Table of contents

---

1. webbrowser

(go to top)

In [1]:
```
import webbrowser
```

In [5]:
```
webbrowser.open('https://facebook.com/')
```

Out[5]:
```
True
```

---

2. Request Module

- The requests.get() function takes a string of a URL to download. By call- ing type() on requests.get()'s return value, you can see that it returns a Response object, which contains the response that the web server gave for your request. I'll explain the Response object in more detail later, but for now, enter the following into the interactive shell while your computer is connected to the internet:

(go to top)

install using `pip install --user requests`

In [13]:
```
import requests # check documentation later
```

In [14]:
```
res = requests.get('https://automatetheboringstuff.com/files/rj.txt')
```

In [19]:
```
len(res.text)
```

Out[19]:
```
468216
```

In [15]:
```
print(res.text[0:250])
```

```
The Project Gutenberg EBook of Romeo and Juliet, by William Shakespea
re

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Projec
```

In [4]:
```
type(res)
```

Out[4]:
```
requests.models.Response
```

- You can tell that the request for this web page succeeded by checking the status_code attribute of the Response object. If it is equal to the value of requests.codes.ok, then everything went fine
- the status code in HTTP protocol
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

In [5]:
```
res.status_code
```

Out[5]:
```
200
```

In [6]:
```
requests.codes.ok
```

Out[6]:
```
200
```

- Check for errors with `raise_for_status()`

In [26]:
```
res = requests.get('https://inventwithpython.com/page_that_does_not_ex
res.raise_for_status()
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last
)
<ipython-input-26-e43daf6a44f2> in <module>
      1 res = requests.get('https://inventwithpython.com/page_that_do
es_not_exist')
----> 2 res.raise_for_status()

~/opt/anaconda3/lib/python3.8/site-packages/requests/models.py in rai
se_for_status(self)
    939
    940         if http_error_msg:
--> 941             raise HTTPError(http_error_msg, response=self)
    942
    943     def close(self):

HTTPError: 404 Client Error: Not Found for url: https://inventwithpyt
hon.com/page_that_does_not_exist
(https://inventwithpython.com/page_that_does_not_exist)
```

- The `raise_for_status()` method is a good way to ensure that a program halts if a bad download occurs. This is a good thing: You want your program to stop as soon as some unexpected error happens. If a failed download isn't a deal breaker for your program, you can wrap the `raise_for_status() line with try and except statements` to handle this error case without crashing.

In [29]:
```
res = requests.get('https://inventwithpython.com/page_that_does_not_ex
try:
    res.raise_for_status()
except Exception as exc:
    print('There was a problem: %s' % (exc))
```

```
There was a problem: 404 Client Error: Not Found for url: https://inv
entwithpython.com/page_that_does_not_exist
(https://inventwithpython.com/page_that_does_not_exist)
```

- `Always call raise_for_status() after calling requests.get()` . You want to be sure that the download has actually worked before your program continues.

```
------------------------------------------------------------------------
----------------------------------------
```

In [30]:
```
res = requests.get('https://www.youtube.com/watch?v=U75SUcvHBvc&t=206s
res.raise_for_status()
```

3. iter_content()

(go to top)

In [ ]:
```
res = requests.get('https://automatetheboringstuff.com/files/rj.txt')
res.raise_for_status()
```

In [41]:
```
for chunk in res.iter_content(10):
    print('1', chunk)
```

```
1 b'<!DOCTYPE '
1 b'html><html'
1 b' style="fo'
1 b'nt-size: 1'
1 b'0px;font-f'
1 b'amily: Rob'
1 b'oto, Arial'
1 b', sans-ser'
1 b'if;" lang='
1 b'"en" typog'
1 b'raphy typo'
1 b'graphy-spa'
1 b'cing><head'
1 b'><meta htt'
1 b'p-equiv="X'
1 b'-UA-Compat'
1 b'ible" cont'
1 b'ent="IE=ed'
1 b'ge"/><scri'
```

In [ ]:
```

```

4. Parse HTML with B-SOUP

(go to top)

- `Beautiful Soup` is a module for extracting information from an HTML page (and is much better for this purpose than regular expressions). The Beautiful Soup module's name is `bs4 (for Beautiful Soup, version 4).`
- To install it, you will need to run `pip install --user beautifulsoup4` from the command line.

