scrapITRA: an introduction

Last revision:

In [8]:

```
import pandas as pd; pd.to_datetime('today')
```

Out[8]:

```
Timestamp('2019-02-26 10:39:45.646664')
```

The website

ITRA (International Trail Running Association) is the world's largest trail running association. Its website https://itra.run contains information about the runners that took part in races included in the circuit.

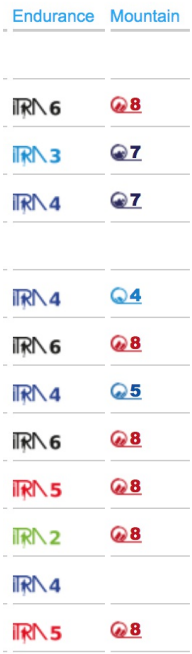
But what is an ultramarathon? From Wikipedia: any footrace longer than the traditional marathon length of 42.195 kilometres.

The majority of the most famous ultra-races are part of ITRA, such as **UTMB, Lavaredo Ultra-Trail, Western States, Marathon des Sables**... in reality also non-ultra, but still on trail, competitions are included. However, some (mostly American) races are not: HardRock 100, BadWater cup...

Their datasets contains **~1.4 million runners**.

ITRA index:

- **race index**: the ITRA race index is an evaluation of the difficulty of the race based on distance, elevation, number of stages, loops, and other criteria. Every race that the runner has completed corresponds to a certain numbers of ITRA points that can be used to participate to other races. For instance, UTMB requires 15 points achieved in maximum 3 races.



- **runner index**: the ITRA runner index is an evaluation of the runner's ability compared to the best performing runners (benchmark). This data is typically available only for ITRA members.

2018-07-29	Skyrun Comapedrosa 2018	21km / 2300m+	02:33:18	1 / 536	1	930
2018-08-12	Sierre-Zinal 2018	30km / 2300m+	02:31:40	1 / 1452	1	926

Only the five best results of the last 36 months are taken into account in the calculation of the ITRA performance index.
More info on the method

.

Goal and ingredients

I am interested in **statistical analysis on ultra-running data**. However, ITRA does not have an open-access data.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

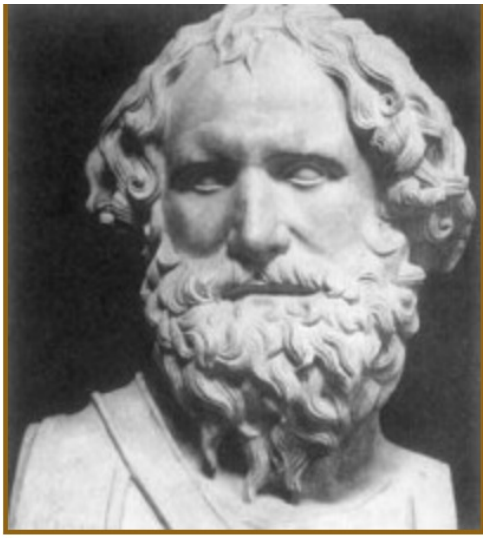
.

.

.

.

.



web scraping!

Web scraping!

Web scraping!

Web scraping!

Web scraping!

Web scraping!

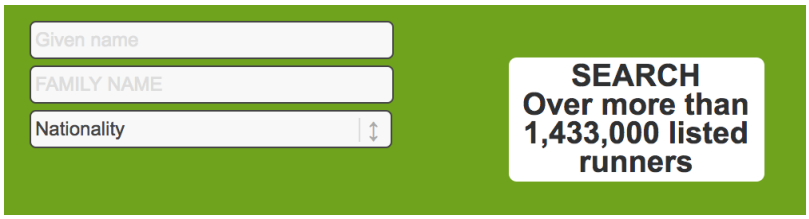
Web scraping!

.....More?.....

Web scraping!

