1. Importing flat files from the web

1.1 Importing flat files from the web: your turn!

You are about to import your first file from the web! The flat file you will import will be `'winequality-red.csv'` from the University of California, Irvine's Machine Learning repository (http://archive.ics.uci.edu/ml/index.php). The flat file contains tabular data of physiochemical properties of red wine, such as pH, alcohol content and citric acid content, along with wine quality rating.

The URL of the file is

> 'https://s3.amazonaws.com/assets.datacamp.com/production/course_1606/datasets/winequality-red.csv' (https://s3.amazonaws.com/assets.datacamp.com/production/course_1606/datasets/winequality-red.csv')

After you import it, you'll check your working directory to confirm that it is there and then you'll load it into a `pandas` DataFrame.

Instructions

- Import the function `urlretrieve` from the subpackage urllib.request.
- Assign the URL of the file to the variable `url` .
- Use the function `urlretrieve()` to save the file locally as `'winequality-red.csv'` .
- Execute the remaining code to load `'winequality-red.csv'` in a pandas DataFrame and to print its head to the shell.

In [1]:

```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/wi


# 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())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  
\
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

1.2 Opening and reading flat files from the web

You have just imported a file from the web, saved it locally and loaded it into a DataFrame. If you just wanted to load a file from the web into a DataFrame without first saving it locally, you can do that easily using `pandas` . In particular, you can use the function `pd.read_csv()` with the URL as the first argument and the separator `sep` as the second argument.

The URL of the file, once again, is

> 'https://s3.amazonaws.com/assets.datacamp.com/production/course_1606/datasets/winequality-red.csv'
> (https://s3.amazonaws.com/assets.datacamp.com/production/course_1606/datasets/winequality-red.csv')

Instructions

- Assign the URL of the file to the variable `url`.
- Read file into a DataFrame df using `pd.read_csv()`, recalling that the separator in the file is `';'`.
- Print the head of the DataFrame `df`.
- Execute the rest of the code to plot histogram of the first feature in the DataFrame `df`.

In [2]:

```python
# Import packages
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/wi


# Read file into a DataFrame: df

df = pd.read_csv(url,sep=';')
# Print the head of the DataFrame
print(df.head())

# Plot first column of df
pd.DataFrame.hist(df.iloc[:, 0:1])
plt.xlabel('fixed acidity (g(tartaric acid)/dm$^3$)')
plt.ylabel('count')
plt.show()
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  
\
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

```
<Figure size 640x480 with 1 Axes>
```

1.3 Importing non-flat files from the web

Congrats! You've just loaded a flat file from the web into a DataFrame without first saving it locally using the pandas function `pd.read_csv()` . This function is super cool because it has close relatives that allow you to load all types of files, not only flat ones. In this interactive exercise, you'll use `pd.read_excel()` to import an Excel spreadsheet.

The URL of the spreadsheet is

'http://s3.amazonaws.com/assets.datacamp.com/course/importing_data_into_r/latitude.xls' (http://s3.amazonaws.com/assets.datacamp.com/course/importing_data_into_r/latitude.xls')

Your job is to use `pd.read_excel()` to read in all of its sheets, print the sheet names and then print the head of the first sheet using its name, not its index.

Note that the output of `pd.read_excel()` is a Python dictionary with sheet names as keys and corresponding DataFrames as corresponding values.

Instructions

- Assign the URL of the file to the variable `url` .
- Read the file in url into a dictionary `xl` using `pd.read_excel()` recalling that, in order to import all sheets you need to pass None to the argument `sheetname` .
- Print the names of the sheets in the Excel spreadsheet; these will be the keys of the dictionary `xl` .
- Print the head of the first sheet using the sheet name, not the index of the sheet! The sheet name is `'1700'`

In [3]:

```

# Import package
import pandas as pd

# Assign url of file: url

url = 'http://s3.amazonaws.com/assets.datacamp.com/course/importing_data_into_r/latitu

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

```

```
odict_keys(['1700', '1900'])
                 country       1700
0            Afghanistan  34.565000
1  Akrotiri and Dhekelia  34.616667
2                Albania  41.312000
3                Algeria  36.720000
4         American Samoa -14.307000
```

2. HTTP requests to import files from the web

2.1 Performing HTTP requests in Python using urllib

Now that you know the basics behind HTTP GET requests, it's time to perform some of your own. In this interactive exercise, you will ping our very own DataCamp servers to perform a GET request to extract information from our teach page, `"http://www.datacamp.com/teach/documentation"` .