In [42]:
```
import requests, bs4
```

In [44]:
```
res = requests.get('https://nostarch.com')
res.raise_for_status()
```

In [47]:
```
noStarchSoup = bs4.BeautifulSoup(res.text, 'lxml')
type(noStarchSoup)
```

Out[47]:
```
bs4.BeautifulSoup
```

In [48]:
```
noStarchSoup
```

```
<h2 class="block-title"><a href="/cart"><span class="cart-block-ico
n-empty" title="View your shopping cart."></span></a><span class="c
art-block-title-bar" title="Show/hide shopping cart contents.">Shop
ping cart<span class="cart-block-arrow arrow-down"></span></span></
h2>
<p class="cart-block-items collapsed uc-cart-empty">There are no pr
oducts in your shopping cart.</p><table class="cart-block-summary">
<tbody><tr><td class="cart-block-summary-items"><span class="num-it
ems">0</span> Items</td><td class="cart-block-summary-total"><label
>Total:</label> <span class="uc-price">$0.00</span></td></tr></tbod
y></table>
</section>
<section class="block block-nostarch-customclearfix" id="block-nost
arch-custom-login-block">
<h2 class="block-title">User login</h2>
<ul>
<li><a href="/user">Log in</a></li>
<li><a href="/user/register">Create account</a></li>
</ul>
</section>
```

In [ ]:
```
```

5. BS4 - Find an element with `select()`

- You can `retrieve a web page element from a BeautifulSoup object by calling the select()method` and passing a `string of a CSS selector` for the element you are looking for. Selectors are like regular expressions: they specify a pattern to look for—in this case, in HTML pages instead of general text strings.

(go to top)

```
noStarchSoup.select?
```

In [123]:
```
noStarchSoup = bs4.BeautifulSoup(res.text, 'lxml')
```

In [50]:
```
noStarchSoup.select('div')
```

```
-toggle collapsed" data-target="#navbar" data-toggle="collapse" typ
e="button">
 <span class="sr-only">Toggle navigation</span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
 </button>
 <a class="navbar-brand text-uppercase" href="/"><img alt="Home" cl
ass="img-responsive" src="https://nostarch.com/sites/all/themes/nos
tarch/logo.png"/></a>
 </div>
 <div class="navbar-collapse collapse text-center" id="navbar">
 <ul class="menu nav navbar-nav"><li class="first leaf"><a href="/c
atalog.htm" title="Explore our catalog">Catalog</a></li>
 <li class="leaf"><a href="https://nostarch.com/no-starch-death-met
al-t-shirt-new" title="Merchandise">Merch</a></li>
 <li class="leaf"><a href="/blog" title="The No Starch Press blog">
Blog</a></li>
 <li class="leaf"><a href="/media.htm" title="Media contact">Media<
/a></li>

 ----------------------------------------------------------------------
----------------------------------------


 ----------------------------------------------------------------------
----------------------------------------
```

- The element with an id attribute of author
  `noStarchSoup.select('#author')`
- All elements that use a CSS class attribute named notice
  `noStarchSoup.select('.notice')`
- All elements named `<span>` that are within an element named `<div>`
  `noStarchSoup.select('div span')`
- All elements named `<span>` that are directly within an element named `<div>` , with no other element in between
  `noStarchSoup.select('div > span')`
- All elements named `<input>` that have a name attribute with any value
  `noStarchSoup.select('input[name]')`
- All elements named `<input>` that have an attribute named type with value button
  `noStarchSoup.select('input[type = "button"]')`

In [100]:
```
noStarchSoup.select('p a')[:15] #a tags inside p tags
```

Out[100]:
```
[<a href="https://www.humblebundle.com/books/learn-you-more-python-books" target="_blank"><img class="banner_image" src="/images/banners/humble_Learn-You-More--Python_desktop_v01.png"/></a>,
 <a href="/hello-web-design">Hello Web Design!</a>,
 <a href="/art-webassembly">The Art of Web Assembly</a>,
 <a href="/arduino-workshop-2nd-edition">Arduino Workshop, 2nd Edition</a>,
 <a href="/computer-graphics-scratch">Computer Graphics from Scratch</a>,
 <a href="/carbon-one-atoms-odyssey">Carbon: One Atom's Odyssey</a>,
 <a href="/how-hack-ghost"> How to Hack Like a Ghost</a>,
 <a href="/Cyberjutsu">Cyberjutsu</a>,
 <a href="/Learn-Python-Visually">Learn Python Visually</a>,
 <a href="howlinuxworks3">How Linux Works</a>,
 <a href="/black-hat-python2E">Black Hat Python, 2nd Edition</a>,
 <a href="/practical-iot-hacking">Practical IoT Hacking</a>,
 <a href="/kill-it-fire">Kill It with Fire</a>,
 <a href="/networkprogrammingwithgo">Network Programming with Go</a>,
 <a href="/practical-deep-learning-python">Practical Deep Learning</a>]
```

