WEBSCRAPING

Tim De Deken & Arno Goyvaerts

K.H.Kempen en Lessius bundelen de krachten en worden *more*.

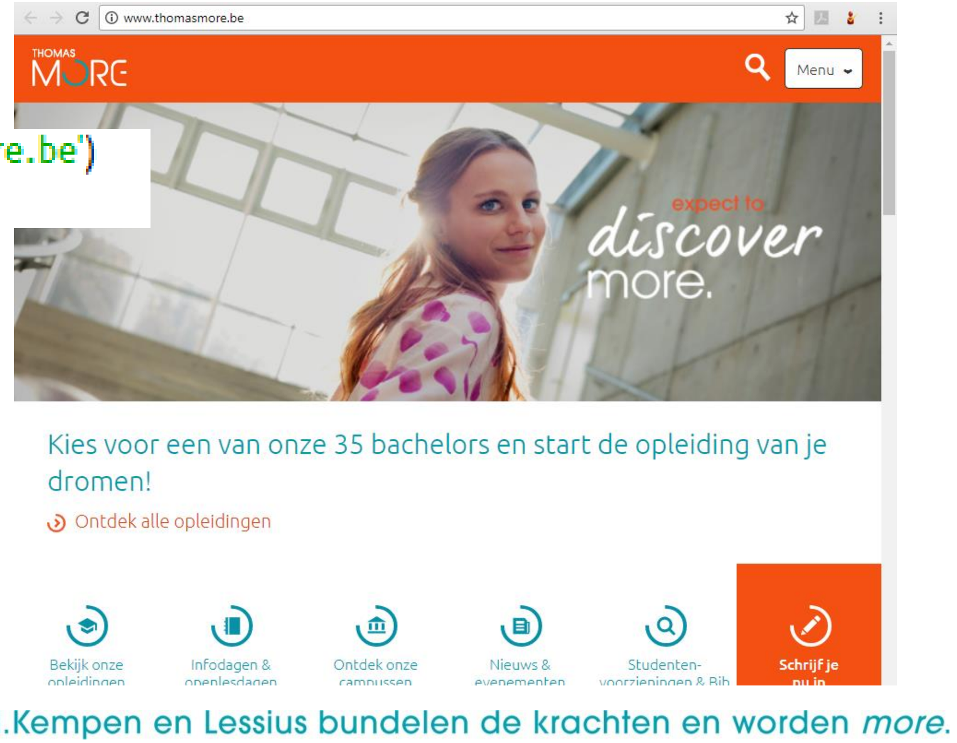
MODULES

- Webbrowser
- Requests
  - pip install requests
- BeautifulSoup4
  - pip install beautifulsoup4
- Selenium
  - pip install selenium

WEBBROWSER

- Moet worden geïmporteerd
  - import webbrowser
- Opent webpagina’s in nieuw tabblad
  - webbrowser.open(‘URL’)

```
>>> webbrowser.open('http://www.thomasmore.be')
True
```



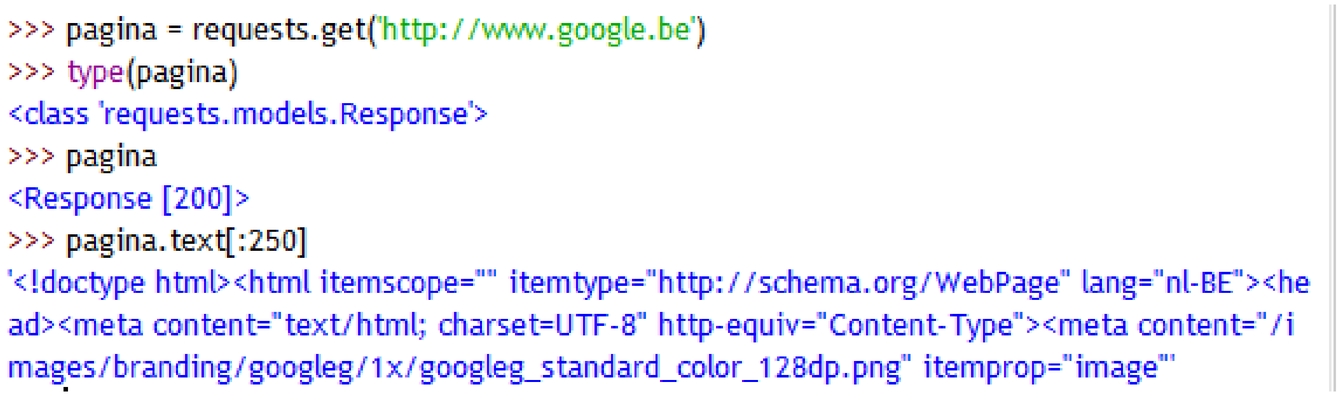
Kies voor een van onze 35 bachelors en start de opleiding van je dromen!

