Statistical Computing
Lesson 4

Gabriele Rovigatti

Bank of Italy

March 19, 2021

This Course

- Python basics
  - General Framework
  - Syntax
  - Modules / Libraries
  - Functions
- Webscraping
  - Data Storage
  - HTML Syntax
  - Data Collection
  - `Requests` Package
    - GET requests
    - Save Scraped Data
    - WikiScrape
  - `Webdriver` Package
- Machine Learning?

Today

- Python basics
  - General Framework
  - Syntax
  - Modules / Libraries
  - Functions
- **Webscraping**
  - Data Storage
  - HTML Syntax
  - Data Collection
  - `Requests` **Package**
    - **GET requests**
    - **Save Scraped Data**
    - **WikiScrape**
  - `Webdriver` Package
- Machine Learning?

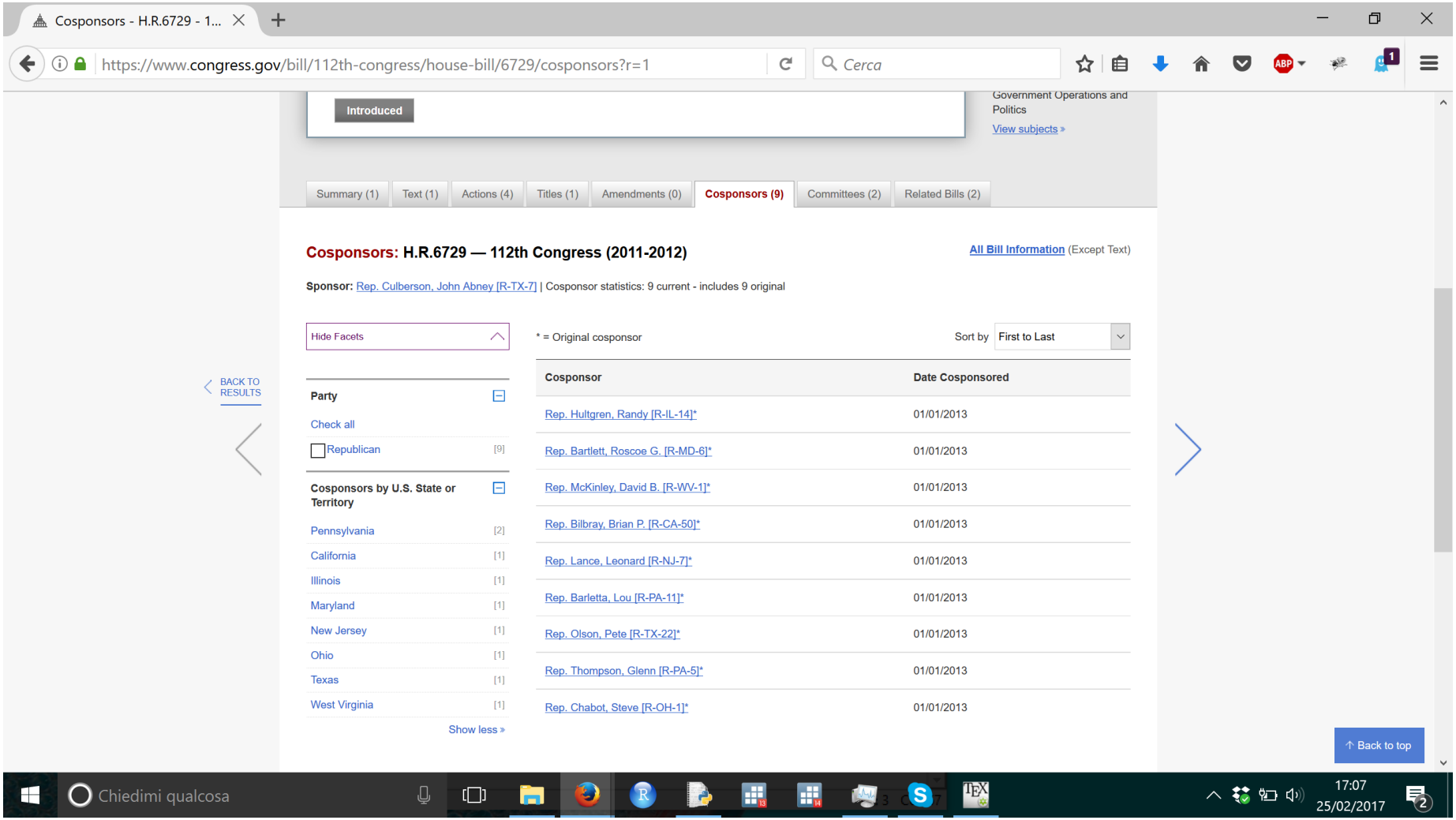
HTML basics

HyperText Markup Language is the language most web pages are built with. It is **not** a programming language - it is a markup language that tells a browser how to layout content.

