10 - Text Acquisition

Data Science Practicum 2021/22, Lesson 10

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Reading files

```
file= open(“filename”, “mode”)
file.read()
file.close()
```

- modes:
  - ‘r’ – read mode
  - ‘w’ – write mode
  - ‘a’ – append mode
  - ‘r+’ – special read and write mode

Reading files

```
file= open("filename", "mode")
file.read()
file.close()
```

- modes:
  - 'r' – read mode
  - 'w' – write mode
  - 'a' – append mode
  - 'r+' – special read and write mode

```
with open("filename") as file:
    data = file.read()
```

Writing to FIles

```
file =open("myfile.txt","w")
file.write("Hello World!")
file.close()
```

Exercise 1

- Download the HTML page http://markotkalcic.com/ using the browser (HTML only)
- Write a function that will read that HTML page and then write the first 75 characters to another file
- Hint: encoding

Exercise 1

- Download the HTML page http://markotkalcic.com/ using the browser (HTML only)
- Write a function that will read that HTML page and then write the first 75 characters to another file
- Hint: encoding

```
def read_write_75char(fname):
    file1 = open(fname, "r", encoding="utf-8")
    content = file1.read()
    file1.close

    file2 = open("out.txt","w")
    file2.write(content[:75])
    file2.close

read_write_75char("tkalcic_home.html")
```

Exercise 2

- Write a function that will count the number of links <a...>

Exercise 2

- Write a function that will count the number of links <a...>

```
def count_links(fname):
    file1 = open(fname, "r", encoding="utf-8")
    content = file1.read()
    file1.close
    nLinks = 0
    iLink = content.find("<a")
    while (iLink != -1):
        iLink = content.find("<a", iLink+1)
        nLinks += 1
    print(nLinks)

count_links("tkalcic_home.html")
```

Exercise 3

- Write a function that will check if each <dt> is closed before a new one is opened

Exercise 3

- Write a function that will check if each <dt> is closed before a new one is opened

```
def check_dt(fname):
    file1 = open(fname, "r", encoding="utf-8")
    content = file1.read()
    file1.close

    iDT = content.find("<dt>")
    while (iDT != -1):
        #print("iDT = ", iDT)
        iEDT = content.find("</dt>", iDT+1)
        #print("iEDT = ", iEDT)
        iDT_next = content.find("<dt>", iDT+1)
        #print("iNext = ", iDT_next)
        if (iDT_next<iEDT):
            print("error at position ",iEDT)
        iDT = iDT_next

check_dt("tkalcic_home.html")
```

Table of Contents

Requests

- The requests module allows you to send HTTP requests using Python.

```
import requests

x = requests.get('https://w3schools.com/python/demopage.htm')

print(x.text)
```

Requests

- The requests module allows you to send HTTP requests using Python.

```
import requests

x = requests.get('https://w3schools.com/python/demopage.htm')

print(x.text)
```

```
requests.methodname(params)
```

Method	Description
delete(url, args)	Sends a DELETE request to the specified url
get(url, params, args)	Sends a GET request to the specified url
head(url, args)	Sends a HEAD request to the specified url
patch(url, data, args)	Sends a PATCH request to the specified url
post(url, data, json, args)	Sends a POST request to the specified url
put(url, data, args)	Sends a PUT request to the specified url
request(method, url, args)	Sends a request of the specified method to the specified url

Requests

- Keep-Alive & Connection Pooling
- Browser-style SSL Verification
- Automatic Decompression
- Multipart File Uploads
- HTTP(S) Proxy Support
- Streaming Downloads

Requests

- Keep-Alive & Connection Pooling
- Browser-style SSL Verification
- Automatic Decompression
- Multipart File Uploads
- HTTP(S) Proxy Support
- Streaming Downloads

```
import requests
response = requests.get("https://api.github.com/user")
print(response.status_code)
print(response.headers["content-type"])
print(response.encoding)
print(response.text)
```

```
401
application/json; charset=utf-8
utf-8
{"message":"Requires authentication","documentation_url":"https://docs.github.com/rest/reference/users#get-the-authenticated-user"
```

Exercise

- Revise the exercise for counting links so that it opens a remote URL using requests

Exercise

- Revise the exercise for counting links so that it opens a remote URL using requests

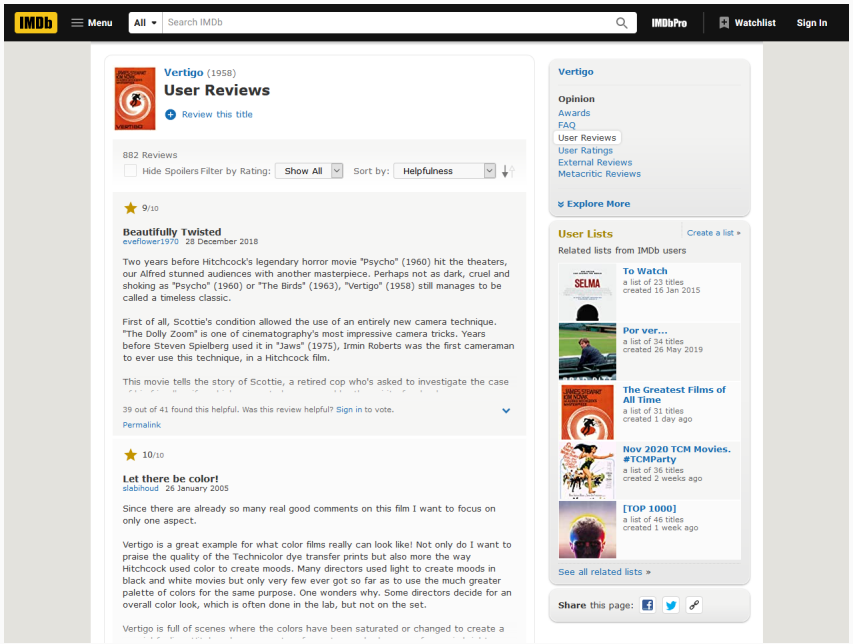
```
import requests

def count_links(url):
    response = requests.get(url)
    content = response.text
    nLinks = 0
    iLink = content.find("<a")
    while (iLink != -1):
        iLink = content.find("<a", iLink+1)
        nLinks += 1
    print(nLinks)

count_links("http://markotkalcic.com")
```

BeautifulSoup

- Parsing HTML/XML

```
import requests

url = "http://bit.do/imdb_rev_page"
response = requests.get(url)
print(response.text[:130])
```

```
<!DOCTYPE html>
<html
    xmlns:og="http://ogp.me/ns#"
    xmlns:fb="http://www.facebook.com/2008/fbml">
    <head>
```



BeautifulSoup

```
from bs4 import BeautifulSoup
import requests
url = "http://bit.do/imdb_rev_page"
req = requests.get(url)
soup = BeautifulSoup(req.text, "html.parser")
print(soup.title)
```

```
<title>Vrtoglavica (1958) - Vrtoglavica (1958) - User Reviews - IMDb</title>
```

Find links

```
for link in html_soup.find_all('a'):
    print(link.get('href'))
```

```
/?ref_=nv_home
https://www.imdb.com/calendar/?ref_=nv_mv_cal
https://www.imdb.com/list/ls016522954/?ref_=nv_tvv_dvd
/chart/top/?ref_=nv_mv_250
/chart/moviemeter/?ref_=nv_mv_mpm
https://www.imdb.com/feature/genre/?ref_=nv_ch_gr
/chart/boxoffice/?ref_=nv_ch_cht
...
```

Opening local file or text

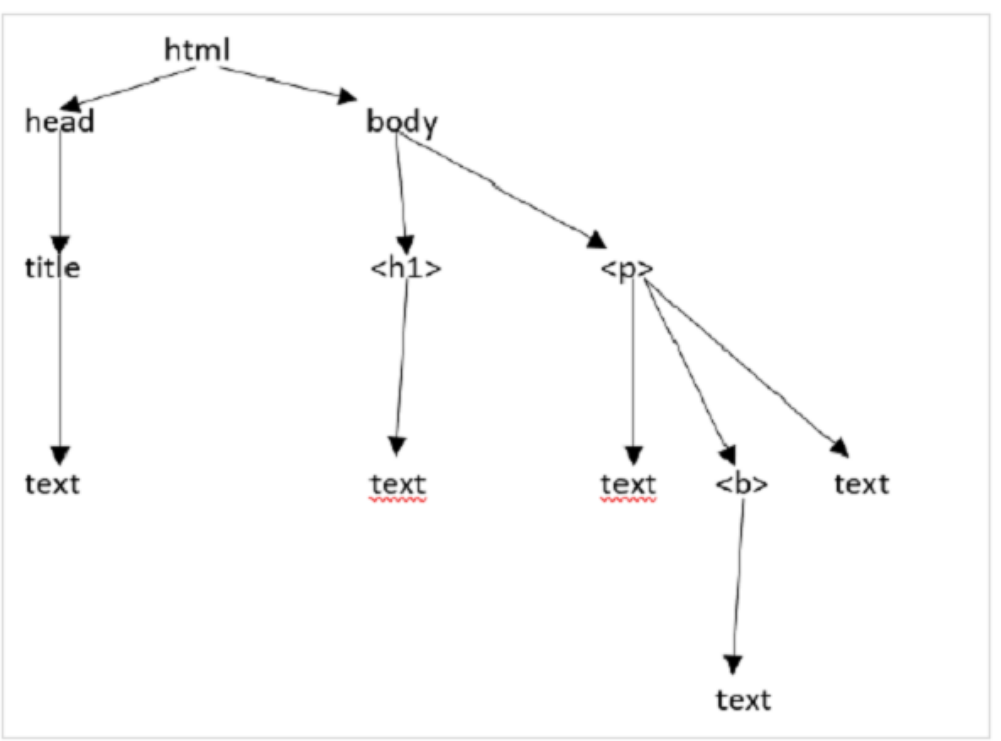
```
from bs4 import BeautifulSoup

with open("tkalcic_home.html", encoding="utf-8") as fp:
    html_soup = BeautifulSoup(fp)

...

html_soup = BeautifulSoup("<html>data</html>")
```

Opening local file or text

```
from bs4 import BeautifulSoup

with open("tkalcic_home.html", encoding="utf-8") as fp:
    html_soup = BeautifulSoup(fp)

...

html_soup = BeautifulSoup("<html>data</html>")
```

- BeautifulSoup uses different parsers
  - lxml
  - html
  - xml
- They work in slightly different ways https://www.crummy.com/software/BeautifulSoup/bs4/doc/#specifying-the-parser-to-use

Document Structure



- BS converts the HTML/XML string to many objects, e.g.:
  - Tag
  - NavigableString
  - BeautifulSoup
  - Comments

Tag

```
from bs4 import BeautifulSoup
soup = BeautifulSoup('<b class="boldest">TutorialsPoint</b>')
tag = soup.html
print(tag)
print(tag.name)

tag.name="MyMarkupLanguage"
print(tag)
```

```
<html><body><b class="boldest">TutorialsPoint</b></body></html>
html
<MyMarkupLanguage><body><b class="boldest">TutorialsPoint</b></body></MyMarkupLanguage>
```

Tag

```
from bs4 import BeautifulSoup
soup = BeautifulSoup('<b class="boldest">TutorialsPoint</b>')
tag = soup.html
print(tag)
print(tag.name)

tag.name="MyMarkupLanguage"
print(tag)
```

```
<html><body><b class="boldest">TutorialsPoint</b></body></html>
html
<MyMarkupLanguage><body><b class="boldest">TutorialsPoint</b></body></MyMarkupLanguage>
```

```
tutorialsP = BeautifulSoup("<div class='tutorialsP'></div>",'lxml')
tag2 = tutorialsP.div
tag2['class']
```

NavigableString

- The navigablestring object is used to represent the contents of a tag. To access the contents, use ".string" with tag.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup("<h2 id='message'>Hello, Tutorialspoint!</h2>")

print(soup.string)
print(type(soup.string))
```

```
Hello, Tutorialspoint!
<class 'bs4.element.NavigableString'>
```

Navigating by Tags