Ontdek alle opleidingen

K.H.Kempen en Lessius bundelen de krachten en worden *more*.

REQUESTS

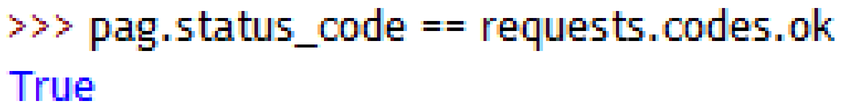
- Moet worden geïmporteerd
  - import requests
- Downloaden van webpagina’s
  - requests.get(‘URL’)
  - Geeft response object terug
  - Object.text geeft inhoud van pagina weer ([:250] geeft enkel eerste 250 tekens)



```
>>> pagina = requests.get('http://www.google.be')
>>> type(pagina)
<class 'requests.models.Response'>
>>> pagina
<Response [200]>
>>> pagina.text[:250]
'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="nl-BE"><he
ad><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/i
mages/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"'
```

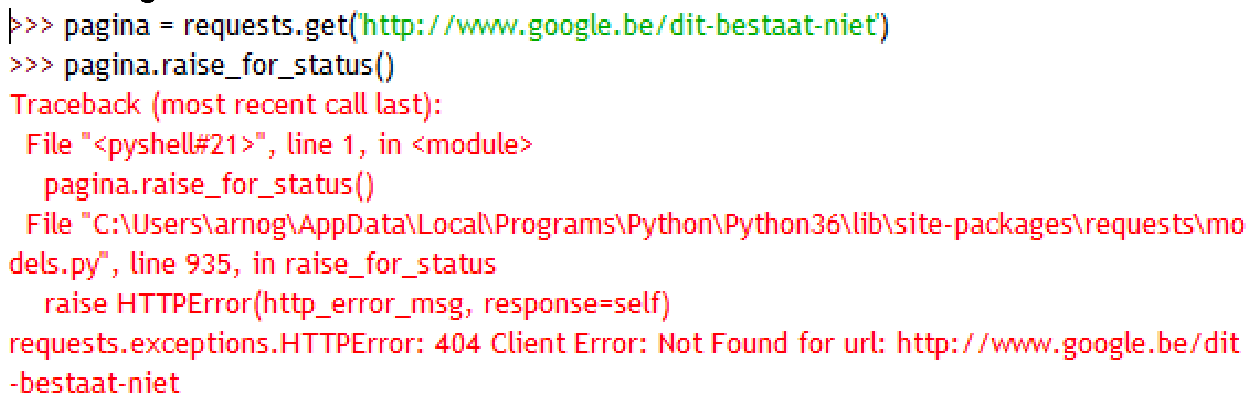
REQUESTS

- Controle op fouten
  - requests.codes.ok
    - Controle naar response code van pagina (200 = ok, 404 = not found,…)



```
>>> pag.status_code == requests.codes.ok
True
```