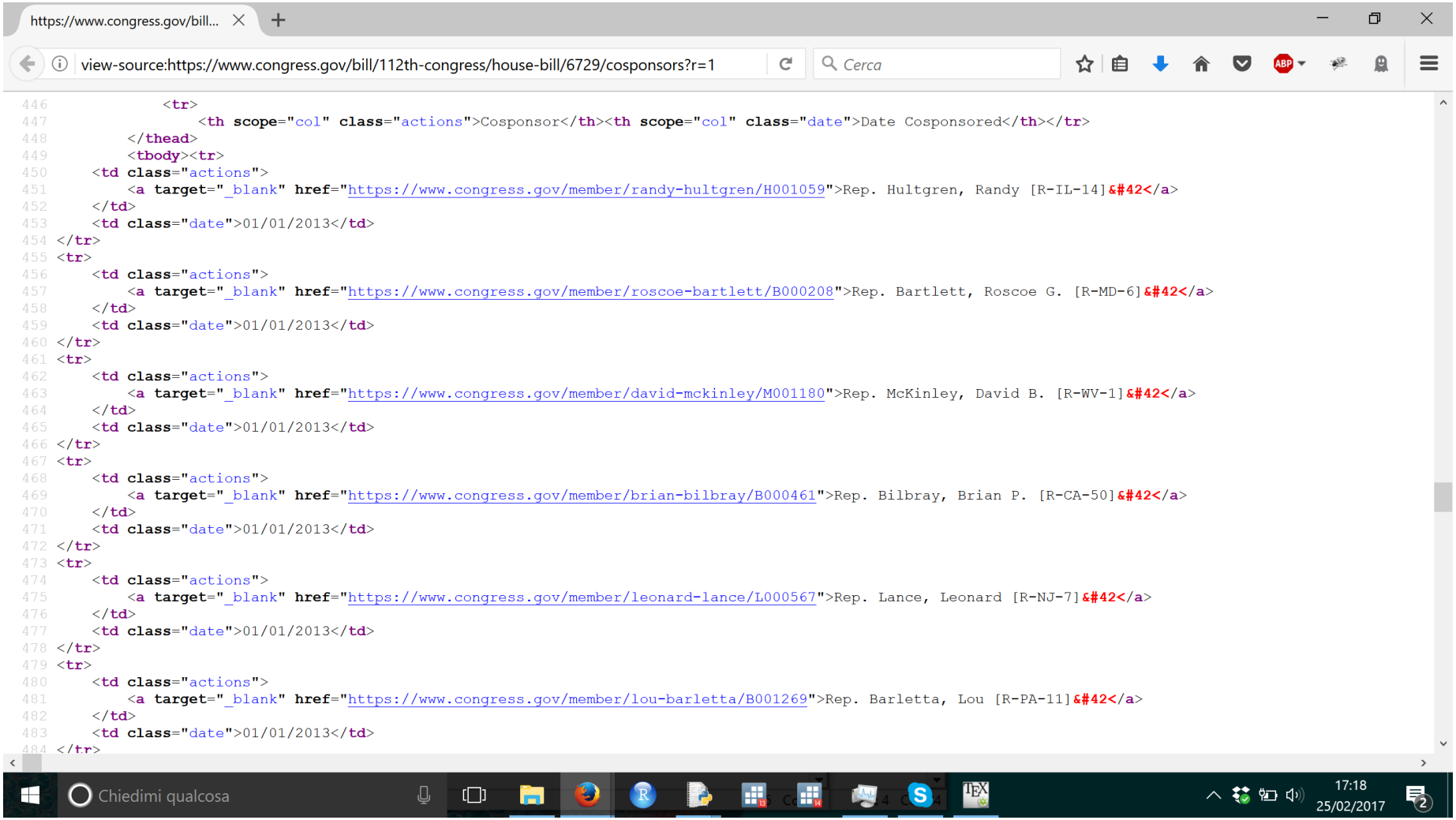
It consists of elements called `tags`. A simple example reads

```
<html>
 <head>
 </head>
 <body>
 <p>
 This generates a line of text
 <a href="https://www.python.org">This one generates a link to www.python.org</a>
 </p>
 </body>
</html>
```

