Center of Excellence for Women in Technology

An Introduction to Web Scraping with Python and DataCamp

Olga Scrivner, Research Scientist, CNS, CEWIT

WIM, February 23, 2018

Cyberinfrastructure for Network Science Center
cns.iu.edu

Objectives

Materials: DataCamp.com

- Review: Importing files
- Accessing Web
- Review: Processing text
- Practice, practice, practice!

Credits

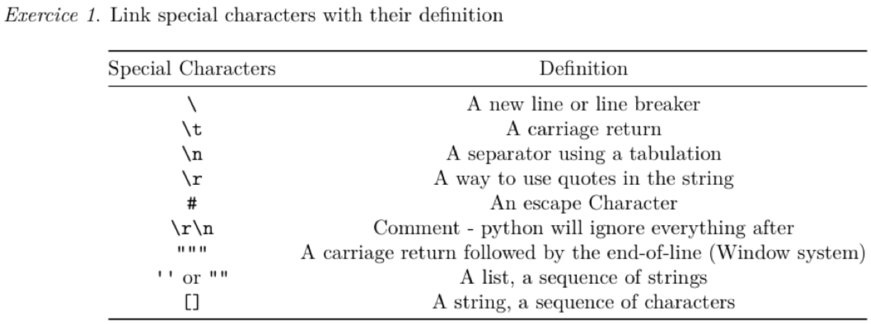
- Hugo Bowne-Anderson - *Importing Data in Python (Part 1 and Part 2)*
- Jeri Wieringa - *Intro to Beautiful Soup*

Importing Files

File Types: Text

- Text files are structured as a sequence of lines
- Each line includes a sequence of characters
- Each line is terminated with a special character **End of Line**

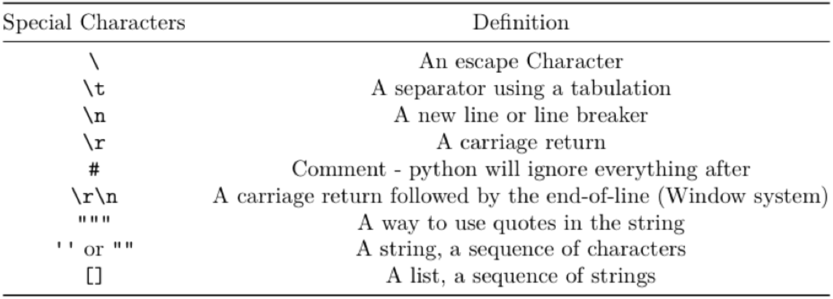
Special Characters: Review



Exercice 1. Link special characters with their definition

Special Characters	Definition
`\`	A new line or line breaker
`\t`	A carriage return
`\n`	A separator using a tabulation
`\r`	A way to use quotes in the string
`#`	An escape Character
`\r\n`	Comment - python will ignore everything after
`"""`	A carriage return followed by the end-of-line (Window system)
`''` or `""`	A list, a sequence of strings
`[]`	A string, a sequence of characters

Special Characters: Answers



Special Characters	Definition
`\`	An escape Character
`\t`	A separator using a tabulation
`\n`	A new line or line breaker
`\r`	A carriage return
`#`	Comment - python will ignore everything after
`\r\n`	A carriage return followed by the end-of-line (Window system)
`"""`	A way to use quotes in the string
`''` or `""`	A string, a sequence of characters
`[]`	A list, a sequence of strings

Modes

- **Reading Mode**
  - **‘r’**
- **Writing Mode**
  - **‘w’**

Modes

- **Reading Mode**
  - **‘r’**
- **Writing Mode**
  - **‘w’**

Quiz question: Why do we use quotes with **‘r’** and **‘w’**?

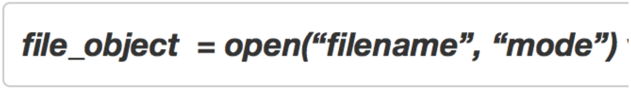
Modes

- **Reading Mode**
  - **'r'**
- **Writing Mode**
  - **'w'**

Quiz question: Why do we use quotes with **'r'** and **'w'**?
Answer: **'r'** and **'w'** are one-character strings

Open - Close

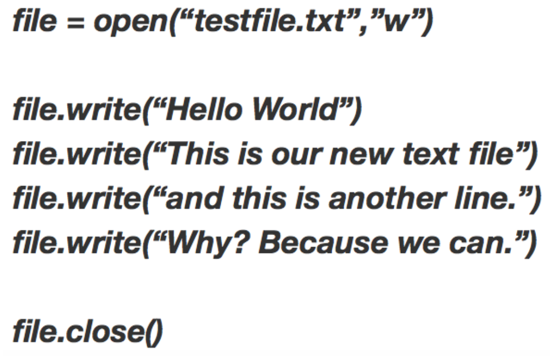
- Open File - **open(name, mode)**



```
file_object  = open("filename", "mode")
```

  - **name = 'filename'**
  - **mode = 'r' or mode = 'w'**

Open New File



```
file = open(“testfile.txt”,”w”)

file.write(“Hello World”)
file.write(“This is our new text file”)
file.write(“and this is another line.”)
file.write(“Why? Because we can.”)

file.close()
```

Open New File

```
file = open(“testfile.txt”,”w”)

file.write(“Hello World”)
file.write(“This is our new text file”)
file.write(“and this is another line.”)
file.write(“Why? Because we can.”)

file.close()
```

```
Hello World
This is our new text file
and this is another line.
Why? Because we can.
```