In the next exercise, you'll extract the HTML itself. Right now, however, you are going to package and send the request and then catch the response.

Instructions

- Import the functions urlopen and Request from the subpackage `urllib.request`.
- Package the request to the url `"http://www.datacamp.com/teach/documentation"` using the function `Request()` and assign it to `request`.
- Send the request and catch the response in the variable response with the function `urlopen()`.
- Run the rest of the code to see the datatype of response and to `close` the connection!

In [4]:

```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 = urlopen(request)

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

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

```
<class 'http.client.HTTPResponse'>
```

Printing HTTP request results in Python using urllib

You have just packaged and sent a GET request to `"http://www.datacamp.com/teach/documentation"` and then caught the response. You saw that such a response is a `http.client.HTTPResponse object`. The question remains: what can you do with this response?

Well, as it came from an HTML page, you could read it to extract the HTML and, in fact, such a `http.client.HTTPResponse` object has an associated `read()` method. In this exercise, you'll build on your previous great work to extract the response and print the HTML.

Instructions

- Send the request and catch the response in the variable response with the function `urlopen()`, as in the previous exercise.
- Extract the response using the `read()` method and store the result in the variable `html`.
- Print the string `html`.
- Hit submit to perform all of the above and to close the response: be tidy!

In [5]:

```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 = urlopen(request)
# Extract the response: html

html = response.read()
# Print the html
print(html)

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

```
b'<!doctype html>\n<html lang="en" data-direction="ltr">\n  <head>\n    <l
ink href="https://fonts.intercomcdn.com" rel="preconnect" crossorigin>\n
<script src="https://www.googletagmanager.com/gtag/js?id=UA-39297847-9" as
ync="async" nonce="xd914cYFGKX5DNQBrX721MCFYTfQ0HOEcyUfKAa9uto="></script>
\n      <script nonce="xd914cYFGKX5DNQBrX721MCFYTfQ0HOEcyUfKAa9uto=">\n
window.dataLayer = window.dataLayer || [];\n        function gtag(){dataLa
yer.push(arguments);}\n        gtag(\'js\', new Date());\n        gtag(\'c
onfig\', \'UA-39297847-9\');\n</script>\n    <meta charset="utf-8">\n    <
meta http-equiv="X-UA-Compatible" content="IE=edge">\n    <title>DataCamp
Help Center</title>\n    <meta name="description" content="">\n    <meta n
ame="viewport" content="width=device-width, initial-scale=1">\n\n      <li
nk rel="alternate" href="http://instructor-support.datacamp.com/en/" hrefl
ang="en">\n\n      <meta name="intercom:trackingEvent" content="{&quot;nam
e&quot;:&quot;Viewed Help Center&quot;,&quot;metadata&quot;:{&quot;action&
quot;:&quot;viewed&quot;,&quot;object&quot;:&quot;educate_home&quot;,&quo
t;place&quot;:&quot;help_center&quot;,&quot;owner&quot;:&quot;educate&quo
t;,&quot;default_locale&quot;:&quot;en&quot;,&quot;current_locale&quot;:&q
uot;en&quot;,&quot;is_default_locale&quot;:true}}" />\n\n    <link rel="st
ylesheet" media="all" href="https://static.intercomassets.com/alexandria/a
```

Performing HTTP requests in Python using requests Now that you've got your head and hands around making `HTTP` requests using the `urllib` package, you're going to figure out how to do the same using the higher-level requests library. You'll once again be pinging DataCamp servers for their `"http://www.datacamp.com/teach/documentation"` page.

Note that unlike in the previous exercises using urllib, you don't have to close the connection when using requests!

Instructions

- Import the package `requests`.
- Assign the URL of interest to the variable `url`.
- Package the request to the URL, send the request and catch the response with a single function `requests.get()`, assigning the response to the variable `r`.

- Use the `text` attribute of the object r to return the HTML of the webpage as a string; store the result in a variable `text`.
- Hit submit to print the HTML of the webpage.