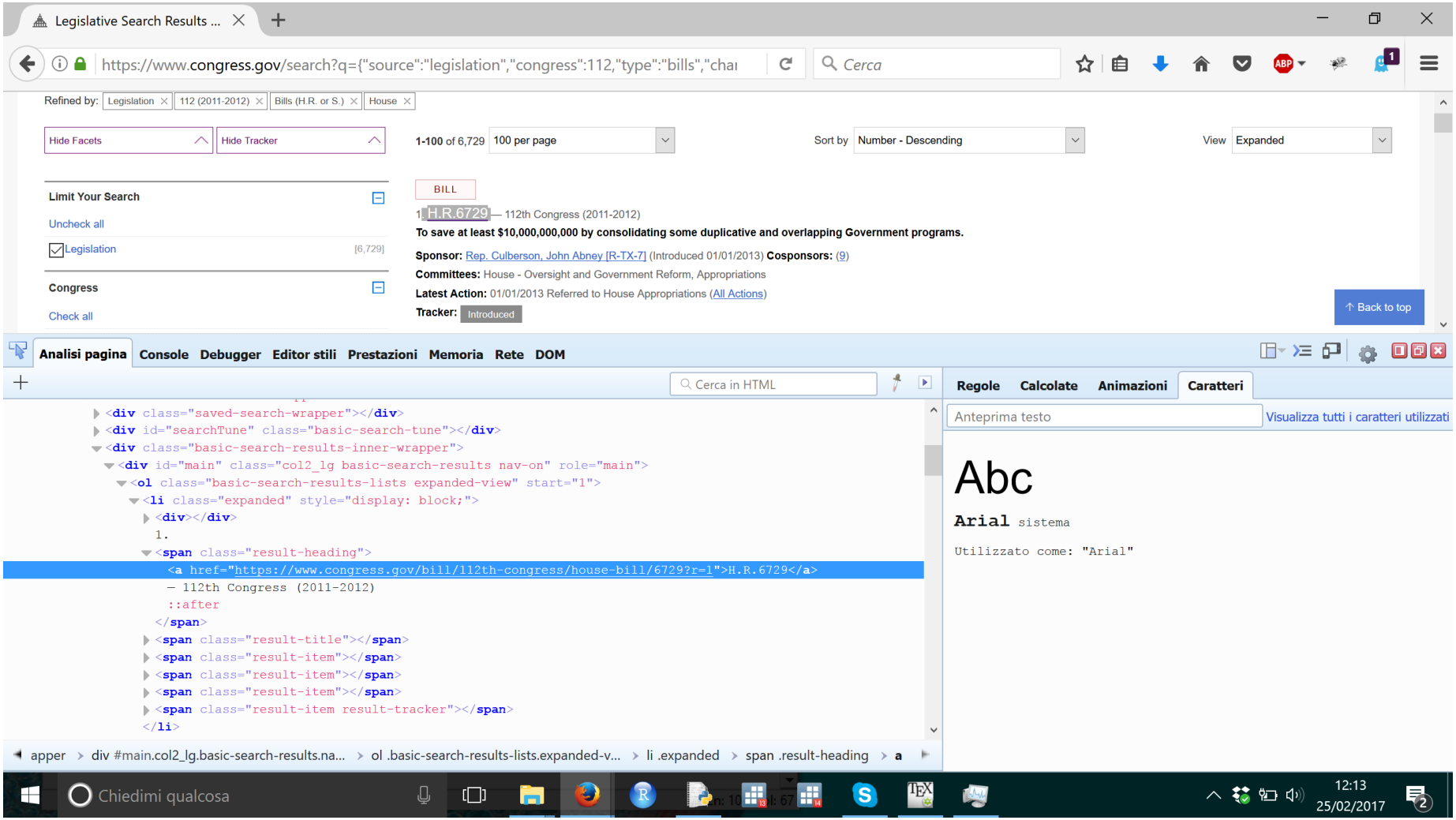
HTML-structured pages: what you see



HTML-structured pages: what's beneath

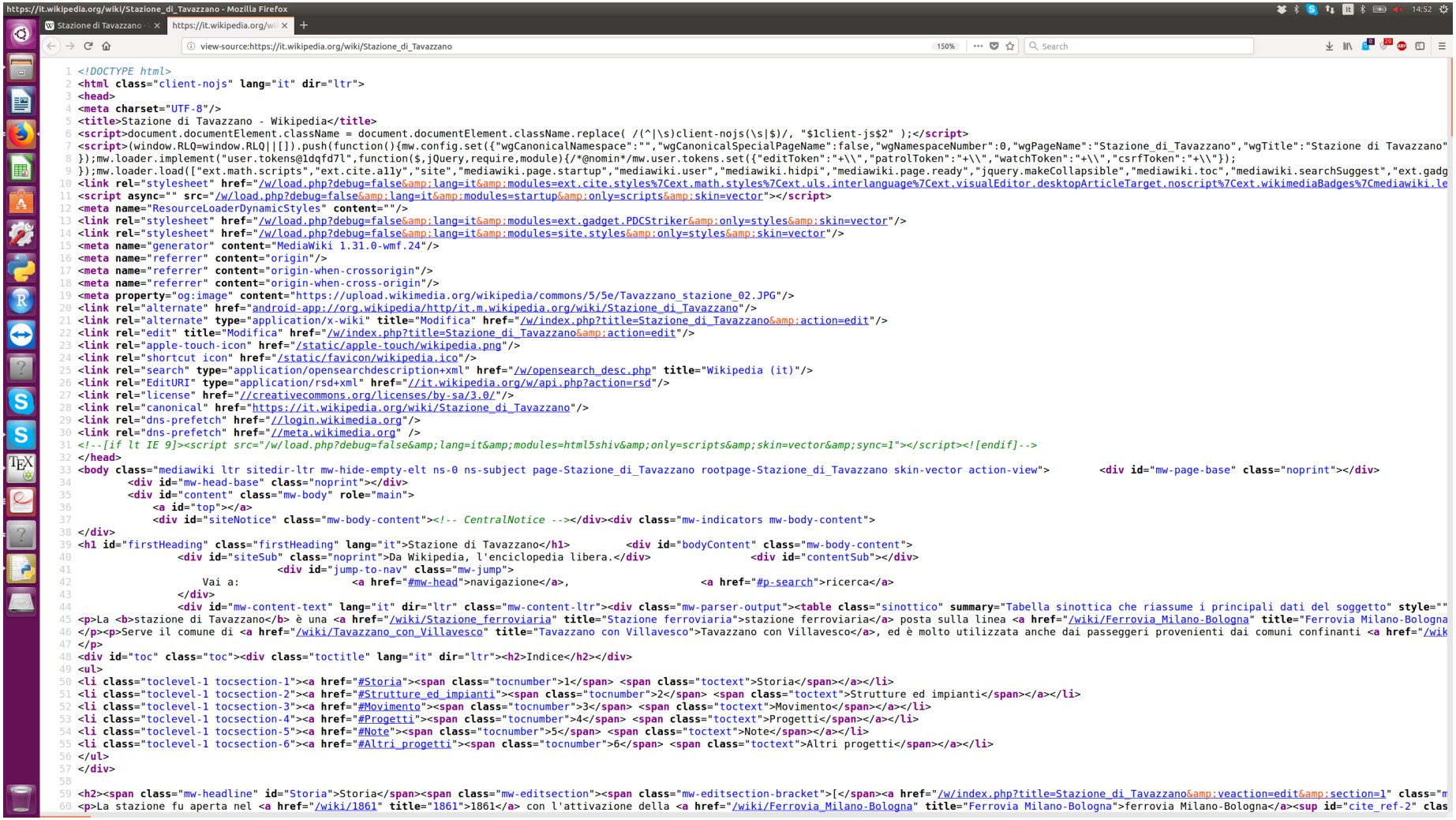


Getting to know your enemy - "Inspect" Webpages



Last year's example: Wikipedia

Last year's example: Wikipedia's HTML Structure



Last year's example: Inspect Wikipedia

Last week's assignment - **Mandatory**

- Write a function called **CipCiop** that takes a number and
  - If the number is divisible by 3, it should return "Cip";
  - If it is divisible by 5, it should return "Ciop";
  - If it is divisible by both 3 and 5, it should return "CipCiop";
  - Otherwise, it should return the original number.
- Write a function that takes as input an integer (called *limit*) and prints all the prime numbers between 0 and limit.
- Program a Rock-Paper-Scissors game, with the input() method (twice!), comparing the plays and printing the winner. (Yes, it is the same exercise for which you have the solutions, I will check your code. Try not to copy!)