Read File

- Read the Entire File - **filename.read()**

```
file = open("testfile.text", "r")
print (file.read())
```

- Read ONE Line - **filename.readline()**

```
print (file.readline())
print (file.readline(3))
```

- Return the FIRST line
- Return the THIRD line

- Read lines - **filename.readlines()**

```
['Hello World', 'This is our new text file', 'and this is another line.', 'Why? Because we can.']
```

Read File

- Read the Entire File - **filename.read()**

```
file = open("testfile.text", "r")
print (file.read())
```

- Read ONE Line - **filename.readline()**

```
print (file.readline())
print (file.readline(3))
```

- Return the FIRST line
- Return the THIRD line

- Read lines - **filename.readlines()**

```
['Hello World', 'This is our new text file', 'and this is another line.', 'Why? Because we can.']
```

What type of object and what is the length of this object?

Python Libraries

Import Modules (Libraries)

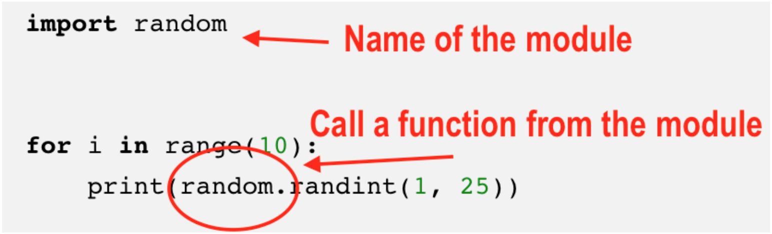
- **Beautiful Soup**
- **urllib**
- More in next slides ...

For installation - `https://programminghistorian.org/lessons/intro-to-beautiful-soup`

Review: Module I

To use external functions (**modules**), we need to import them:

1. Declare it at the top of the code
2. Use **import**
3. Call the module

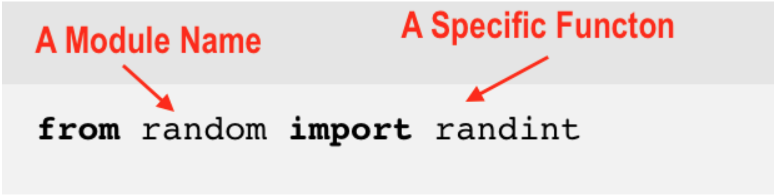


```
import random


for i in range(10):
    print(random.randint(1, 25))
```

Review: Modules II

To refer and import a specific function from the module

1. Declare it at the top pf the code
2. Use **from import**



```
from random import randint
```

3. Call the **randint** function from **random** module:

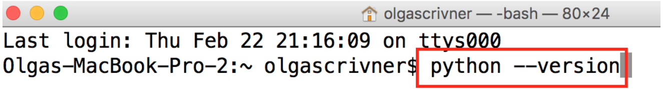
random.randint()

How to Import Packages with Modules

1. Install via a **terminal** or **console**
   - Type **command prompt** in window search
   - Type **terminal** in Mac search

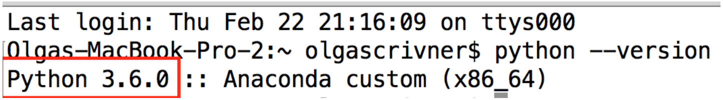
How to Import Packages with Modules

1. Install via a **terminal** or **console**
   - Type **command prompt** in window search
   - Type **terminal** in Mac search
2. Check your Python Version



```
Last login: Thu Feb 22 21:16:09 on ttys000
Olgas-MacBook-Pro-2:~ olgascrivner$ python --version
```

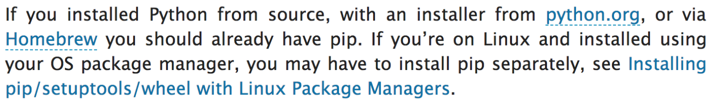
3. Click **return/enter**



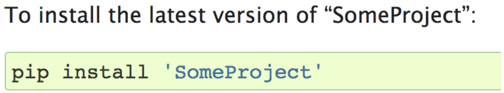
```
Last login: Thu Feb 22 21:16:09 on ttys000
Olgas-MacBook-Pro-2:~ olgascrivner$ python --version
Python 3.6.0 :: Anaconda custom (x86_64)
```

Python 2 (pip) or Python 3 (pip3)

pip or **pip3** - a tool for installing Python packages


If you installed Python from source, with an installer from python.org, or via Homebrew you should already have pip. If you're on Linux and installed using your OS package manager, you may have to install pip separately, see Installing pip/setuptools/wheel with Linux Package Managers.