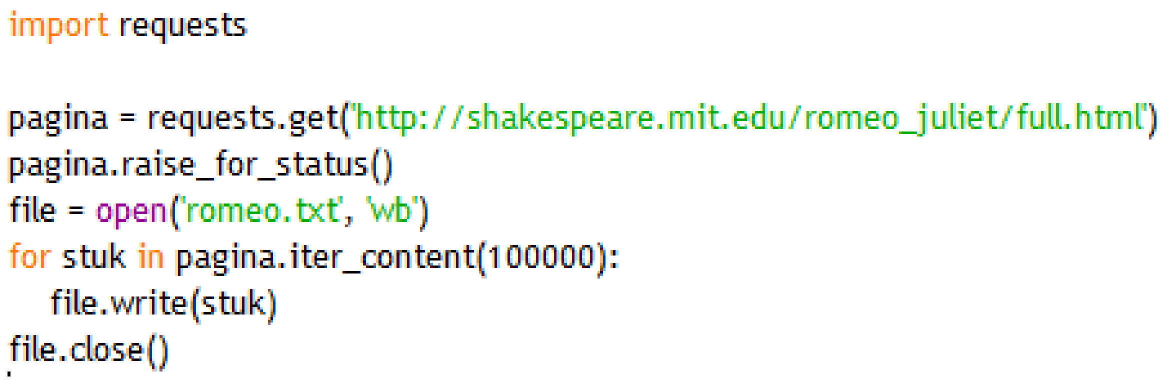
  - .raise_for_status()
    - Laat programma stoppen indien er iets fout ging bij downloaden
    - Try en Except als download geen noodzaak is



```
>>> pagina = requests.get('http://www.google.be/dit-bestaat-niet')
>>> pagina.raise_for_status()
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    pagina.raise_for_status()
  File "C:\Users\arnog\AppData\Local\Programs\Python\Python36\lib\site-packages\requests\mo
dels.py", line 935, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Client Error: Not Found for url: http://www.google.be/dit
-bestaat-niet
```

REQUESTS

- Wegschrijven naar file
  - Open -> schrijf -> sluit – principe
  - Open in ‘wb’ mode (Write Binary)
    - Om unicode te behouden
  - iter_content(100000)
    - Geeft object van het type bytes
    - Gebruik in for-loop om stukken data van
    100000 bytes door te geven



```
import requests

pagina = requests.get('http://shakespeare.mit.edu/romeo_juliet/full.html')
pagina.raise_for_status()
file = open('romeo.txt', 'wb')
for stuk in pagina.iter_content(100000):
    file.write(stuk)
file.close()
```

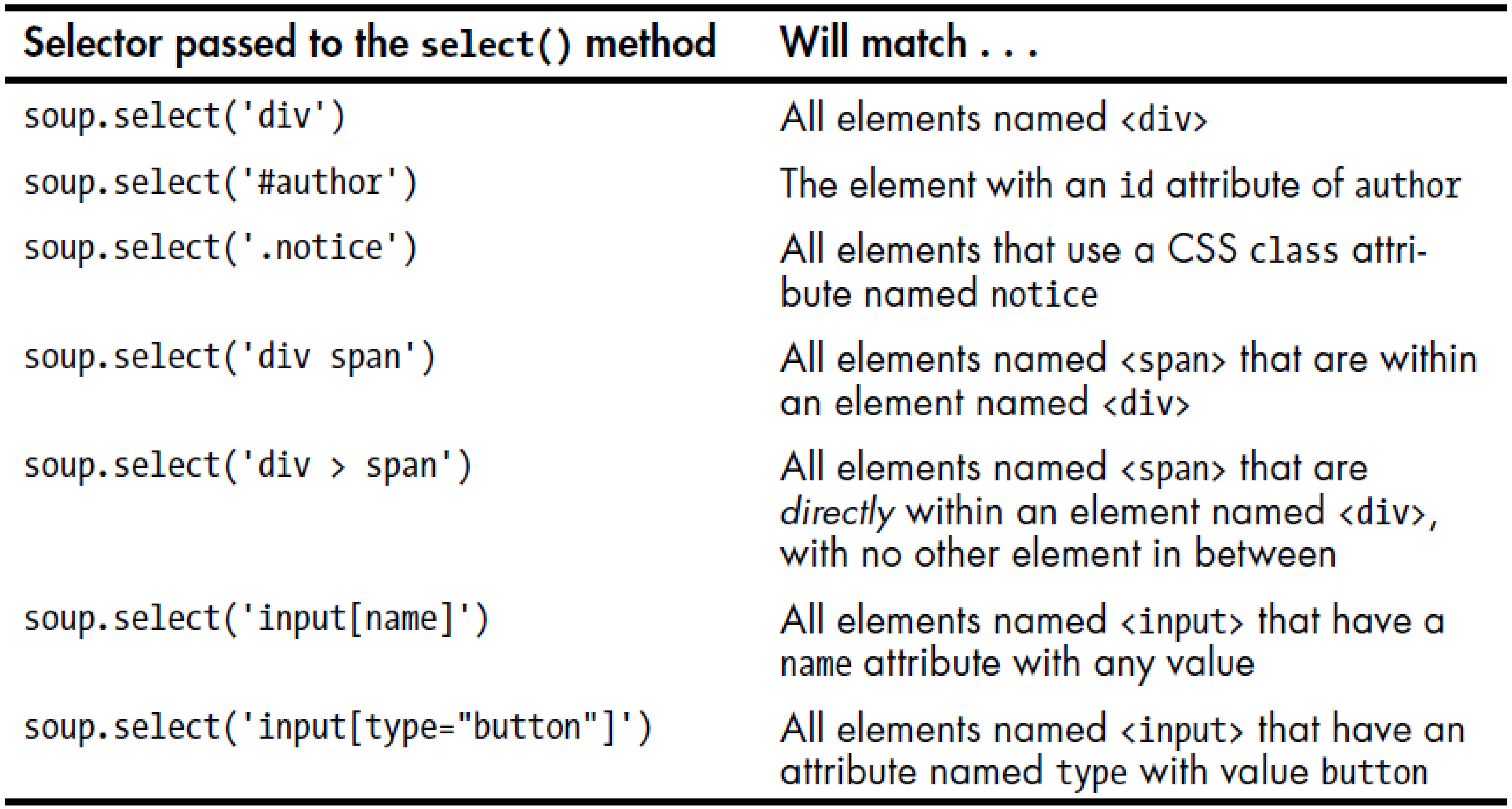
K.H.Kempen en Lessius bundelen de krachten en worden *more*.

THOMAS MORE

BEAUTIFULSOUP

- Moet worden geïmporteerd
  - import bs4
- Doorloopt pagina en selecteert delen
  - Gebruikt CSS selectors!
- Toepassen op tekst van requests.get() object
  - pagesoup = bs4.BeautifulSoup(pagina.text)
  - Geeft bs4 object
- Element selecteren
  - pagesoup.select(css_selector)
    - Bv: pagesoup.select('div') geeft alle div's
    - Bv: pagesoup.select('#titel') geeft alle elementen met id titel
  - Geeft list terug met alle gevonden elementen

BEAUTIFULSOUP



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

BEAUTIFULSOUP

- .getText()
  - Geeft enkel tekst van gevonden element (inner HTML)
- .get()
  - Geeft waarde van attributen bij elementen
  - Wordt toegepast op één element van lijst na .select()
  - Bv: pagesoup[0].get('id') geeft waarde van attribuut id van 1^e element in de list pagesoup

SELENIUM

- Importeren

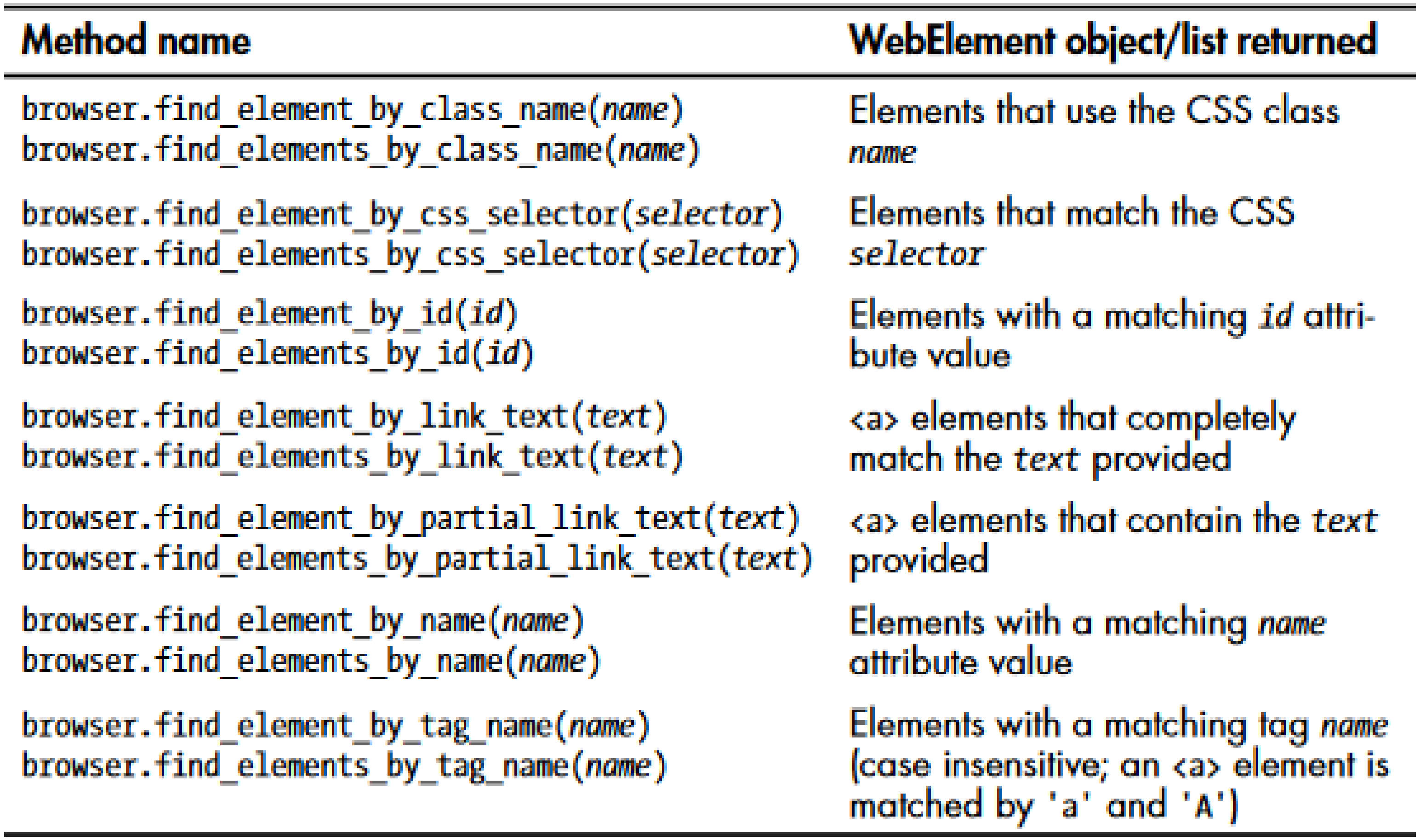
```
>>> from selenium import webdriver
```

- Browser starten met selenium

```
>>> browser = webdriver.Firefox()
```

- WebDriver data type
- Browser.get('http://inventwithpython.com')
  - Opent de gespecifieerde website in geopende browser

SELENIUM



Method name	WebElement object/list returned

`browser.find_element_by_class_name(`*name*`)`
`browser.find_elements_by_class_name(`*name*`)`
	Elements that use the CSS class *name*

`browser.find_element_by_css_selector(`*selector*`)`
`browser.find_elements_by_css_selector(`*selector*`)`
	Elements that match the CSS *selector*

`browser.find_element_by_id(`*id*`)`
`browser.find_elements_by_id(`*id*`)`
	Elements with a matching *id* attribute value

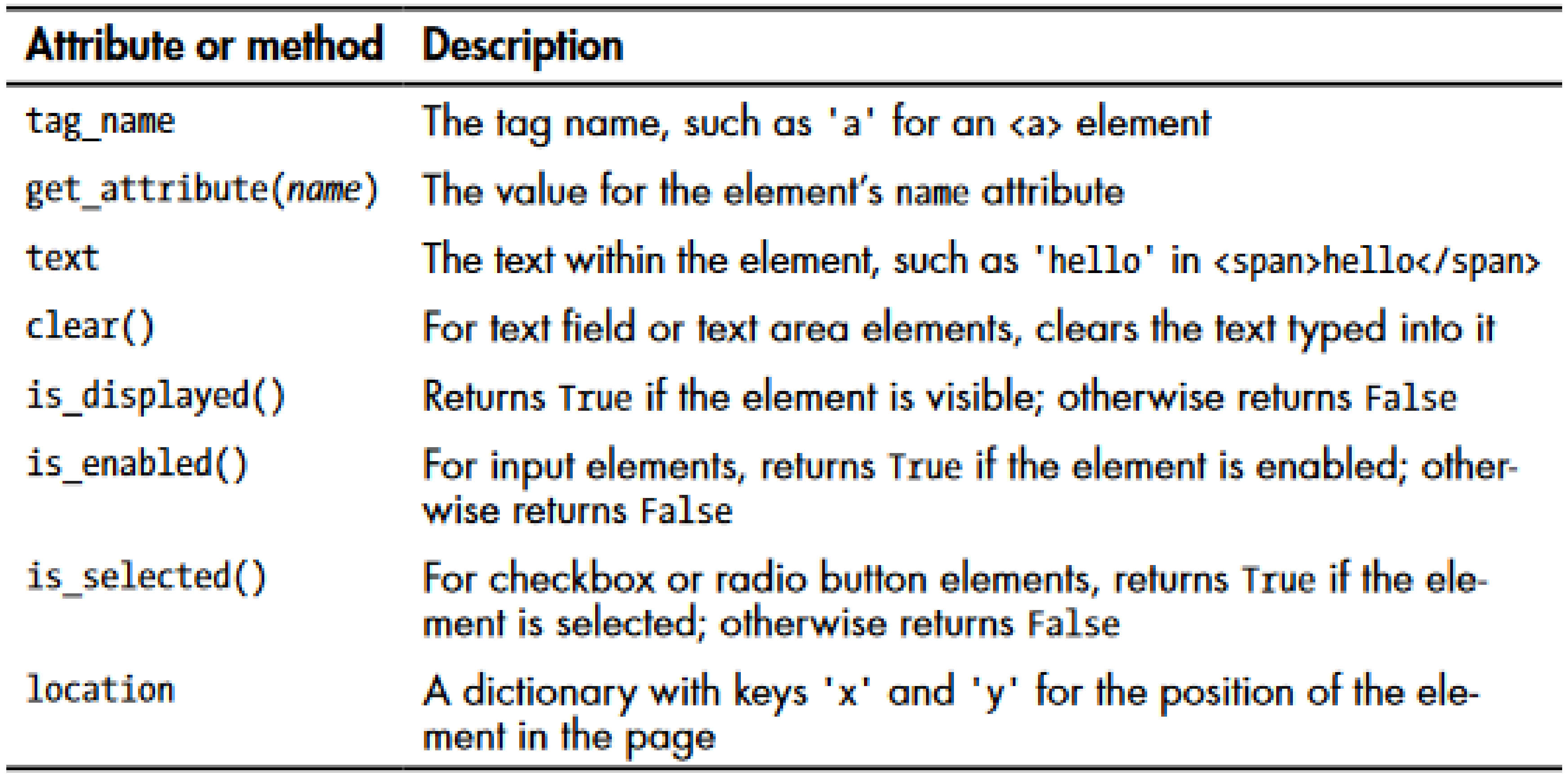
`browser.find_element_by_link_text(`*text*`)`
`browser.find_elements_by_link_text(`*text*`)`
	`<a>` elements that completely match the *text* provided

`browser.find_element_by_partial_link_text(`*text*`)`
`browser.find_elements_by_partial_link_text(`*text*`)`
	`<a>` elements that contain the *text* provided

`browser.find_element_by_name(`*name*`)`
`browser.find_elements_by_name(`*name*`)`
	Elements with a matching *name* attribute value

`browser.find_element_by_tag_name(`*name*`)`
`browser.find_elements_by_tag_name(`*name*`)`
	Elements with a matching tag *name* (case insensitive; an `<a>` element is matched by `'a'` and `'A'`)

SELENIUM



Attribute or method	Description
`tag_name`	The tag name, such as `'a'` for an `<a>` element
`get_attribute(`*`name`*`)`	The value for the element's *name* attribute
`text`	The text within the element, such as `'hello'` in `<span>hello</span>`
`clear()`	For text field or text area elements, clears the text typed into it
`is_displayed()`	Returns `True` if the element is visible; otherwise returns `False`
`is_enabled()`	For input elements, returns `True` if the element is enabled; otherwise returns `False`
`is_selected()`	For checkbox or radio button elements, returns `True` if the element is selected; otherwise returns `False`
`location`	A dictionary with keys `'x'` and `'y'` for the position of the element in the page

SELENIUM

```
from selenium import webdriver
browser = webdriver.Firefox()
browser.get('http://inventwithpython.com')
try:
    elem = browser.find_element_by_class_name('bookcover')
    print('Found <%s> element with that class name!' % (elem.tag_name))
except:
    print('Was not able to find an element with that name.')
```

Output:

```
Found <img> element with that class name!
```

SELENIUM

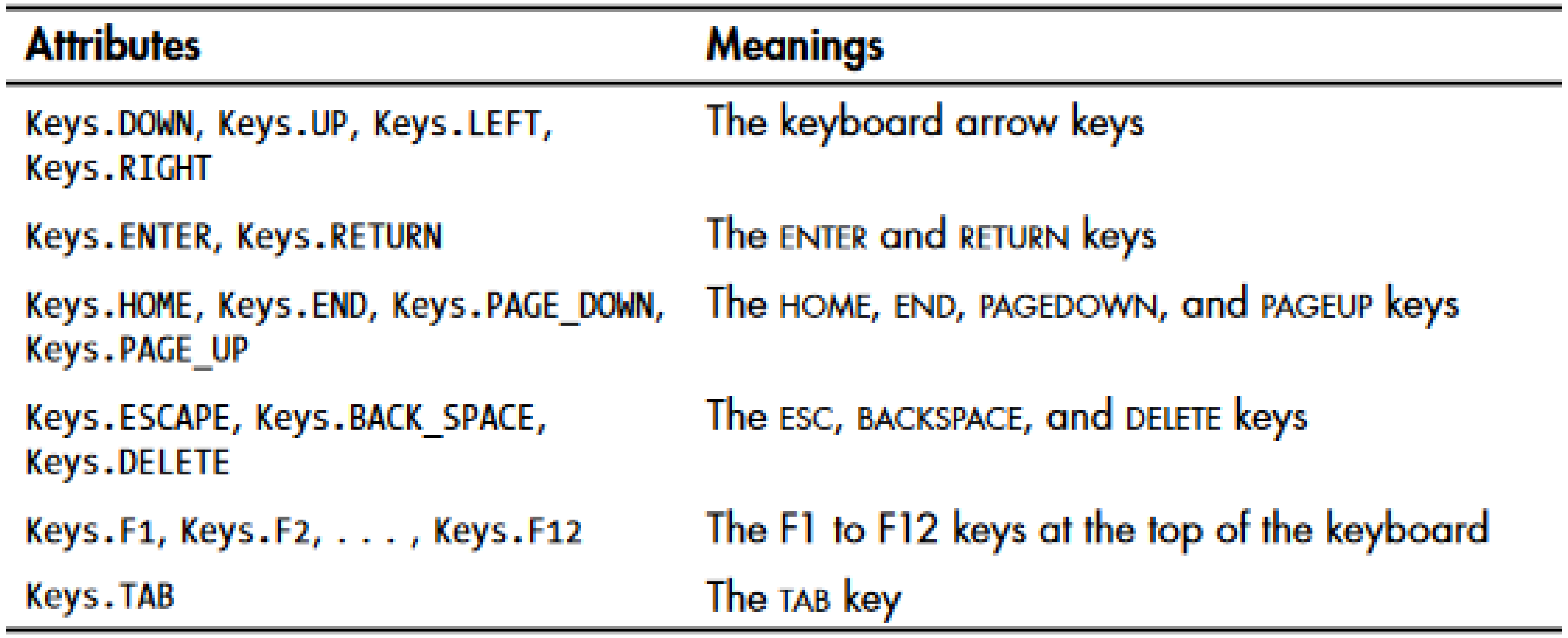
- Klikken op de pagina
  - elem.click()
- Forms invullen
  - Elem.sendkeys()
  - Elem.submit()

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://inventwithpython.com')
>>> linkElem = browser.find_element_by_link_text('Read It Online')
>>> type(linkElem)
<class 'selenium.webdriver.remote.webelement.WebElement'>
>>> linkElem.click()    # follows the "Read It Online" link
```

```
>>> from selenium import webdriver
>>> browser = webdriver.Firefox()
>>> browser.get('http://gmail.com')
>>> emailElem = browser.find_element_by_id('Email')
>>> emailElem.send_keys('not_my_real_email@gmail.com')
>>> passwordElem = browser.find_element_by_id('Passwd')
>>> passwordElem.send_keys('12345')
>>> passwordElem.submit()
```

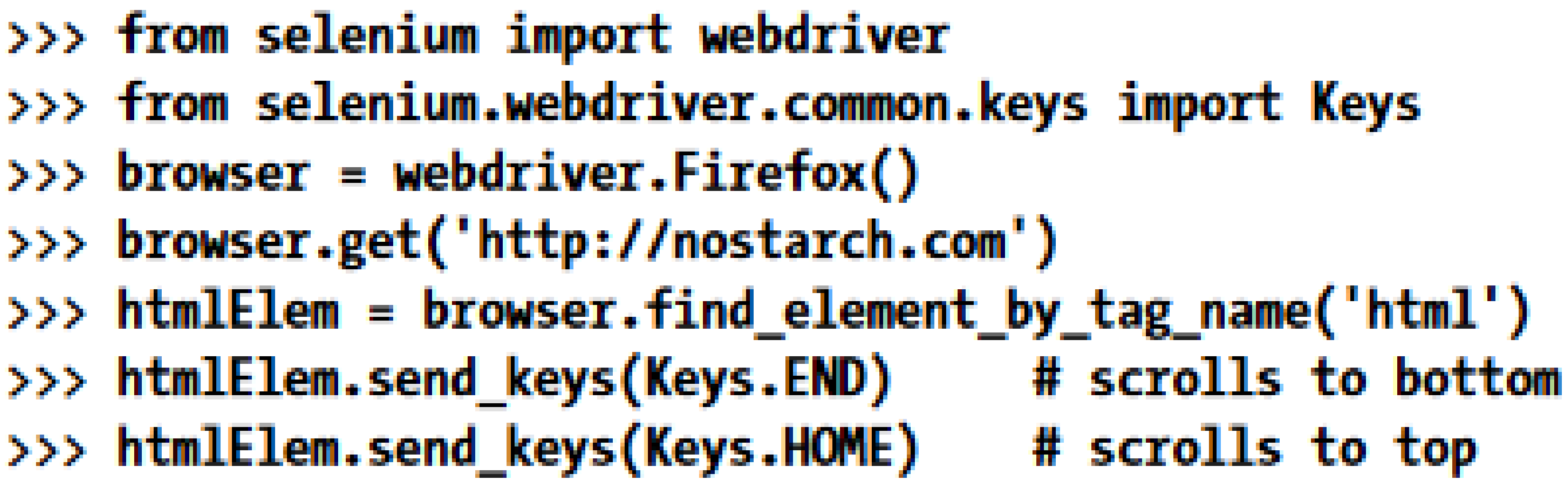
SELENIUM

- Speciale toetsen doorsturen



Attributes	Meanings
Keys.DOWN, Keys.UP, Keys.LEFT, Keys.RIGHT	The keyboard arrow keys
Keys.ENTER, Keys.RETURN	The ENTER and RETURN keys
Keys.HOME, Keys.END, Keys.PAGE_DOWN, Keys.PAGE_UP	The HOME, END, PAGEDOWN, and PAGEUP keys
Keys.ESCAPE, Keys.BACK_SPACE, Keys.DELETE	The ESC, BACKSPACE, and DELETE keys
Keys.F1, Keys.F2, . . . , Keys.F12	The F1 to F12 keys at the top of the keyboard
Keys.TAB	The TAB key

SELENIUM



```
>>> from selenium import webdriver
>>> from selenium.webdriver.common.keys import Keys
>>> browser = webdriver.Firefox()
>>> browser.get('http://nostarch.com')
>>> htmlElem = browser.find_element_by_tag_name('html')
>>> htmlElem.send_keys(Keys.END)      # scrolls to bottom
>>> htmlElem.send_keys(Keys.HOME)     # scrolls to top
```

SELENIUM

- Browser.back()
  - De terugknop klikken
- Browser.forward()
  - De vooruitknop klikken
- Browser.refresh()
  - De vernieuwknop klikken
- Browser.quit()
  - De sluit venster knop klikken

QUIZ

- Beschrijf kort de verschillen (of hun doel) tussen de modules webbrowser, requests, beautifulsoup en selenium.
- Op welke twee manieren kan je controleren of het downloaden van de pagina is gelukt?
- Hoe vind je (met beautifulsoup) een button met als attribuut *value* en als waarde *favoriet*?
- Stel je hebt een beautifulsoup tag object *spam* met als inhoud <*div*>*Hello world!*</*div*>. Hoe krijg je de string “Hello world!” dan uit deze variabele spam?
- Wat is het verschil tussen find_element_* en find_elements_*?
- Welke methodes heeft een Selenium WebElement om muisklikken en toetsen van het toetsenbord te simuleren?
- Hoe kan je de terug, vooruit en herlaadknoppen simuleren?