Last week's assignment - **Complimentary**

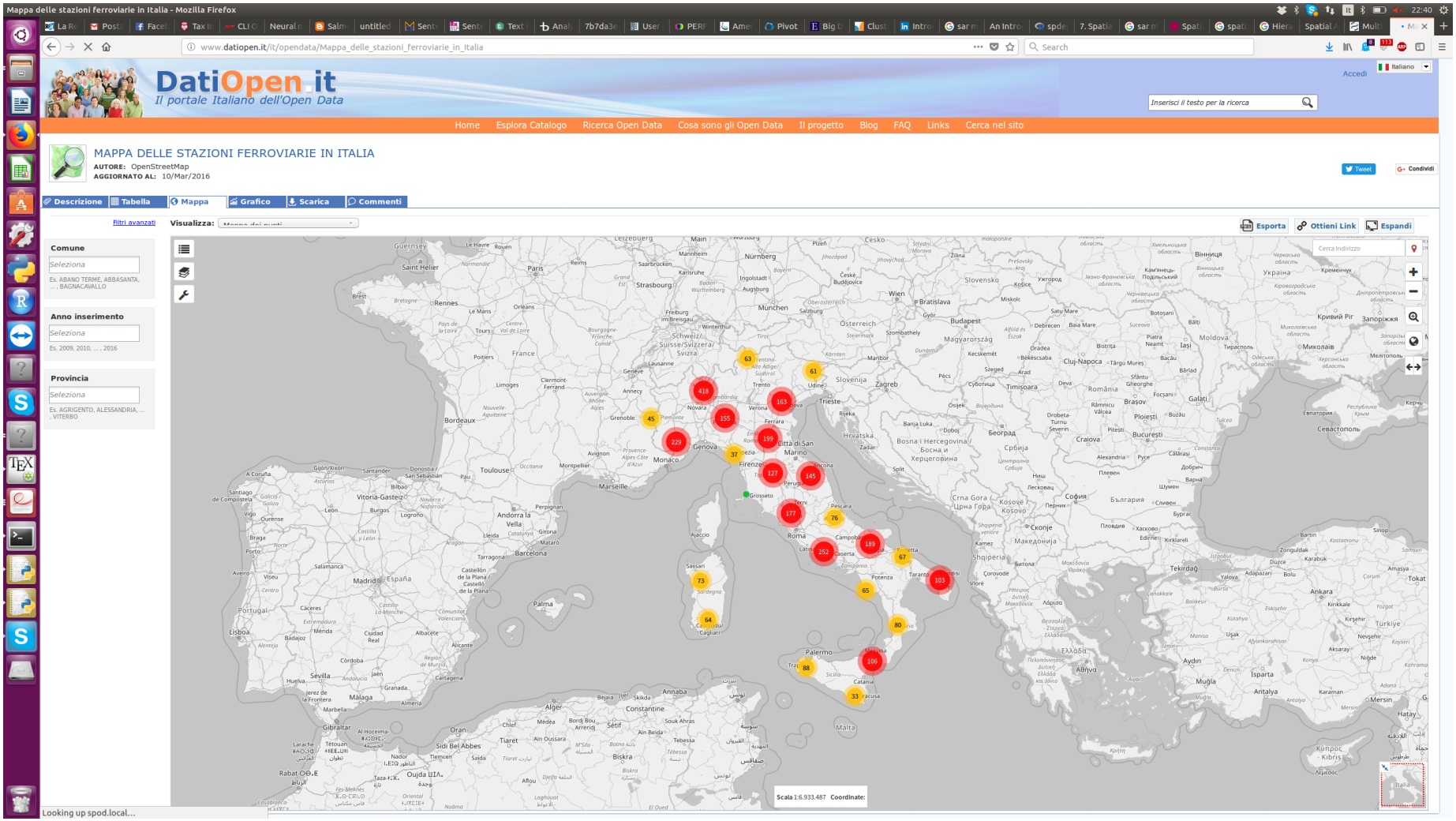
- Write a program that takes two lists as input and returns the common elements between the two lists. Make sure your program works on two lists of different sizes. Solution
- Write a one-line python code that takes a numeric list as input and returns the even elements of the list; in **one** line. Solution

Set up a project

- Final Goal: scrape info on **all** Italian railstations
  - Station Name
  - Station Municipality
  - Opening Date
  - Closing Date - if applicable
  - Municipal Istat Code
  - Railway Line(s)
- Requirements:
  - List of all Italian train stations
  - Relative links to Wikipedia pages

Train stations data

- http://www.datiopen.it/it/opendata/Mappa_delle_stazioni_ferroviarie_in_Italia



Train stations data / 2

Municipality	Province	Region	Name	Insertion Date	Insertion Date & Hour	OpenStreetMap ID	Longitude	Latitude
ALTRO	ALTRO	ALTRO	Albonago	2014	2014-11-11T16:22:37Z	984003073	8.9750146	46.0085186
ALTRO	ALTRO	ALTRO	Camedo	2014	2014-05-04T21:49:40Z	2836876118	8.610916	46.1547927
Alpignano	TORINO	Piemonte	Alpignano	2013	2013-05-04T21:56:45Z	1588214497	7.5238245	45.0912816
Avigliana	TORINO	Piemonte	Avigliana	2015	2015-05-11T21:25:04Z	1598967748	7.4009405	45.0853893
Balangero	TORINO	Piemonte	Balangero	2011	2011-08-14T22:12:24Z	822596562	7.5194836	45.265646
Bardonecchia	TORINO	Piemonte	Bardonecchia	2015	2015-05-11T21:25:04Z	2043486315	6.7097395	45.0764633
Borgaro Torinese	TORINO	Piemonte	Borgaro Torinese	2015	2015-05-11T21:25:02Z	3511147159	7.6487388	45.1532959
Borgofranco d'Ivrea	TORINO	Piemonte	Borgofranco d'Ivrea	2012	2012-11-20T09:13:02Z	1640450736	7.8546427	45.5147371
Borgone Susa	TORINO	Piemonte	Borgone di Susa	2015	2015-05-11T21:25:05Z	1598967749	7.2360913	45.1233785
Bosconero	TORINO	Piemonte	Bosconero	2011	2011-08-12T13:48:27Z	1262195587	7.7618626	45.2675134
Brandizzo	TORINO	Piemonte	Brandizzo	2014	2014-02-06T10:42:06Z	2043486321	7.8411549	45.1787318
Bruzolo	TORINO	Piemonte	Bruzolo	2015	2015-05-11T21:25:05Z	1598967750	7.2010571	45.1312692
Bussoleno	TORINO	Piemonte	Bussoleno	2015	2015-05-11T21:25:05Z	1446542028	7.1448124	45.1396163
Caluso	TORINO	Piemonte	Caluso	2012	2012-11-20T09:12:59Z	1003779384	7.9025604	45.3031447
Caluso	TORINO	Piemonte	Rodallo	2012	2012-11-20T09:12:59Z	1640251816	7.8780343	45.2787068

- 2,999 train stations
- Information already usable in tabular format
- Use station names to build and scrape urls

Train stations data / 2

Municipality	Province	Region	Name	Insertion Date	Insertion Date & Hour	OpenStreetMap ID	Longitude	Latitude
ALTRO	ALTRO	ALTRO	Albonago	2014	2014-11-11T16:22:37Z	984003073	8.9750146	46.0085186
ALTRO	ALTRO	ALTRO	Camedo	2014	2014-05-04T21:49:40Z	2836876118	8.610916	46.1547927
Alpignano	TORINO	Piemonte	Alpignano	2013	2013-05-04T21:56:45Z	1588214497	7.5238245	45.0912816
Avigliana	TORINO	Piemonte	Avigliana	2015	2015-05-11T21:25:04Z	1598967748	7.4009405	45.0853893
Balangero	TORINO	Piemonte	Balangero	2011	2011-08-14T22:12:24Z	822596562	7.5194836	45.265646
Bardonecchia	TORINO	Piemonte	Bardonecchia	2015	2015-05-11T21:25:04Z	2043486315	6.7097395	45.0764633
Borgaro Torinese	TORINO	Piemonte	Borgaro Torinese	2015	2015-05-11T21:25:02Z	3511147159	7.6487388	45.1532959
Borgofranco d'Ivrea	TORINO	Piemonte	Borgofranco d'Ivrea	2012	2012-11-20T09:13:02Z	1640450736	7.8546427	45.5147371
Borgone Susa	TORINO	Piemonte	Borgone di Susa	2015	2015-05-11T21:25:05Z	1598967749	7.2360913	45.1233785
Bosconero	TORINO	Piemonte	Bosconero	2011	2011-08-12T13:48:27Z	1262195587	7.7618626	45.2675134
Brandizzo	TORINO	Piemonte	Brandizzo	2014	2014-02-06T10:42:06Z	2043486321	7.8411549	45.1787318
Bruzolo	TORINO	Piemonte	Bruzolo	2015	2015-05-11T21:25:05Z	1598967750	7.2010571	45.1312692
Bussoleno	TORINO	Piemonte	Bussoleno	2015	2015-05-11T21:25:05Z	1446542028	7.1448124	45.1396163
Caluso	TORINO	Piemonte	Caluso	2012	2012-11-20T09:12:59Z	1003779384	7.9025604	45.3031447
Caluso	TORINO	Piemonte	Rodallo	2012	2012-11-20T09:12:59Z	1640251816	7.8780343	45.2787068

- 2,999 train stations
- Information already usable in tabular format
- Use station names to build and scrape urls

Inspect: Point to meaningful information

Wikipydia: scrape html pages

```
# Import the required libraries
import requests
from bs4 import BeautifulSoup # new
# specify the url to be scraped: Stazione di Tavazzano
wikiurl = 'https://it.wikipedia.org/wiki/Stazione_di_Tavazzano'
# GET method: scrape the url
r = requests.get(wikiurl)
html_page = r.text
# read the page with BeautifulSoup
soup = BeautifulSoup(r.text, 'html.parser')
# scrape the title name, and the other info
station_name = soup.find('h1', {'id' : 'firstHeading'}).text # station name
latitude = soup.find('span', {'class' : 'latitude'}).text # latitude
longitude = soup.find('span', {'class' : 'longitude'}).text # longitude
start_date = soup.find('th', string='Attivazione').findNext('a').text # sta
lines = soup.find('th', string='Linee').findNext('a').text # train line
status = soup.find('th', string='Stato attuale').findNext('td').text # acti
print([station_name,latitude,longitude,start_date,lines,status])
```

Write down in readable formats - CSV

Scrape www.congress.gov/ with `trumpy.py`

```
import csv
import requests
from bs4 import BeautifulSoup
from os.path import dirname, abspath
### CONGRESS 112 ###
basedir = dirname(dirname(abspath(__file__))) + '/congress'
o112 = basedir + '/output.csv'
hrurl = 'https://www.congress.gov/bill/112th-congress/house-bill/6729/cosponsors?r=1'
hrpage = requests.get(hrurl)
data = hrpage.text
soup = BeautifulSoup(data,'lxml') #here we make BS read the html page
with open(o112,'w') as f: #open the outpufile in .csv
    fwriter = csv.writer(f, delimiter = ';') #declare the writing object
    fwriter.writerow(['Congress','BillType','BillNumber','cosponsorhref']) #write headers
    for l in soup.findAll('table', { 'class' : 'item_table'}): #identify cosponsors' table
        for a in l.findAll('a', href = True, text = True, target = '_blank'): #cosponsors' entries
            cosponsorhref = a['href']
            print([112,'HR',6729,cosponsorhref])
            fwriter.writerow([112,'HR',6729,cosponsorhref])
    f.close()
```