To check if pip is installed:


To install the latest version of "SomeProject":

```
pip install 'SomeProject'
```

https://packaging.python.org/tutorials/installing-packages/

To install the latest version of "SomeProject":

```
pip install 'SomeProject'
```

Web Scraping Workflow

Web Concept

1. Import the necessary modules (functions)
2. Specify URL
3. Send a REQUEST
4. Catch RESPONSE
5. Return HTML as a STRING
6. Close the RESPONSE

URLs

URLs

1. URL - Uniform/Universal Resource Locator
2. A URL for web addresses consists of two parts:
   2.1 Protocol identifier - http: or https:
   2.2 Resource name - datacamp.com

URLs

1. URL - Uniform/Universal Resource Locator
2. A URL for web addresses consists of two parts:
   2.1 Protocol identifier - http: or https:
   2.2 Resource name - datacamp.com
3. HTTP - HyperText Transfer Protocol
4. HTTPS - more secure form of HTTP
5. Going to a website = sending HTTP request (GET request)
6. HTML - HyperText Markup Language

URLLIB package

Provide interface for getting data across the web. Instead of file names we use URLS

Step 1 Install the package urllib (**pip install urllib**)

Step 2 Import the function **urlretrieve** - to RETRIEVE urls during the REQUEST

Step 3 Create a variable **url** and provide the url link

url = ‘https:somepage’

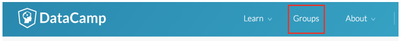
Step 4 Save the retrieved document locally

Step 5 Read the file

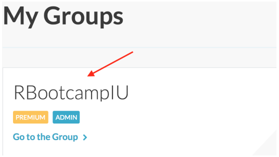
Your Turn - DataCamp

DataCamp.com - create a free account using IU email

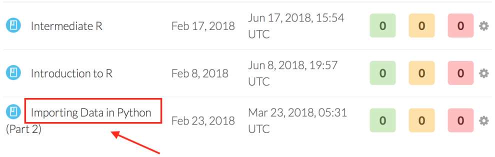
1. Log in
2. Select Groups



3. Select **RBootcampIU** - see Jennifer if you do not see it

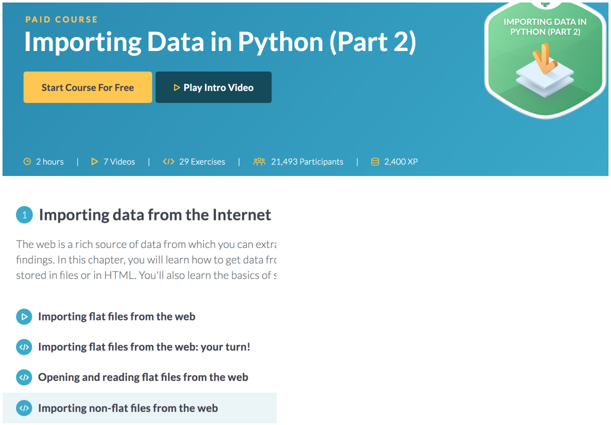


4. Go to Assignments and select Importing Data in Python



Intermediate R	Feb 17, 2018	Jun 17, 2018, 15:54 UTC	0	0	0
Introduction to R	Feb 8, 2018	Jun 8, 2018, 19:57 UTC	0	0	0
Importing Data in Python (Part 2)	Feb 23, 2018	Mar 23, 2018, 05:31 UTC	0	0	0

Today's Practice



PAID COURSE

Importing Data in Python (Part 2)

Start Course For Free
Play Intro Video
IMPORTING DATA IN PYTHON (PART 2)
2 hours | 7 Videos | 29 Exercises | 21,493 Participants | 2,400 XP

1 Importing data from the Internet

The web is a rich source of data from which you can extra
findings. In this chapter, you will learn how to get data fro
stored in files or in HTML. You'll also learn the basics of s

- Importing flat files from the web
- Importing flat files from the web: your turn!
- Opening and reading flat files from the web
- Importing non-flat files from the web

Importing Flat Files

```python
# Import package
from ____ import ____

# Import pandas
import pandas as pd
                  A library for data structure and analysis
# Assign url of file: url

  url = ?

# Save file locally
  urlretrieve(?, ?)

# Read file into a DataFrame and print its head
df = pd.read_csv('winequality-red.csv', sep=';')
print(df.head())
```

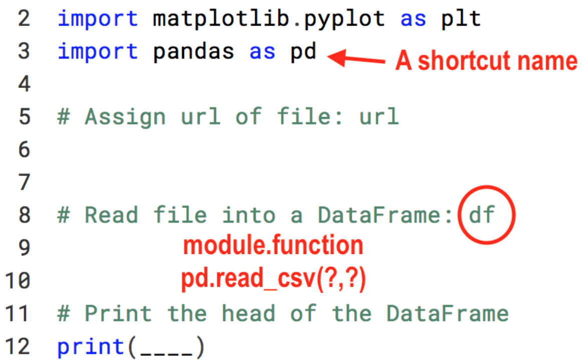
urlretrieve has two arguments: url (input) and file name (output)

Example: **urlretrieve(url, 'file.name')**

Importing Flat Files

```python
# Import package
from urllib.request import urlretrieve

# Import pandas
import pandas as pd

# Assign url of file: url
url = 'https://s3.amazonaws.com/assets.datacamp.com/production
/course_1606/datasets/winequality-red.csv'

# Save file locally
urlretrieve(url,'winequality-red.csv')

# Read file into a DataFrame and print its head
df = pd.read_csv('winequality-red.csv', sep=';')
print(df.head())