```
html_doc = """
<html><head><title>Tutorials Point</title></head>
<body>
<p class="title"><b>The Biggest Online Tutorials Library, It's all Free</b></p>
<p class="prog">Top 5 most used Programming Languages are:
<a href="https://www.tutorialspoint.com/java/java_overview.htm" class="prog" id="link1">Java</a>,
<a href="https://www.tutorialspoint.com/cprogramming/index.htm" class="prog" id="link2">C</a>,
<a href="https://www.tutorialspoint.com/python/index.htm" class="prog" id="link3">Python</a>,
<a href="https://www.tutorialspoint.com/javascript/javascript_overview.htm" class="prog" id="link4">JavaScript</a> and
<a href="https://www.tutorialspoint.com/ruby/index.htm" class="prog" id="link5">C</a>;
as per online survey.</p>
<p class="prog">Programming Languages</p>
"""

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

- Find the first tag

```
print(soup.head)
```

```
<head><title>Tutorials Point</title></head>
```

Navigating by Tags

```
html_doc = """
<html><head><title>Tutorials Point</title></head>
<body>
<p class="title"><b>The Biggest Online Tutorials Library, It's all Free</b></p>
<p class="prog">Top 5 most used Programming Languages are:
<a href="https://www.tutorialspoint.com/java/java_overview.htm" class="prog" id="link1">Java</a>,
<a href="https://www.tutorialspoint.com/cprogramming/index.htm" class="prog" id="link2">C</a>,
<a href="https://www.tutorialspoint.com/python/index.htm" class="prog" id="link3">Python</a>,
<a href="https://www.tutorialspoint.com/javascript/javascript_overview.htm" class="prog" id="link4">JavaScript</a> and
<a href="https://www.tutorialspoint.com/ruby/index.htm" class="prog" id="link5">C</a>;
as per online survey.</p>
<p class="prog">Programming Languages</p>
"""

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

- Find the first tag

```
print(soup.head)
```

```
<head><title>Tutorials Point</title></head>
```

```
print(soup.body.b)
```

```
print(soup.body.b)

<b>The Biggest Online Tutorials Library, It's all Free</b>
```

Navigating by Tags

```
soup.a
```

```
<a class="prog" href="https://www.tutorialspoint.com/java/java_overview.htm" id="link1">Java</a>
```

Navigating by Tags

```
soup.a
```

```
<a class="prog" href="https://www.tutorialspoint.com/java/java_overview.htm" id="link1">Java</a>
```

```
res = soup.find_all("a")
print(len(res))
print(res[3])
```

```
5
<a class="prog" href="https://www.tutorialspoint.com/javascript/javascript_overview.htm" id="link4">JavaScript</a>
```

Exercise

- Pick a URL of a web page
- Write a Python program to retrieve and print all the paragraph tags from a given html document.

Exercise

- Pick a URL of a web page
- Write a Python program to retrieve and print all the paragraph tags from a given html document.

```
response = requests.get("http://markotkalcic.com")
content = response.text
soup = BeautifulSoup(content, 'html.parser')
for p in soup.find_all("p"):
    print(p)
```

```
<p><span class="menu-button" style="font-weight: 700"><a href="index.html">Home</a> </span>
<span class="menu-button"><a href="about.html">About</a> </span>
<span class="menu-button"><a href="research.html">Research</a> </span>
<span class="menu-button"><a href="publications.html">Publications</a> </span>
<span class="menu-button"><a href="teaching.html">Teaching</a> </span>
<span class="menu-button"><a href="cv.html">CV</a> </span>
<span class="menu-button"><a href="contact.html">Contact</a></span></p>
<p><img id="profile_pic" src="resources/markot3.jpg" style="float:left;width:20%;"/></p>
<p>I am associate professor at the Faculty of Mathematics,
    Natural Sciences and Information Technologies (FAMNIT) at the University of Primorska in Koper, Slovenia. I aim at
    improving personalized services (e.g. recommender systems) through the usage of psychological models in personalization
    algorithms. To achieve this, I use diverse research methodologies, including data mining, machine learning, and user studies.<
<p>If you are further interested in my work, please download my <a href="resources/Tkalcic_CV_Public.pdf">CV</a>.</p>
<p> </p>
<p> </p>
<p><span class="footer"><em>Last updated on Nov 12, 2020. Created using <a href="https://github.com/eakbas/TSPW">TSPW
</a> and <a href="http://pandoc.org/">pandoc</a>.</em></span></p>
```

Exercise

- Write a Python program to find the length of the text of the first <h2> tag of a given html document.

Exercise

- Write a Python program to find the length of the text of the first <h2> tag of a given html document.

```
response = requests.get("http://markotkalcic.com")
content = response.text
soup = BeautifulSoup(content, 'html.parser')
print("Length of the text of the first <h2> tag:")
print(len(soup.find('h2').text))
print(soup.find('h2').text)
```

```
Length of the text of the first <h2> tag:
4
Home
```

Exercise

- Write a Python program to find the href of the fifth <a> tag of a given html document.

Exercise

- Write a Python program to find the href of the fifth <a> tag of a given html document.

```
response = requests.get("http://markotkalcic.com")
content = response.text
soup = BeautifulSoup(content, 'html.parser')
res = soup.find_all("a");
print(res[4]["href"])
print(res[4].attrs["href"])
```

```
teaching.html
teaching.html
```

Exercise

- Write a Python program to extract all the URLs from the webpage `https://python.org` that are nested within `<li>` tags.

Exercise

- Write a Python program to extract all the URLs from the webpage `https://python.org` that are nested within `<li>` tags.

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.python.org/'
reqs = requests.get(url)
soup = BeautifulSoup(reqs.text, 'html.parser')

urls = []
for h in soup.find_all('li'):
    a = h.find('a')
    urls.append(a.attrs['href'])
print(urls)
```

```
['/', '/psf-landing/', 'https://docs.python.org', 'https://pypi.org/', '/jobs/', '/community-landing/', '#',
'javascript:;', 'javascript:;', 'javascript:;', '#', 'https://www.facebook.com/pythonlang?fref=ts',
'https://twitter.com/ThePSF', '/community/irc/', '/about/', '/about/apps/', '/about/quotes/', '/about/gettingstarted/',
'/about/help/', 'http://brochure.getpython.info/', '/downloads/', '/downloads/', '/downloads/source/',
'/downloads/windows/', '/downloads/mac-osx/', '/download/other/', 'https://docs.python.org/3/license.html',
'/download/alternatives', '/doc/', '/doc/', '/doc/av', 'https://wiki.python.org/moin/BeginnersGuide', 'https://devguide.python.org
```

Exercise

- Write a program that
  - downloads the movie reviews at `http://bit.do/imdb_rev_page`
  - extract the text of each of the reviews
  - for each review, count the number of occurrences of the following words:
    - good, like, not
  - hint:
    `https://www.crummy.com/software/BeautifulSoup/bs4/doc/#searching-by-css-class`

Exercise

- Write a program that
  - downloads the movie reviews at http://bit.do/imdb_rev_page
  - extract the text of each of the reviews
  - for each review, count the number of occurrences of the following words:
    - good, like, not
  - hint:
    https://www.crummy.com/software/BeautifulSoup/bs4/doc/#searching-by-css-class

```
import requests
from bs4 import BeautifulSoup

words = ["good", "like", "not"]
url = "http://bit.do/imdb_rev_page"
response = requests.get(url)
html_soup = BeautifulSoup(response.text,"html.parser")

movie_containers = html_soup.find_all("div", class_="text show-more__control")
iReview = 0
for review in movie_containers:
    print("---\nReview ",iReview)
    for w in words:
        print(w, " ", review.text.count(w))
    iReview += 1
```

```
---
Review  0
good   0
like   1
not   4
---
...
```

Table of Contents

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://www.w3resource.com/python-exercises/BeautifulSoup/index.php
- https://www.tutorialspoint.com/beautiful