Run

Wikipydia: automate the analysis of rail stations

```
# Import the required libraries
import requests
from bs4 import BeautifulSoup
import os
import csv
# define directories
basedir = '/home/gabriele/Desktop/Dropbox (Francesco Decarolis)/Lessons/Tor
stationdir = os.path.join(basedir, 'Mappa-delle-stazioni-ferroviarie-in-Ita
outputdir = os.path.join(basedir, 'railstation_data.csv')
wrong_counter = 0
with open(stationdir, 'r+', encoding='latin1') as f: # without the encoding
    reader = csv.reader(f, delimiter=';') # in this file's case, the delimi
    with open(outputdir, 'w+') as g:
        writer = csv.writer(g)
        writer.writerow(['Comune','Provincia','Regione','Nome','Longitudine
        for i, line in enumerate(reader): # going through all stations in t
         # DOWNLOAD ROUTINE #
     g.close()
f.close()
```

Wikipydia: automate the analysis of rail stations

```
for i, line in enumerate(reader): # going through all stations in the list
            if i <= 17:
                continue # skip the first iterations: title + foreign stati
            station_name = line[3] # line is the (iterable) .csv row: the 4
            url_name = station_name.replace(' - ', '-').replace(' ','_') #
            wikiurl = 'https://it.wikipedia.org/wiki/Stazione_di_' + url_na
            print(wikiurl)
            # GET method: scrape the url
            r = requests.get(wikiurl)
            # read the page with BeautifulSoup
            soup = BeautifulSoup(r.text, 'html.parser')
```

Wikipydia: automate the analysis of rail stations

```
try:
    status = soup.find('th', string='Stato attuale').findNext('td').text
except AttributeError:
    status = '-'
try:
    start_date = soup.find('th', string='Attivazione').findNext('a').text
except AttributeError:
    start_date = '.'
try:
    lines = soup.find('th', string='Linee').findNext('a').text
except AttributeError:
    lines = '-'
if lines == '-' and start_date == '.' and status == '-': # the link is brok
    lines = 'CHECK'
    start_date = 'CHECK'
    status = 'CHECK'
    wron_counter += 1
printline = line[0:4]+line[7:9]+[status]+[start_date]+[lines]+[wikiurl] # i
writer.writerow(printline)
```

Scraped Data

Municipality	Province	Region	Name	Longitude	Latitude	Status	Opening Date	Lines
Airasca	TORINO	Piemonte	Airasca	7.4832665	44.9280462	in uso	1854	Ferrovia Torino-Torre Pellice
Alpignano	TORINO	Piemonte	Alpignano	7.5238245	45.0912816	in uso	0	Frejus
Avigliana	TORINO	Piemonte	Avigliana	7.4009405	45.0853893	in uso	0	Frejus
Balangero	TORINO	Piemonte	Balangero	7.5194836	45.265646	In uso	1876	Ferrovia Torino-Ceres
Bardonecchia	TORINO	Piemonte	Bardonecchia	6.7097395	45.0764633	in uso	1871	Ferrovia del Frejus
Borgaro Torinese	TORINO	Piemonte	Borgaro Torinese	7.6487388	45.1532959	In uso	1871	Ferrovia Torino-Ceres
Borgofranco d'Ivrea	TORINO	Piemonte	Borgofranco d'Ivrea	7.8546427	45.5147371	CHECK	CHECK	CHECK
Borgone Susa	TORINO	Piemonte	Borgone di Susa	7.2360913	45.1233785	CHECK	CHECK	CHECK
Bosconero	TORINO	Piemonte	Bosconero	7.7618626	45.2675134	in uso	0	Canavesana
Brandizzo	TORINO	Piemonte	Brandizzo	7.8411549	45.1787318	In uso	1856	Torino-Milano
Bruzolo	TORINO	Piemonte	Bruzolo	7.2010571	45.1312692	in uso	0	Frejus
Bussoleno	TORINO	Piemonte	Bussoleno	7.1448124	45.1396163	in uso	1854	Torino-Modane
Caluso	TORINO	Piemonte	Caluso	7.9025604	45.3031447	in uso	0	Chivasso-Aosta
Caluso	TORINO	Piemonte	Rodallo	7.8780343	45.2787068	in uso	0	Chivasso-Aosta
Cambiano	TORINO	Piemonte	Cambiano-Santena	7.7731065	44.9664372	in uso	1849	Torino-Genova
Candia Canavese	TORINO	Piemonte	Candia Canavese	7.891791	45.3300205	in uso	0	Chivasso-Aosta
Candiolo	TORINO	Piemonte	Candiolo	7.5990181	44.9611065	in uso	1854	Ferrovia Torino-Torre Pellice
Carmagnola	TORINO	Piemonte	Carmagnola	7.7278083	44.8470455	in uso	1853	Torino-Savona
Caselle Torinese	TORINO	Piemonte	Caselle Aeroporto	7.6416415	45.1918384	In uso	2001	Ferrovia Torino-Ceres

- Works pretty well!
- Still needs refinements: station names for URLs, multiple lines, etc.
- Is the station list comprehensive?