```

Opening and Reading Files

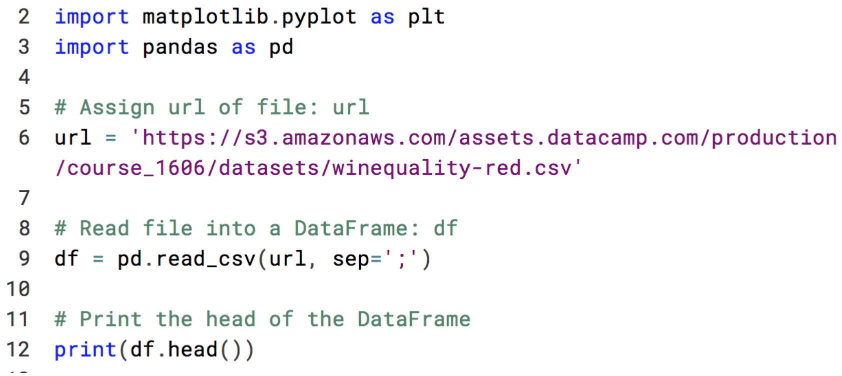


```python
import matplotlib.pyplot as plt
import pandas as pd

# Assign url of file: url


# Read file into a DataFrame: df


# Print the head of the DataFrame
print(____)
```

read_csv has two arguments: url and sep (separator)

pd.head()

Opening and Reading Files

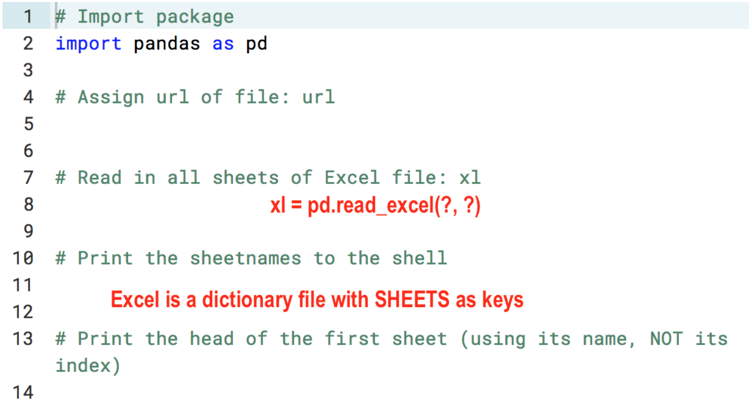


```
import matplotlib.pyplot as plt
import pandas as pd

# Assign url of file: url
url = 'https://s3.amazonaws.com/assets.datacamp.com/production
/course_1606/datasets/winequality-red.csv'

# Read file into a DataFrame: df
df = pd.read_csv(url, sep=';')

# Print the head of the DataFrame
print(df.head())
```

read_csv has two arguments: url and sep (separator)

pd.head()

Importing Non-flat Files



```python
# Import package
import pandas as pd

# Assign url of file: url


# Read in all sheets of Excel file: xl
                          xl = pd.read_excel(?, ?)

# Print the sheetnames to the shell

      Excel is a dictionary file with SHEETS as keys
# Print the head of the first sheet (using its name, NOT its
index)

```

read_excel has two arguments: url and sheetname

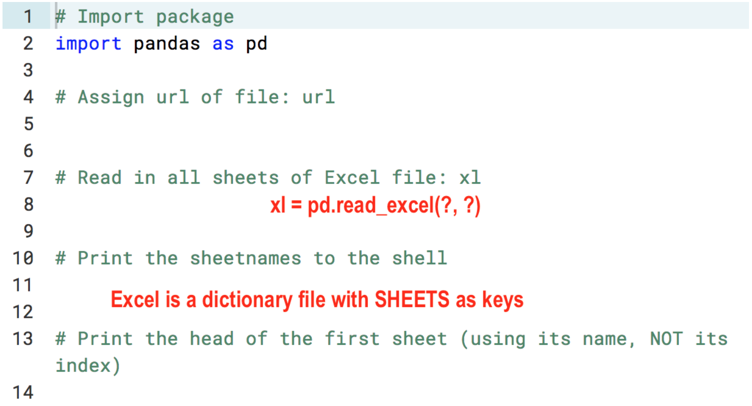
To read all sheets, sheetname = None

Let's use a sheetname '1700'

Importing Non-flat Files

```python
# Import package
import pandas as pd

# Assign url of file: url
url = 'http://s3.amazonaws.com/assets.datacamp.com/course
/importing_data_into_r/latitude.xls'

# Read in all sheets of Excel file: xl
xl = pd.read_excel(url, sheetname=None)

# Print the sheetnames to the shell
print(xl.keys())

# Print the head of the first sheet (using its name, NOT its
index)
print(xl['1700'].head())
```

HTTP Requests



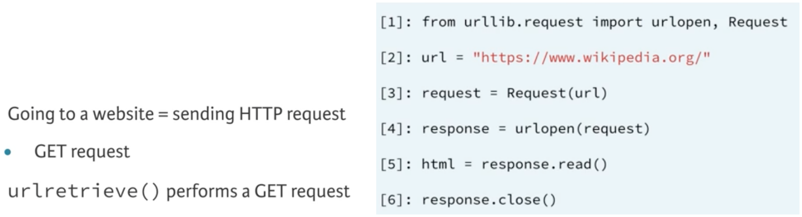
```
# Import package
import pandas as pd

# Assign url of file: url


# Read in all sheets of Excel file: xl
                    xl = pd.read_excel(?, ?)

# Print the sheetnames to the shell

     Excel is a dictionary file with SHEETS as keys

# Print the head of the first sheet (using its name, NOT its
index)

```

read_excel has two arguments: url and sheetname

To read all sheets, sheetname = None

Let's use a sheetname '1700'

GET request



Going to a website = sending HTTP request

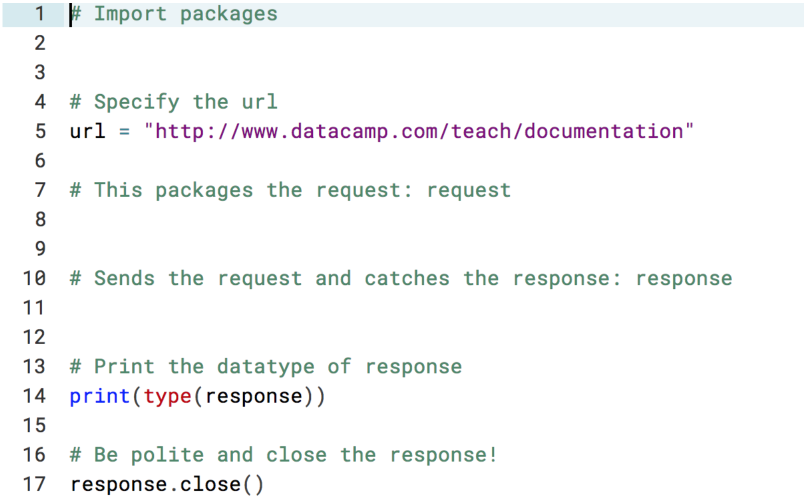
- GET request

`urlretrieve()` performs a GET request

```
[1]: from urllib.request import urlopen, Request

[2]: url = "https://www.wikipedia.org/"

[3]: request = Request(url)

[4]: response = urlopen(request)

[5]: html = response.read()

[6]: response.close()
```

Import **request** package

HTTP with urllib

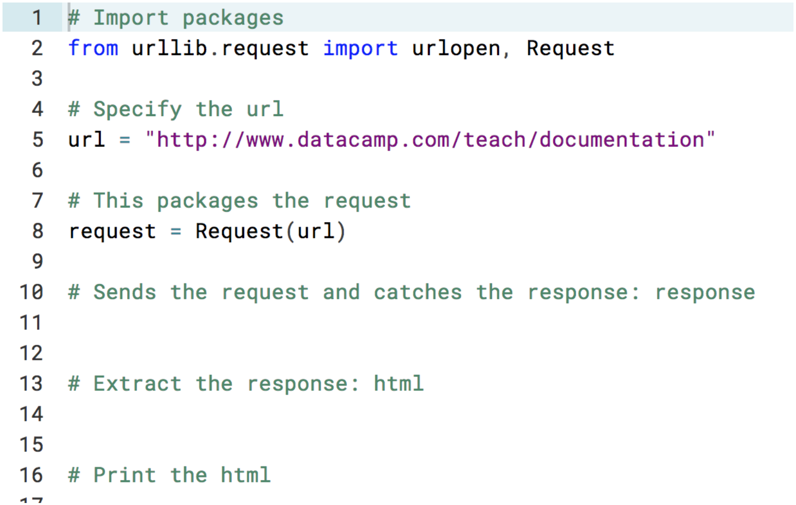


```python
# Import packages


# Specify the url
url = "http://www.datacamp.com/teach/documentation"

# This packages the request: request


# Sends the request and catches the response: response


# Print the datatype of response
print(type(response))

# Be polite and close the response!
response.close()
```

HTTP with urllib

```python
# Import packages
from urllib.request import urlopen, Request

# Specify the url
url = "http://www.datacamp.com/teach/documentation"

# This packages the request: request
request = Request(url)

# Sends the request and catches the response: response
response = urlopen(request)

# Print the datatype of response
print(type(response))

# Be polite and close the response!
response.close()
```

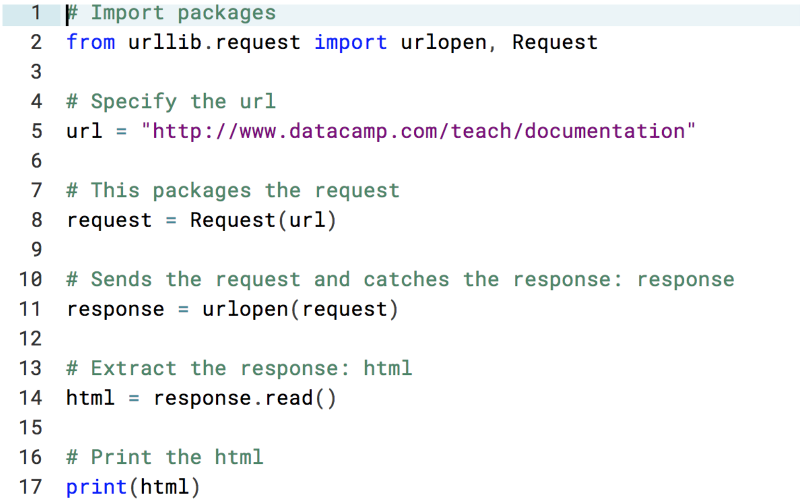
Print HTTP with urllib



```python
# Import packages
from urllib.request import urlopen, Request

# Specify the url
url = "http://www.datacamp.com/teach/documentation"

# This packages the request
request = Request(url)

# Sends the request and catches the response: response


# Extract the response: html


# Print the html
```

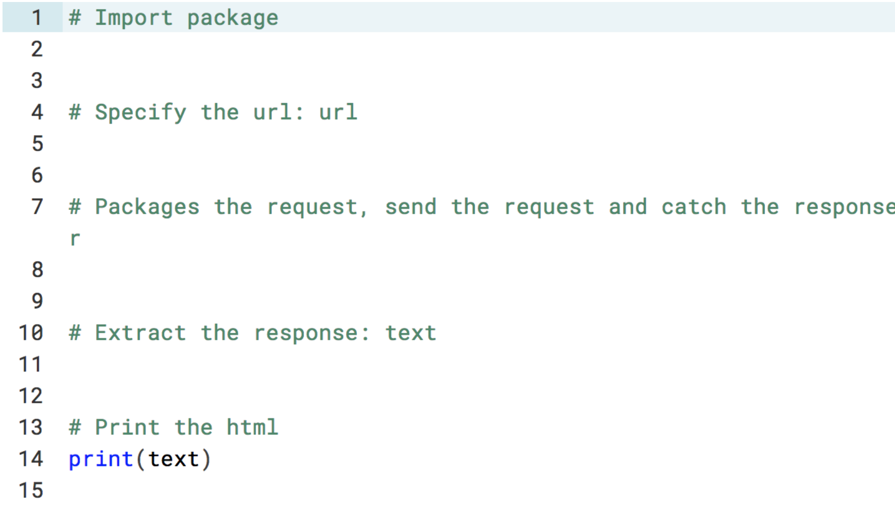
Use **response.read()**

Print HTTP with urllib



```python
# Import packages
from urllib.request import urlopen, Request

# Specify the url
url = "http://www.datacamp.com/teach/documentation"

# This packages the request
request = Request(url)

# Sends the request and catches the response: response
response = urlopen(request)

# Extract the response: html
html = response.read()

# Print the html
print(html)
```

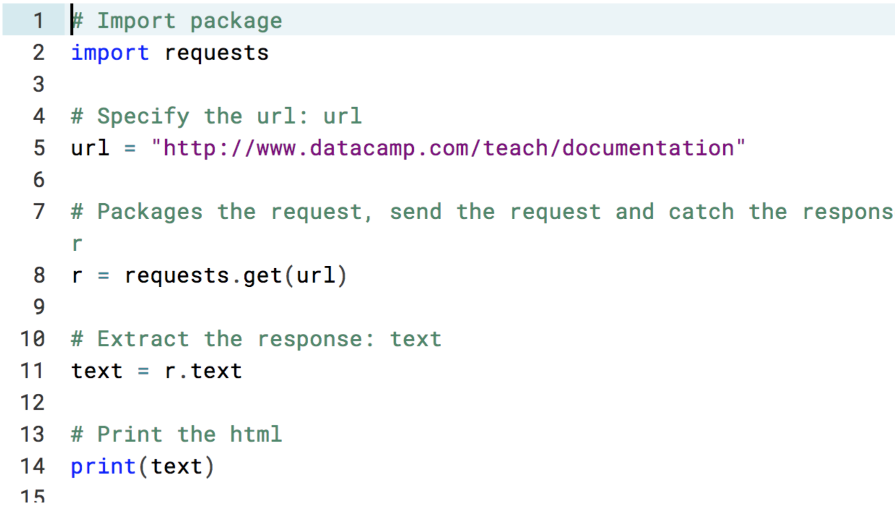
Return Web as a String



```python
# Import package


# Specify the url: url


# Packages the request, send the request and catch the respons
r


# Extract the response: text


# Print the html
print(text)

```

Use **r.text**

Return Web as a String



```python
# Import package
import requests

# Specify the url: url
url = "http://www.datacamp.com/teach/documentation"

# Packages the request, send the request and catch the respons
r
r = requests.get(url)

# Extract the response: text
text = r.text

# Print the html
print(text)

```

Scraping Web - HTML

Mix of unstructured and structured data

Structured data:

- Has pre-defined data model, or
- Organized in a defined manner

Unstructured data: neither of these properties

Scraping Web - HTML

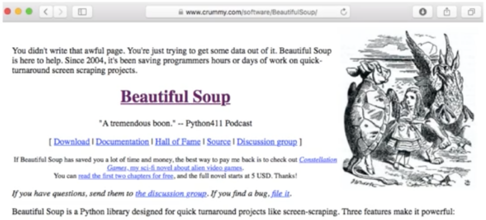
Mix of unstructured and structured data

Structured data:

- Has pre-defined data model, or
- Organized in a defined manner

Unstructured data: neither of these properties

- Parse and extract structured data from HTML



You didn't write that awful page. You're just trying to get some data out of it. Beautiful Soup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Beautiful Soup

"A tremendous boon." -- Python411 Podcast

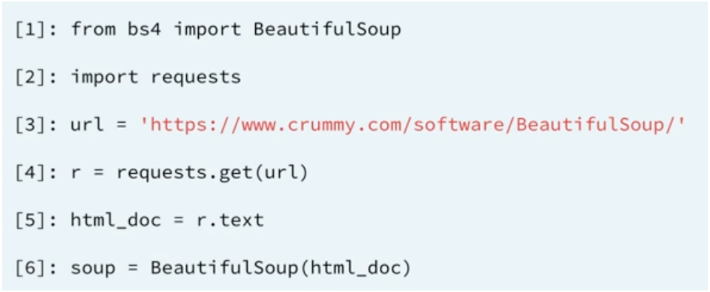
[Download | Documentation | Hall of Fame | Source | Discussion group]

If Beautiful Soup has saved you a lot of time and money, the best way to pay me back is to check out *Constellation Games*, my sci-fi novel about alien video games.
You can read the first two chapters for free, and the full novel starts at 5 USD. Thanks!