In [104]:
```
type(noStarchSoup.select('p a')[:15]) #a tags inside p tags
```

Out[104]:
```
list
```

```
_____________________________________________________________________
_____________________________________________
```

```
_____________________________________________________________________
_____________________________________________
```

In [72]:
```
inputGroups = noStarchSoup.select('p a')
```

In [101]:
```
type(inputGroups)
```

Out[101]:
```
bs4.element.ResultSet
```

In [74]:
```
type(inputGroups[0])
```

Out[74]:
```
bs4.element.Tag
```

In [75]:
```
len(inputGroups)
```

Out[75]:
```
308
```

In [98]:
```
inputGroups[0]
```

Out[98]:
```
<a href="https://www.humblebundle.com/books/learn-you-more-python-boo
ks" target="_blank"><img class="banner_image" src="/images/banners/hu
mble_Learn-You-More--Python_desktop_v01.png"/></a>
```

In [99]:
```
inputGroups[0].attrs
```

Out[99]:
```
{'href': 'https://www.humblebundle.com/books/learn-you-more-python-bo
oks',
 'target': '_blank'}
```

In [105]:
```
str(inputGroups[2])
```

Out[105]:
```
'<a href="/art-webassembly">The Art of Web Assembly</a>'
```

In [78]:
```
inputGroups[2].getText()
```

Out[78]:
```
'The Art of Web Assembly'
```

In [88]:
```
for i,j in enumerate(inputGroups[:10]):
    print('inputGroups[' + str(i) +'].getText(): ', j.getText())
```

```
inputGroups[0].getText():
inputGroups[1].getText():  Hello Web Design!
inputGroups[2].getText():  The Art of Web Assembly
inputGroups[3].getText():  Arduino Workshop, 2nd Edition
inputGroups[4].getText():  Computer Graphics from Scratch
inputGroups[5].getText():  Carbon: One Atom's Odyssey
inputGroups[6].getText():   How to Hack Like a Ghost
inputGroups[7].getText():  Cyberjutsu
inputGroups[8].getText():  Learn Python Visually
inputGroups[9].getText():  How Linux Works
```

In [110]:
```
pElems = noStarchSoup.select('p')
```

In [120]:
```
for i,j in enumerate(pElems[:10]):
    print('\n' + 'pElems[' + str(i) +'].getText(): ', j.getText())
```

```
pElems[0].getText():

pElems[1].getText():  There are no products in your shopping cart.

pElems[2].getText():  Don't wish for your own website, make it yourse
lf with Hello Web Design! Written for beginners, brimming with profes
sional insights, and sure to inspire.

pElems[3].getText():  The Art of Web Assembly is a thorough and pract
ice-based introduction to the new web standard dramatically speeding
up web performance.

pElems[4].getText():  Arduino Workshop, 2nd Edition covers the latest
version of the homemade-electronics platform's open-source IDE, and u
pdates dozens of projects with new hardware and cool features.

pElems[5].getText():  Computer Graphics from Scratch demystifies the
algorithms used in modern graphics software and guides beginners thro
ugh building photorealistic 3D renders.

pElems[6].getText():  Carbon: One Atom's Odyssey is an exquisitely il
lustrated, beautifully adapted story that traces a key element of pla
netary life over billions of years.

pElems[7].getText():   How to Hack Like a Ghost is a fast-paced adven
ture that lets you shadow a master hacker targeting a shady foe with
advanced cloud security.

pElems[8].getText():  Based on techniques adapted from authentic Japa
nese ninja scrolls, Cyberjutsu teaches ancient approaches to modern s
ecurity problems.

pElems[9].getText():  Learn Python Visually takes a visual approach t
o teaching total beginners key programming concepts and coding techni
ques used in creative technology
```

6.Just read the BS4 DOC

(go to top)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

7. Title

(go to top)

8. Title

(go to top)

9. Title

(go to top)

10. Title

(go to top)

11. Title

(go to top)

12. Title

(go to top)

13. Title

(go to top)

14. Title

(go to top)

15. Title

(go to top)

16. Title

(go to top)

17. Title

(go to top)

18. Title

(go to top)

19. Title

(go to top)

20. Title

(go to top)

In [ ]:

In [ ]: