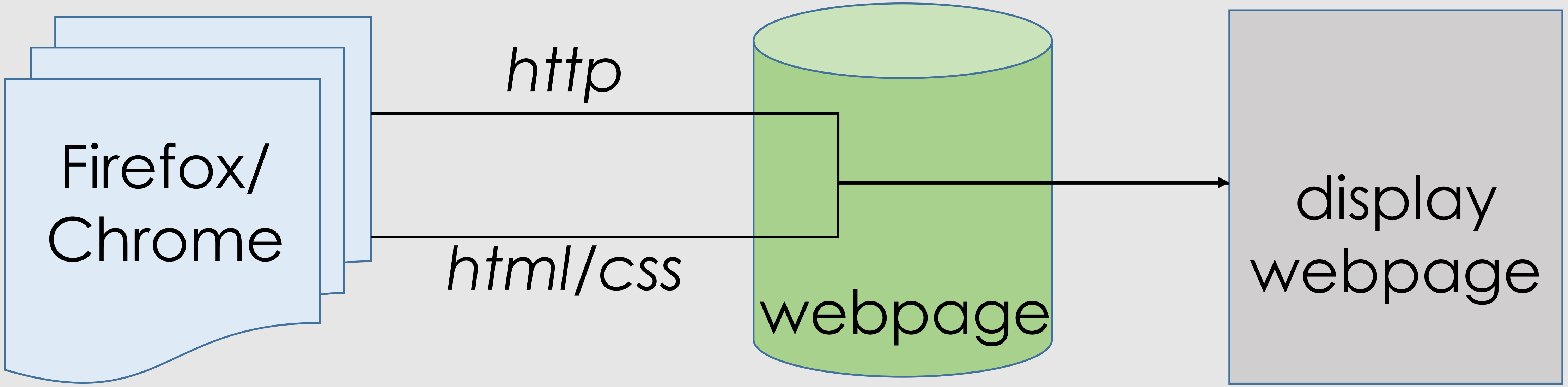


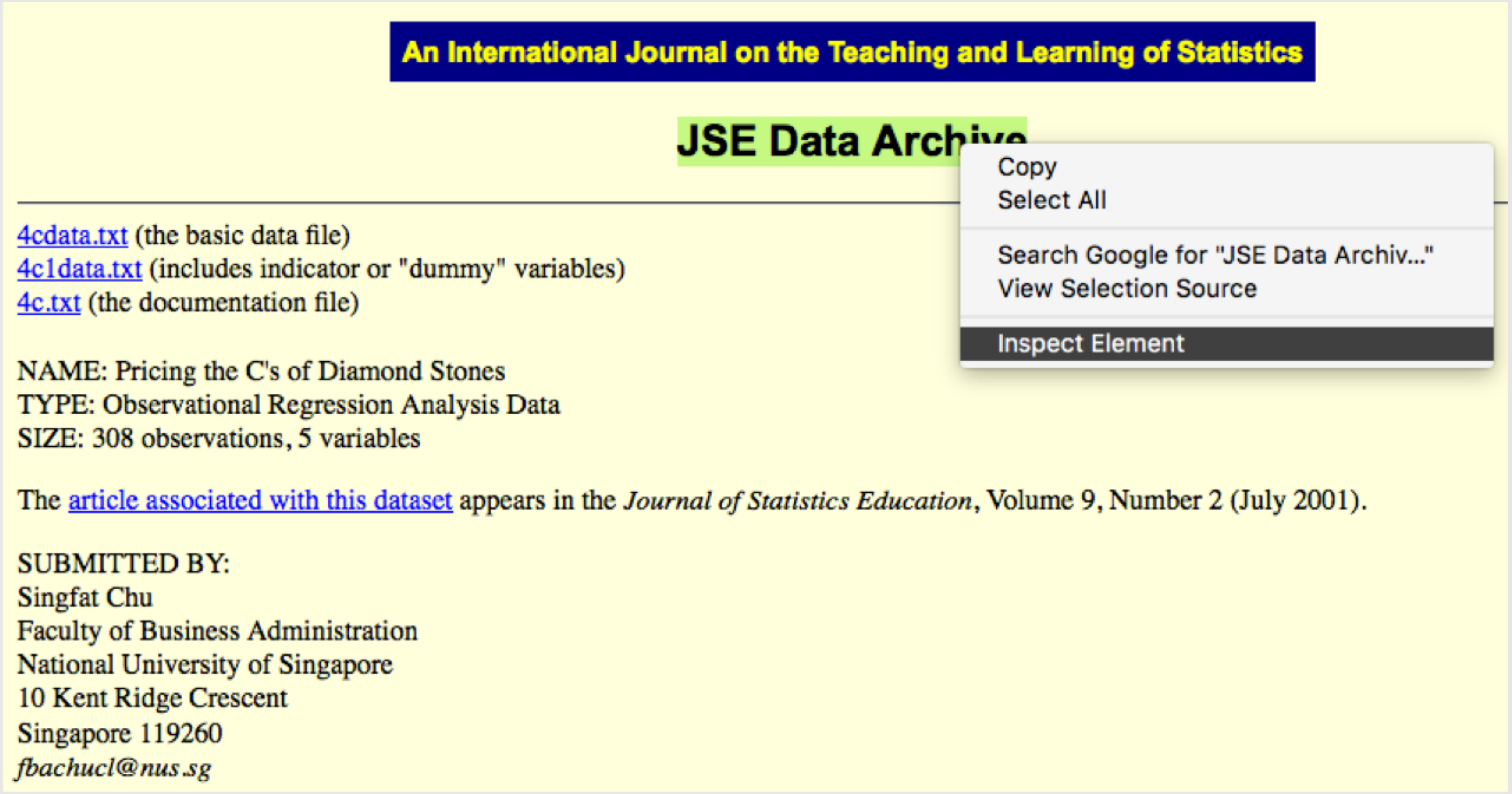
Review • **A**ccess • **P**arse • **T**ransform • st**OR**e

RAPTOR

Review	Structure of webpage
Access	HTML files on web server
Parse	HTML tags, attributes, etc.
Transform	Convert page content to desired format
St**OR**e	Write to text, CSV, or other file format

review: web browser behavior




An International Journal on the Teaching and Learning of Statistics

JSE Data Archive

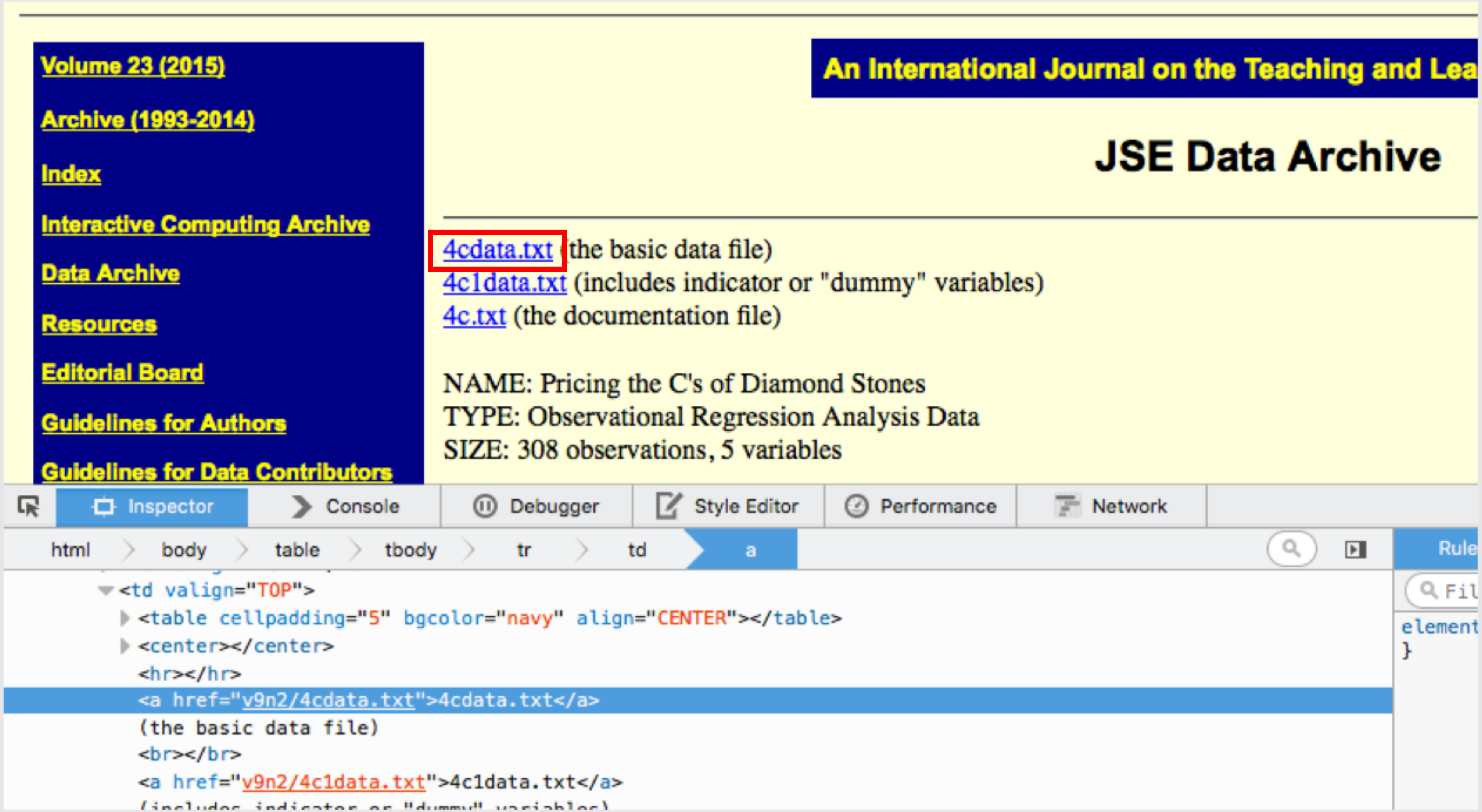
4cdata.txt (the basic data file)
4c1data.txt (includes indicator or "dummy" variables)
4c.txt (the documentation file)

NAME: Pricing the C's of Diamond Stones
TYPE: Observational Regression Analysis Data
SIZE: 308 observations, 5 variables

The article associated with this dataset appears in the *Journal of Statistics Education*, Volume 9, Number 2 (July 2001).

SUBMITTED BY:
Singfat Chu
Faculty of Business Administration
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260
fbachucl@nus.sg

review



Volume 23 (2015)

Archive (1993-2014)

Index

Interactive Computing Archive

Data Archive

Resources

Editorial Board

Guidelines for Authors

Guidelines for Data Contributors

An International Journal on the Teaching and Lea

JSE Data Archive

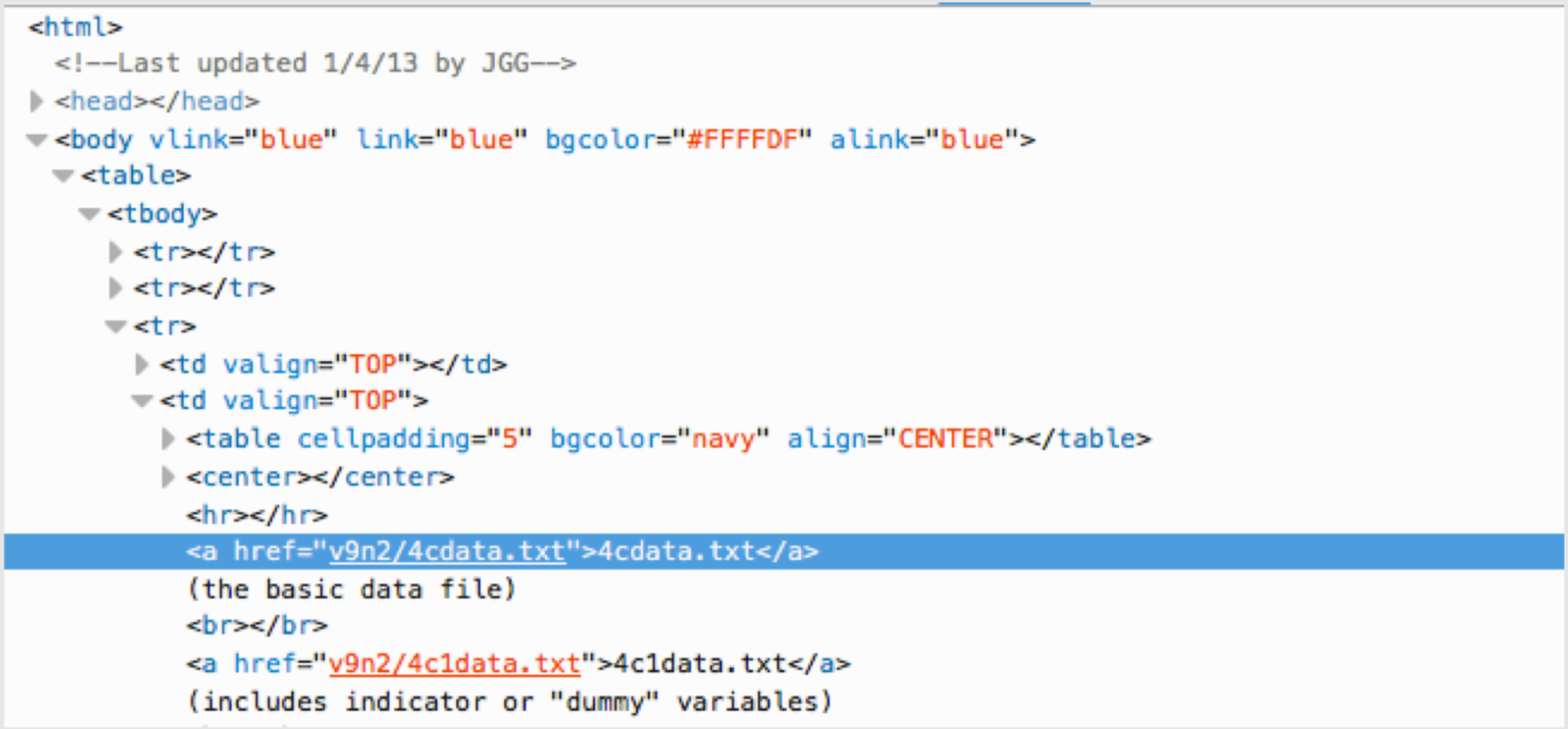
4cdata.txt (the basic data file)
4c1data.txt (includes indicator or "dummy" variables)
4c.txt (the documentation file)

NAME: Pricing the C's of Diamond Stones
TYPE: Observational Regression Analysis Data
SIZE: 308 observations, 5 variables

```
<td valign="TOP">
  <table cellpadding="5" bgcolor="navy" align="CENTER"></table>
  <center></center>
  <hr></hr>
  <a href="v9n2/4cdata.txt">4cdata.txt</a>
  (the basic data file)
  <br></br>
  <a href="v9n2/4c1data.txt">4c1data.txt</a>
```

```
element
}
```

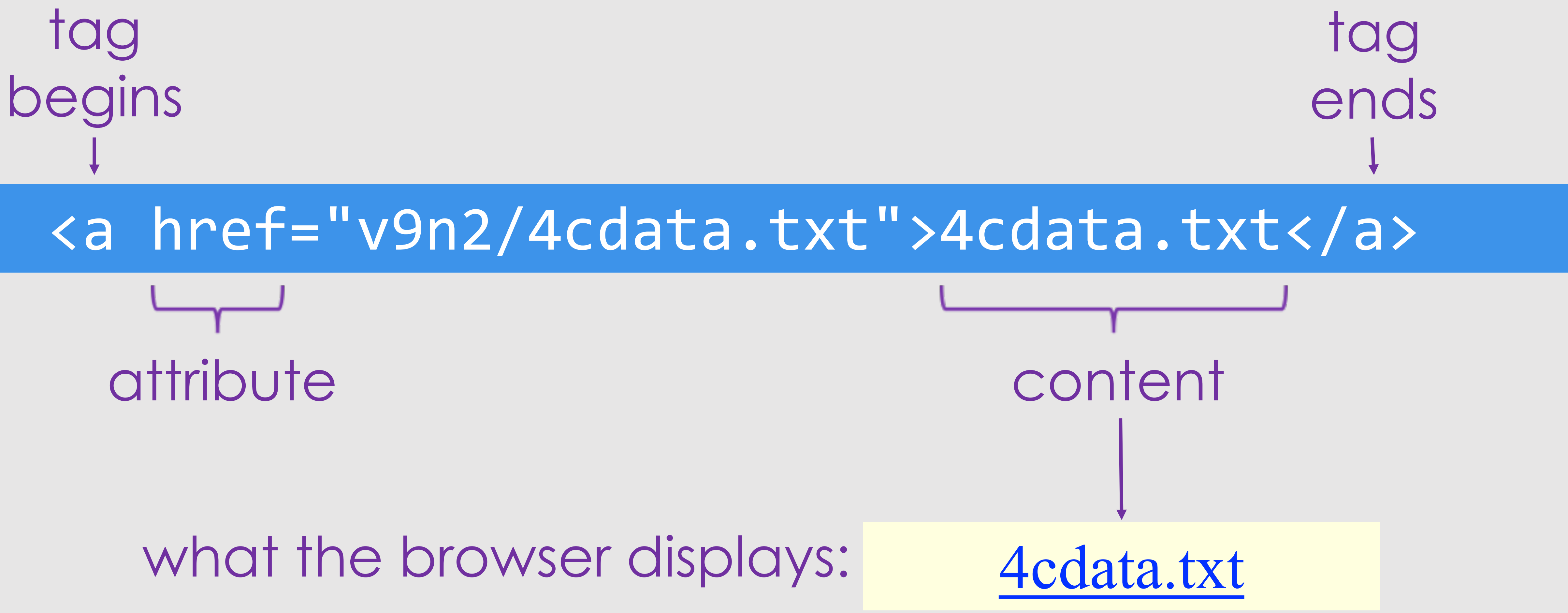
review



```
<html>
  <!--Last updated 1/4/13 by JGG-->
  <head></head>
  <body vlink="blue" link="blue" bgcolor="#FFFFDF" alink="blue">
    <table>
      <tbody>
        <tr></tr>
        <tr></tr>
        <tr>
          <td valign="TOP"></td>
          <td valign="TOP">
            <table cellpadding="5" bgcolor="navy" align="CENTER"></table>
            <center></center>
            <hr></hr>
            <a href="v9n2/4cdata.txt">4cdata.txt</a>
            (the basic data file)
            <br></br>
            <a href="v9n2/4c1data.txt">4c1data.txt</a>
            (includes indicator or "dummy" variables)
```

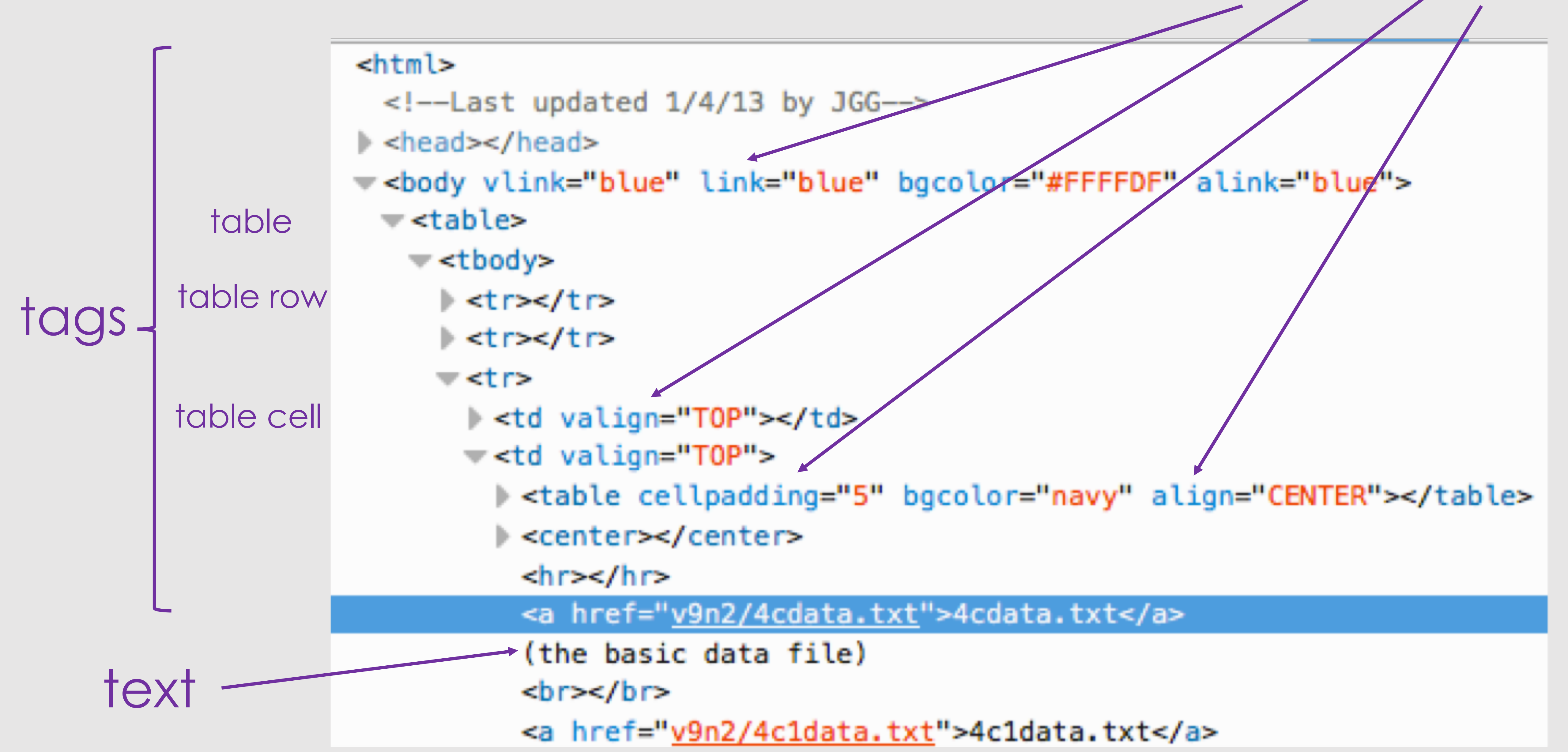
review

html elements



```
<a href="v9n2/4cdata.txt">4cdata.txt</a>
```

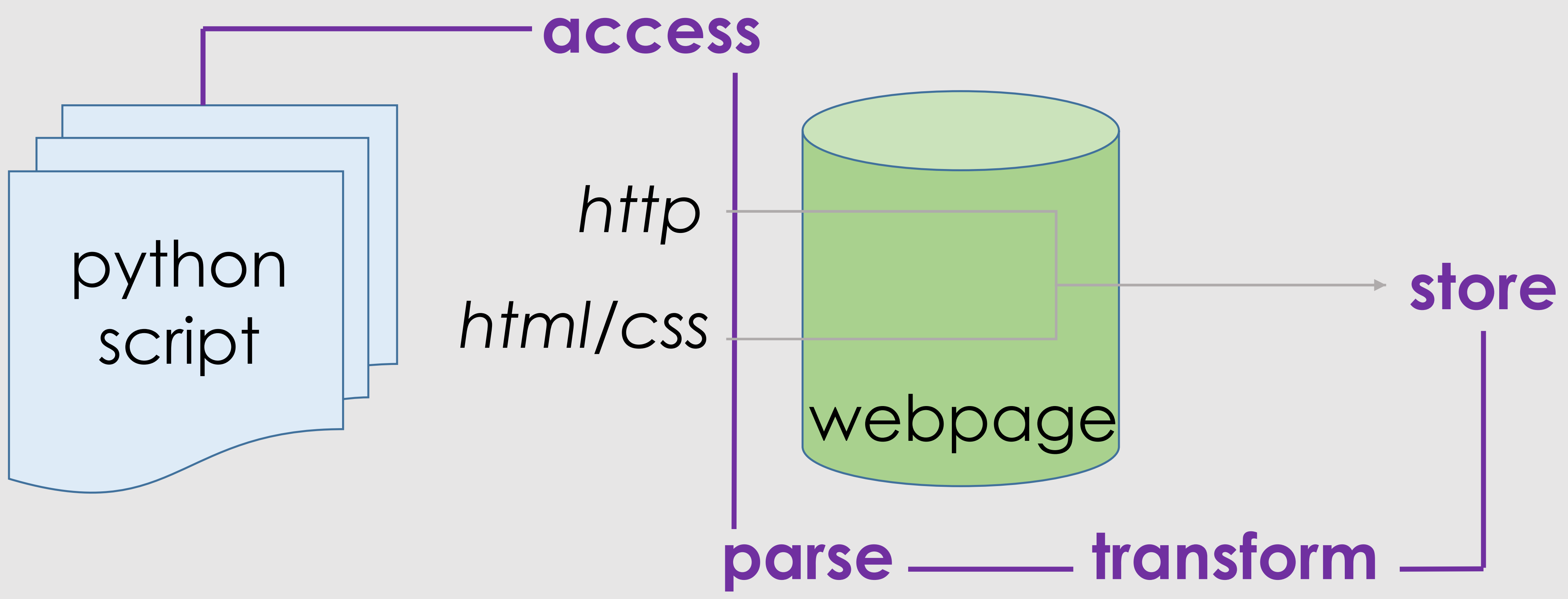
html elements



```
<html>
  <!--Last updated 1/4/13 by JGG-->
▸ <head></head>
▾ <body vlink="blue" link="blue" bgcolor="#FFFFDF" alink="blue">
  ▾ <table>
    ▾ <tbody>
      ▸ <tr></tr>
      ▸ <tr></tr>
      ▾ <tr>
        ▸ <td valign="TOP"></td>
        ▾ <td valign="TOP">
          ▸ <table cellpadding="5" bgcolor="navy" align="CENTER"></table>
          ▸ <center></center>
            <hr></hr>
            <a href="v9n2/4cdata.txt">4cdata.txt</a>
            (the basic data file)
            <br></br>
            <a href="v9n2/4c1data.txt">4c1data.txt</a>
```

take a 10-minute break

script workflow



access: request data from web server

Import statements allow you to add functions

```
import requests
import bs4
import csv

webpage = ‘http://www.amstat.org/...’
server_response = requests.get(webpage)
```

use the **get()** function from the **requests** package

parse

```
soup = bs4.BeautifulSoup(server_response.text)

link_info_list = []
for tag in soup.find_all(‘a’):
    link = tag[‘href’]
    name = tag.text
    if name[-3:] == ‘txt’:
        link_info_list.append({‘link’: link,
                               ‘name’: name})
```

Check every instance of the ‘a’ html tag to get the url and filename

Save the info for each link in its own dictionary inside the list

what is python doing?

1. Create an empty list: `link_info_list = []`
2. Find all the html chunks with ‘a’ tags: `soup.find_all(‘a’)`
3. Go to the first tag in the list:

    ```
    <a href="v9n2/4cdata.txt">4cdata.txt</a>
    ```

4. Assign the url to a variable called link and the text to a variable called name
5. If the last three letters in the value assigned to the name variable are `‘txt’`, proceed. (If not, go to the next tag.)

what is python doing?

6. Save the two variables as values in a dictionary
7. Add the dictionary to the list:

```
                                              link
link_info_list = [{‘link’: ‘v9n2/4cdata.txt’
                   ‘name’: ‘4cdata.txt’}]
                                          name
```

8. Repeat steps 3 through 7 until all tags have been checked

transform

```
host = ‘http://www.amstat.org/publications/jse/’
for dataset in link_info_list[:3]:
    url = host + dataset[‘link’]
    data_response = requests.get(url)
    if data_response.text[:5] == ‘NAME:’:
        dataset[‘type’] = ‘doc’
    else:
        dataset[‘type’] = ‘dat’
```

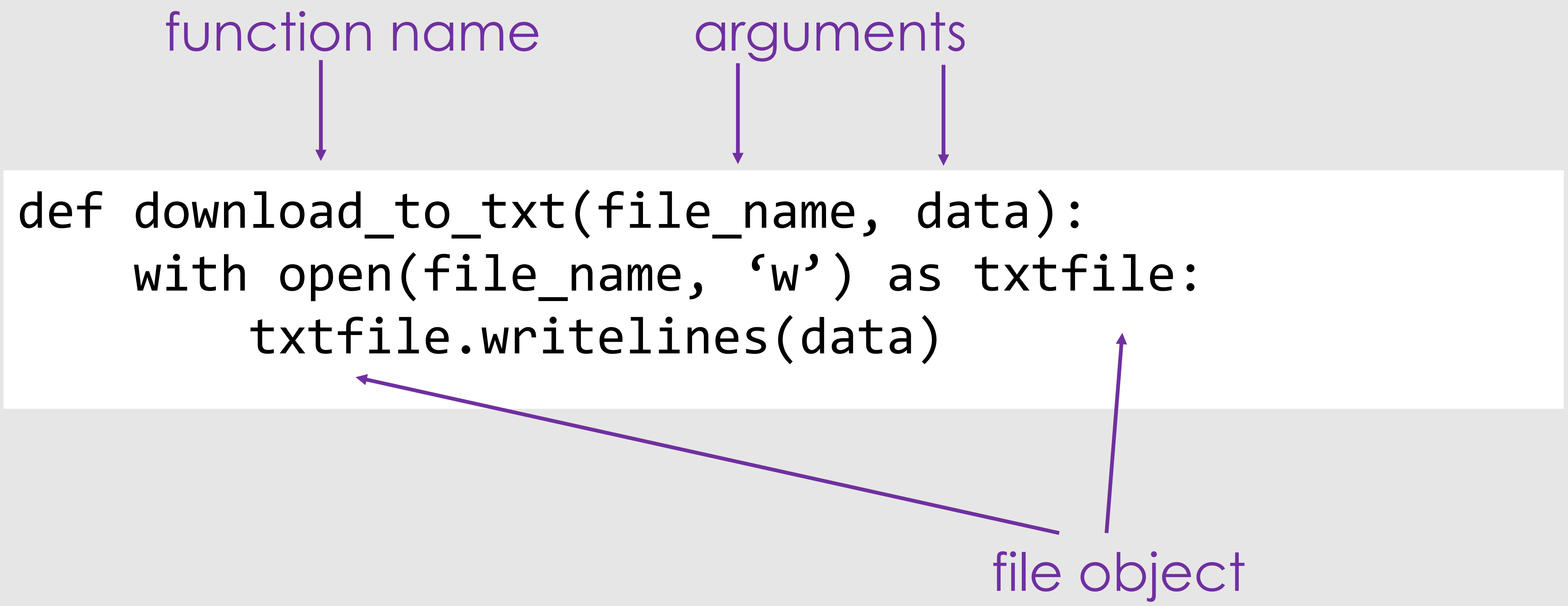
what is python doing?

1. Build the address for the link and assign it to the url variable:

```
url =‘http://www.amstat.org/publications/jse/v9n2/4cdata.txt’
```

2. Using the requests library, retrieve the web page information
3. If the text on the webpage starts with `‘NAME:’`, add a new key `‘type’` to the link’s dictionary with the value `‘doc’`
4. If not, add a new key `‘type’` to the link’s dictionary with the value `‘dat’`

store: helper function 1



```python
def download_to_txt(file_name, data):
    with open(file_name, ‘w’) as txtfile:
        txtfile.writelines(data)
```

store: helper function 2

```
def strip_extension(file_name):
    i = -1
    while i > -len(file_name):
        if file_name[i] == ‘.’:
            break
        else:
            i -= -1  # this is the same as i = i – 1
    return file_name[:i]
```

Note: We need to do something with the return value, e.g.,

```
stripped1 = strip_extension(‘my_file.txt’)
```

store

function call

```
for dataset in link_info_list[:3]:
    url = host + dataset[‘link’]
    data_response = requests.get(url)

    description = strip_extension(dataset[‘name’])
    filename = description + ‘_’ + dataset[‘type’] + ‘.txt’

    download_to_text(filename, data_response.text)
```

function call

store

```
with open(‘data_links.csv’, ‘w’) as csvfile:
    fieldnames = [‘link’, ‘name’, ‘type’]
    writer = csv.DictWriter(csvfile, fieldnames)

    writer.writeheader()
    for link in link_info_list:
        writer.writerow(link)
    print(‘Links added: ’ + str(len(link_info_list)))
```

run your web scraper

From IDE:
Run the program in the interpreter: **F5** or **Run** or ▶

OR

From the Command Line:

```
$ python3 scraper.py
```

Questions?

linkedin: nalettebrodnax
github: nmbrodnax
twitter: @nbrodnax

email: nmbrodnax@gmail.com
web: www.nalettebrodnax.com