If you have questions, send them to the discussion group. If you find a bug, file it.

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

Scraping Web - BeautifulSoup Workflow

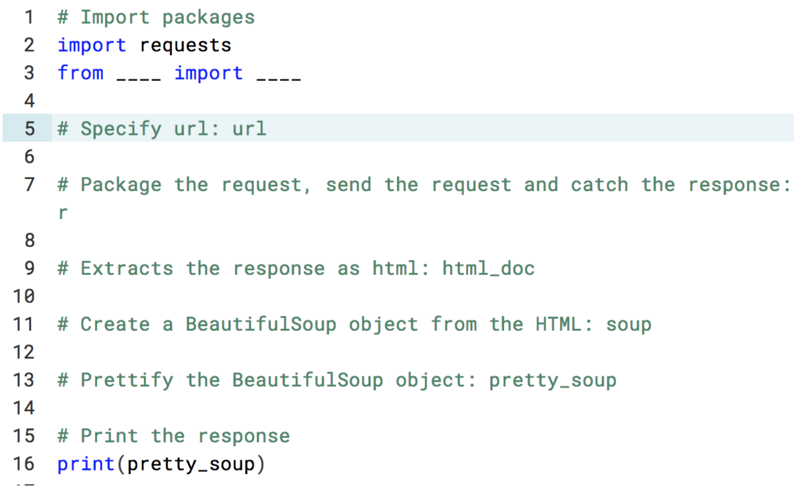


```
[1]: from bs4 import BeautifulSoup

[2]: import requests

[3]: url = 'https://www.crummy.com/software/BeautifulSoup/'

[4]: r = requests.get(url)

[5]: html_doc = r.text

[6]: soup = BeautifulSoup(html_doc)
```

Many Useful Functions

- soup.title
- soup.get_text()
- soup.find_all('a')

Parsing HTML with BeautifulSoup

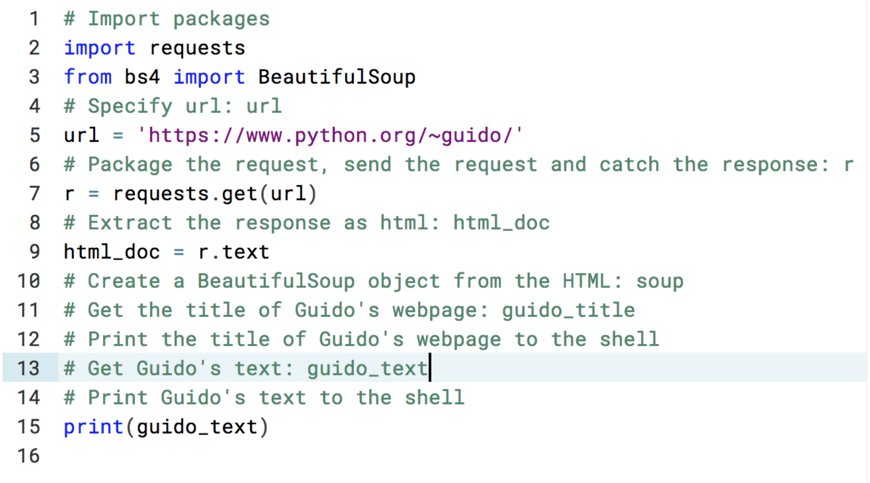


```python
# Import packages
import requests
from ____ import ____

# Specify url: url

# Package the request, send the request and catch the response: r

# Extracts the response as html: html_doc

# Create a BeautifulSoup object from the HTML: soup

# Prettify the BeautifulSoup object: pretty_soup

# Print the response
print(pretty_soup)
```

Parsing HTML with BeautifulSoup

```python
# Import packages
import requests
from bs4 import BeautifulSoup

# Specify url: url
url = 'https://www.python.org/~guido/'
# Package the request, send the request and catch the response: r
r = requests.get(url)
# Extracts the response as html: html_doc
html_doc = r.text
# Create a BeautifulSoup object from the HTML: soup
soup = BeautifulSoup(html_doc)
# Prettify the BeautifulSoup object: pretty_soup
pretty_soup = soup.prettify()
# Print the response
print(pretty_soup)
```

Turning a Webpage into Data with BeautifulSoup

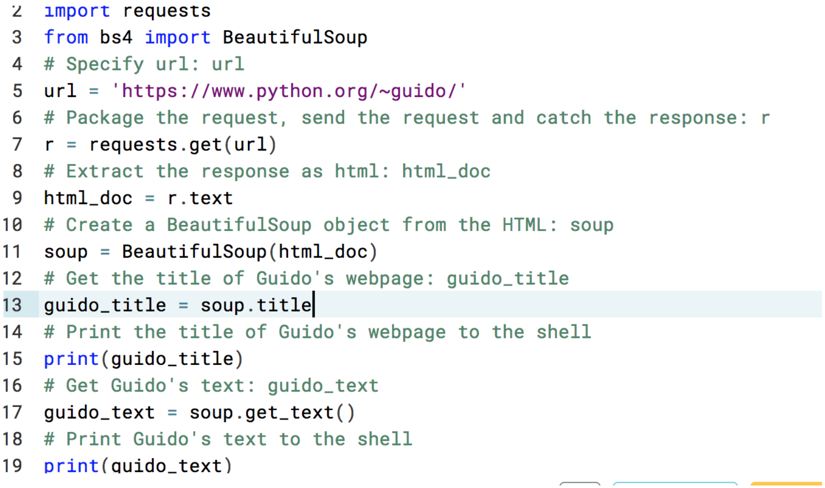


```python
# Import packages
import requests
from bs4 import BeautifulSoup
# Specify url: url
url = 'https://www.python.org/~guido/'
# Package the request, send the request and catch the response: r
r = requests.get(url)
# Extract the response as html: html_doc
html_doc = r.text
# Create a BeautifulSoup object from the HTML: soup
# Get the title of Guido's webpage: guido_title
# Print the title of Guido's webpage to the shell
# Get Guido's text: guido_text
# Print Guido's text to the shell
print(guido_text)

```

soup.title

soup.get_text()

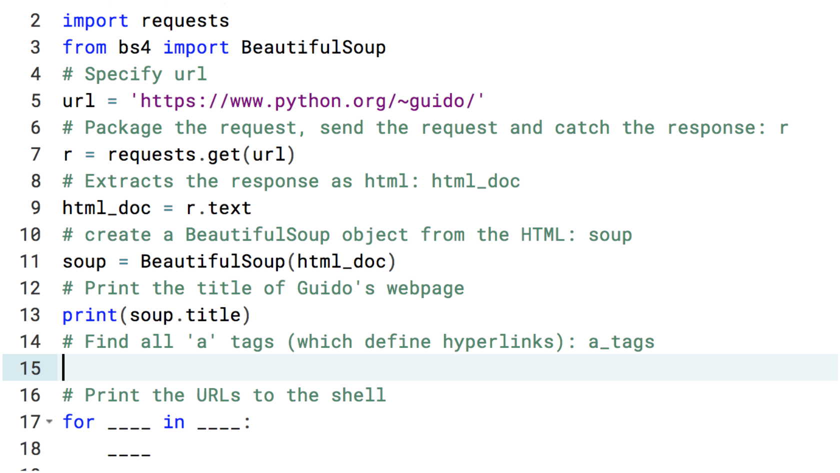
Turning a Webpage into Data with BeautifulSoup



```
import requests
from bs4 import BeautifulSoup
# Specify url: url
url = 'https://www.python.org/~guido/'
# Package the request, send the request and catch the response: r
r = requests.get(url)
# Extract the response as html: html_doc
html_doc = r.text
# Create a BeautifulSoup object from the HTML: soup
soup = BeautifulSoup(html_doc)
# Get the title of Guido's webpage: guido_title
guido_title = soup.title
# Print the title of Guido's webpage to the shell
print(guido_title)
# Get Guido's text: guido_text
guido_text = soup.get_text()
# Print Guido's text to the shell
print(guido_text)
```

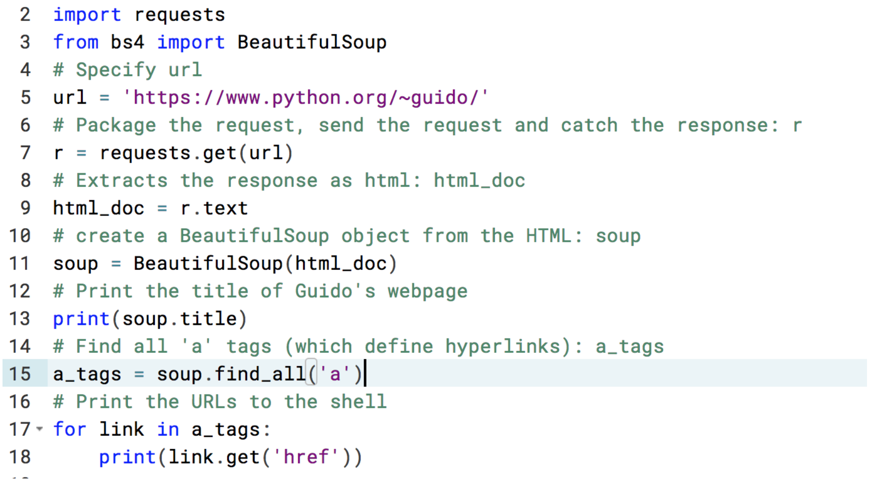
Turning a Webpage into Data - Hyperlinks

- HTML tag - <a>
- find_all('a')
- Collect all href: link.get('href')



```python
import requests
from bs4 import BeautifulSoup
# Specify url
url = 'https://www.python.org/~guido/'
# Package the request, send the request and catch the response: r
r = requests.get(url)
# Extracts the response as html: html_doc
html_doc = r.text
# create a BeautifulSoup object from the HTML: soup
soup = BeautifulSoup(html_doc)
# Print the title of Guido's webpage
print(soup.title)
# Find all 'a' tags (which define hyperlinks): a_tags

# Print the URLs to the shell
for ____ in ____:
    ____
```

Turning a Webpage into Data - Hyperlinks



```python
import requests
from bs4 import BeautifulSoup
# Specify url
url = 'https://www.python.org/~guido/'
# Package the request, send the request and catch the response: r
r = requests.get(url)
# Extracts the response as html: html_doc
html_doc = r.text
# create a BeautifulSoup object from the HTML: soup
soup = BeautifulSoup(html_doc)
# Print the title of Guido's webpage
print(soup.title)
# Find all 'a' tags (which define hyperlinks): a_tags
a_tags = soup.find_all('a')
# Print the URLs to the shell
for link in a_tags:
    print(link.get('href'))
```