In [6]:

```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 response: r
r = requests.get(url)

# Extract the response: text
text = r.text

# Print the html
print(text)
```

```
<!doctype html>
<html lang="en" data-direction="ltr">
  <head>
    <link href="https://fonts.intercomcdn.com" rel="preconnect" crossorigi
n>
      <script src="https://www.googletagmanager.com/gtag/js?id=UA-39297847
-9" async="async" nonce="+WofeGB/3aofQfy5Yx3EBfIeUwTJKnwZ7AJcV3A85Ys="></s
cript>
      <script nonce="+WofeGB/3aofQfy5Yx3EBfIeUwTJKnwZ7AJcV3A85Ys=">
        window.dataLayer = window.dataLayer || [];
        function gtag(){dataLayer.push(arguments);}
        gtag('js', new Date());
        gtag('config', 'UA-39297847-9');
</script>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>DataCamp Help Center</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

3. Scraping the web in Python

3.1 Parsing HTML with BeautifulSoup

In this interactive exercise, you'll learn how to use the BeautifulSoup package to parse, prettify and extract information from HTML. You'll scrape the data from the webpage of Guido van Rossum, Python's very own Benevolent Dictator for Life (https://en.wikipedia.org/wiki/Benevolent_dictator_for_life). In the following exercises, you'll prettify the HTML and then extract the text and the hyperlinks.

The URL of interest is `url = 'https://www.python.org/~guido/'`.

Instructions

- Import the function `BeautifulSoup` from the package `bs4`.

- Assign the URL of interest to the variable `url` .
- Package the request to the URL, send the request and catch the response with a single function `requests.get()` , assigning the response to the variable `r` .
- Use the text attribute of the object `r` to return the HTML of the webpage as a string; store the result in a variable `html_doc` .
- Create a BeautifulSoup object `soup` from the resulting HTML using the function `BeautifulSoup()` .
- Use the method `prettify()` on `soup` and assign the result to `pretty_soup` .
- Hit submit to print to prettified HTML to your shell!

In [7]:

```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)
```

```
    <p>
    </p>
   </li>
  </ul>
  <h3>
   The Audio File Formats FAQ
  </h3>
  <p>
   I was the original creator and maintainer of the Audio File Formats
FAQ.  It is now maintained by Chris Bagwell
at
   <a href="http://www.cnpbagwell.com/audio-faq">
    http://www.cnpbagwell.com/audio-faq (http://www.cnpbagwell.com/audio-f
aq)
   </a>
   .  And here is a link to
   <a href="http://sox.sourceforge.net/">
    SOX
   </a>
```

3.2Turning a webpage into data using BeautifulSoup: getting the text

As promised, in the following exercises, you'll learn the basics of extracting information from HTML soup. In this exercise, you'll figure out how to extract the text from the BDFL's webpage, along with printing the webpage's title.

Instructions

- In the sample code, the HTML response object html_doc has already been created: your first task is to Soupify it using the function `BeautifulSoup()` and to assign the resulting soup to the variable `soup` .
- Extract the title from the HTML `soup` using the attribute title and assign the result to `guido_title` .
- Print the title of Guido's webpage to the shell using the `print()` function.
- Extract the text from the HTML soup soup using the method `get_text()` and assign to `guido_text` .
- Hit submit to print the text from Guido's webpage to the shell.

In [8]:

```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
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.text

# Print Guido's text to the shell
print(guido_text)
```

```
<title>Guido's Personal Home Page</title>


Guido's Personal Home Page




Guido van Rossum - Personal Home Page


"Gawky and proud of it."
Who
I Am
Read
my "King's
Day Speech" for some inspiration.

I am the author of the Python
programming language.  See also my resume
and my publications list, a brief bio, assorted writings, presentations an
d interviews (all about Python), some
pictures of me,
my new blog, and
my old
blog on Artima.com.  I am
@gvanrossum on Twitter.

In January 2013 I joined
Dropbox.  I work on various Dropbox
```

products and have 50% for my Python work, no strings attached.
Previously, I have worked for Google, Elemental Security, Zope
Corporation, BeOpen.com, CNRI, CWI, and SARA. (See
my resume.) I created Python while at CWI.

How to Reach Me

You can send email for me to guido (at) python.org.
I read everything sent there, but if you ask
me a question about using Python, it's likely that I won't have time
to answer it, and will instead refer you to
help (at) python.org,
comp.lang.python or
StackOverflow. If you need to
talk to me on the phone or send me something by snail mail, send me an
email and I'll gladly email you instructions on how to reach me.

My Name

My name often poses difficulties for Americans.

Pronunciation: in Dutch, the "G" in Guido is a hard G,
pronounced roughly like the "ch" in Scottish "loch". (Listen to the
sound clip.) However, if you're
American, you may also pronounce it as the Italian "Guido". I'm not
too worried about the associations with mob assassins that some people
have. :-)

Spelling: my last name is two words, and I'd like to keep it
that way, the spelling on some of my credit cards notwithstanding.
Dutch spelling rules dictate that when used in combination with my
first name, "van" is not capitalized: "Guido van Rossum". But when my
last name is used alone to refer to me, it is capitalized, for
example: "As usual, Van Rossum was right."

Alphabetization: in America, I show up in the alphabet under
"V". But in Europe, I show up under "R". And some of my friends put
me under "G" in their address book...

More Hyperlinks

Here's a collection of essays relating to Python
that I've written, including the foreword I wrote for Mark Lutz' book
"Programming Python".
I own the official
Python license.

The Audio File Formats FAQ

I was the original creator and maintainer of the Audio File Formats
FAQ. It is now maintained by Chris Bagwell
at http://www.cnpbagwell.com/audio-faq. (http://www.cnpbagwell.com/audio-faq.) And here is a link to
SOX, to which I contributed
some early code.

"On the Internet, nobody knows you're
a dog."

3.3Turning a webpage into data using BeautifulSoup: getting the hyperlinks

In this exercise, you'll figure out how to extract the URLs of the hyperlinks from the BDFL's webpage. In the process, you'll become close friends with the soup method `find_all()` .

Instructions

- Use the method `find_all()` to find all hyperlinks in soup, remembering that hyperlinks are defined by the HTML tag `<a\>` but passed to `find_all()` without angle brackets; store the result in the variable `a_tags` .
- The variable a_tags is a results set: your job now is to enumerate over it, using a for loop and to print the actual URLs of the hyperlinks; to do this, for every element link in `a_tags` , you want to print() `link.get('href')` .

In [9]:

```python
# Import packages
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'))
```

```
<title>Guido's Personal Home Page</title>
pics.html
pics.html
http://www.washingtonpost.com/wp-srv/business/longterm/microsoft/stories/1
998/raymond120398.htm (http://www.washingtonpost.com/wp-srv/business/longt
erm/microsoft/stories/1998/raymond120398.htm)
http://metalab.unc.edu/Dave/Dr-Fun/df200004/df20000406.jpg (http://metala
b.unc.edu/Dave/Dr-Fun/df200004/df20000406.jpg)
http://neopythonic.blogspot.com/2016/04/kings-day-speech.html (http://neop
ythonic.blogspot.com/2016/04/kings-day-speech.html)
http://www.python.org (http://www.python.org)
Resume.html
Publications.html
bio.html
http://legacy.python.org/doc/essays/ (http://legacy.python.org/doc/essay
s/)
http://legacy.python.org/doc/essays/ppt/ (http://legacy.python.org/doc/ess
ays/ppt/)
interviews.html
pics.html
http://neopythonic.blogspot.com (http://neopythonic.blogspot.com)
http://www.artima.com/weblogs/index.jsp?blogger=12088 (http://www.artima.c
om/weblogs/index.jsp?blogger=12088)
https://twitter.com/gvanrossum (https://twitter.com/gvanrossum)
http://www.dropbox.com (http://www.dropbox.com)
Resume.html
http://groups.google.com/groups?q=comp.lang.python (http://groups.google.c
om/groups?q=comp.lang.python)
http://stackoverflow.com (http://stackoverflow.com)
guido.au
http://legacy.python.org/doc/essays/ (http://legacy.python.org/doc/essay
s/)
```

images/license.jpg
http://www.cnpbagwell.com/audio-faq (http://www.cnpbagwell.com/audio-faq)
http://sox.sourceforge.net/ (http://sox.sourceforge.net/)
images/internetdog.gif