Requests - POST method

Scrape www.semrush.com with pygliaSEM.py

```
import requests
# list credentials and generate the login dict
username = "myloginmail@notmail.com"
password = "verysecretpassword"
login_data = {
    'email': username,
    'password': password
}
# define the base URL --> the one in which we need to login
base_url = 'https://www.semrush.com/json_users/login'
# start a request session and POST the login data
s = requests.session() #declare a SESSION of requests
r = s.post(base_url, data = login_data) #this way the session is logged in!
# proceed in parsing the .csv file
r = __csvParse__(domain,s,database,timestamp,'domain_adwords')
```

Requests - GET method with parameters

Scrape www.semrush.com with pygliaSEM.py

```
def __csvParse__(domain,s,database,timestamp,domain_type):
    domain_url = 'https://www.semrush.com/info/' + domain + 'ad_domain'
    try:
        # initialize parameter dict and launch GET request
        params = {
            'action' : 'report_tpye',
            'database' : 'domain_name',
            'rnd_m' : 'rnd_number',
            'key' : 'some_key',
            'domain' : domain,
            'type' : 'domain_whatever',
            'display_filter' : '',
            'export_hash' : 'hash_key',
            'display_sort' : 'nq_desc',
            'export_columns' : 'column_list',
            'export_decode' : '',
            'export_escape' : '',
            'exchange_rate' : '',
            'currency' : 'usd',
            'export' : 'csv'
            }
        # download the .csv
        r = s.get('https://' + timestamp + database + '.backend.semrush.com/', params = params)
    except:
        r = ''
    return r
```

Webdriver - when URLs are not enough

Scrape www.coeweb.istat.it with pystat.py

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
import time
from os.path import dirname, abspath
######## ROUTINE ##########
search_url = 'https://www.coeweb.istat.it/predefinite/tutto_paese_merce.asp?livello=L2&riga=MERCE&territorio=S&
basedir = dirname(dirname(abspath(__file__))) + "/coeweb"
downloadedname = basedir + '/tabella.xls'
# FF settings
profile = webdriver.FirefoxProfile()
profile.set_preference('browser.download.folderList', basedir) # custom location
profile.set_preference('browser.download.dir', basedir)
profile.set_preference('browser.helperApps.neverAsk.saveToDisk', 'text/csv')
browser = webdriver.Firefox(profile)
browser.implicitly_wait(6) # wait some seconds for uploading the page
browser.get(search_url) # go to the page to be scraped
browser.find_element_by_name("XLS").click() # choose Excel format
browser.find_element_by_name("MESE").send_keys('Ultimo Disponibile') # Month
browser.find_element_by_id("radio5").click() # Cumulative
browser.find_element_by_name("B2").click() # download
time.sleep(10)
browser.close()
```

Run

Countless Applications!

- Istat - Coeweb Code
- Consiglio di Stato - Pareri Code
- Corte dei Conti - Sentenze Code
- Repubblica - Articles Code
- Corriere della Sera - Articles Code
- American Congress - all bills / sponsors / cosponsors Code
- Californian Schools - repolist

Part 1: webscraping with requests

The goal of the python code will be to automatically download and save to local disk all .xls files available at *http://www.cde.ca.gov/ds/sp/ai/*. These are clustered in three groups:

- SAT test results 1998-2016
- ACT test results 1998-2016
- AP test results 1998-2016

each consisting of 18 individual .xls files. The scraper should parse the webpage (with `requests.get()`), identify the links of the files (with `BeautifulSoup()`), and save the contents of each individual URL in a distinct file (with `requests.get()` and `.write()`).

Tips

- Pay attention to the embedded URLs: from 2015-2016 to 2013-2014 they are complete, whereas from 2012-2013 on they only store the final "branches", which need the base url to properly work!
- You may find useful to use the names provided in the URL to save the relative file on your local disk: the `basename()` function from `os.path` will help:

```
from os.path import basename
```

Solution 2 - 1

```python
# define the lists to be compared
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
# for loop, if, and, or solution
import sys
def ls_overlap(A,B):
    if not isinstance(A,list) or not isinstance(B, list): # check that both inputs are lists
        sys.exit("both inputs must be lists!")
    res = []
    for e in set(A): # for every unique elment in A
        if e in B and B not in res: # if A element is in B and not already in res
            res.append(e)
    return(res)
res = ls_overlap(a,b)
print(res)
# use "set()" and "intersection()"
res = list(set(a).intersection(set(b)))
print(res)
```

back

Solution 2 - 2

```
# define the list
ls_a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
# one-line solution: list comprehension, for, in and modulus
res = [x for x in ls_a if x % 2 == 0]
print(res)
# for loop, in, if and append
res = []
for a in ls_a:
    if a % 2 == 0:
        res.append(a)
print(res)
```

back