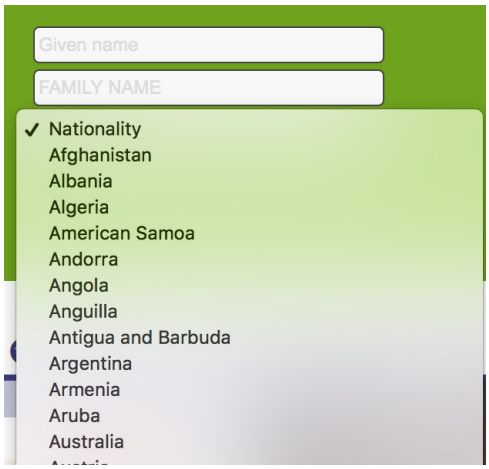
Tools:

- **Selenium**: from Wikipedia: "Selenium is a portable framework for testing web applications. Selenium provides a playback (formerly also recording) tool for authoring functional tests without the need to learn a test scripting language". In other terms, we can automate manual operations with Selenium: say that you have this in front of you



so first you need to fill in some information

choosing some option from the list



and then push the button

the Python implementation of Selenium can automate all these operations.

- **BeautifulSoup**: from crummy.com: "Beautiful Soup is a Python library for pulling data out of HTML and XML files.". Beautifulsoup extracts the HTML/XML page and helps you find the desired elements. For example, if we want to extract something from here

Beautifulsoup can climb down the tree to catch it.

The package

scrapITRA is a Python package to download races and runners information from the ITRA website.

The package is available at https://github.com/ricfog/ScrapITRA. In the github repo the package can be installed from `.tar.gz` file in the dist folder through the command `pip install ScrapITRA-0.1.0.tar.gz`. After the installation, you can import the package in your Python code for instance with `import scrapITRA as sc`.
Alternatively, you can call the functions from outside the folder, as I show below.

Structure

ScrapITRA/ |-- .gitignore |-- MANIFEST |-- README.md |-- setup.py |-- bin/ | |-- __init__.py | |-- write_races.py |-- dist/ | |-- ScrapITRA-0.1.0.tar.gz |-- scrapITRA/ | |-- __init__.py | |-- src/ | | |-- __init__.py | | |-- get_browser.py | | |-- get_races.py | | |-- get_runner.py | | |-- login.py | | |-- scrape_history.py

Outside the package, I have stored a folder with the WebDrivers, and the login credentials *py file. Therefore as long as ScrapITRA is not a package, it is important to call the functions from outside the package (syntax is `python -m path`).

The structure might have changed since the last revision.

Requirements

Packages: the following packages need to be installed: BeautifulSoup, Selenium.

WebDriver and **Web browser**: the Web Browser identical to the WebDriver used for Selenium (see below) needs to be installed. This means that if you are using Chrome as WebDriver, then Chrome should be installed as well in your system. If you wish to run the driver on some remote server for which you lack root permission, you might find useful the following installation of Firefox https://support.mozilla.org/en-US/kb/install-firefox-linux under "Installing outside of a package manager".

What does this allow you to do?

Two main functionalities:

- input runner information and retrieve historical data (performance and past races)
- retrieve races information (elevation, distance, participants with related infos, ...)

Other functions:

- retrieve list of available nationalities in the dropdown options

A deeper look into the modules

I'll go over two examples that cover all the modules.

Let's start!

First import scrapITRA - here I am importing the package directly from the folder.

In [10]:

```
import ScrapITRA.scrapITRA as sc
```

This will read the __init.py__ first that will tell Python to read also the other nested **init**.py files, ie one inside src. The init files also support the wild import *.

Then we can initialize the browser.

In [11]:

```
browser = sc.get_browser('https://itra.run/community/', False)
```

This reads the function from get_browser.py. This initializes the automated browser with Firefox on the https://itra.run/community/ webpage. The head options allows you to opt for a headless browser or not.

Now you can login in case you are an ITRA member.

In [12]:

```
try:
   from login_credentials import *
   sc.login(username, password, browser)
except ImportError:
    pass
```

This reads the function from login.py. The password is stored in a login_credentials.py file file.

The initialization is similar for both examples. Now let's look at two different functionalities.

Get runner's history and performance

From the get_runner.py file we can obtain the list of runners.

In [13]:

```
runner_list = sc.get_runner(browser, 'Kilian', 'Jornet')
```

This outputs a list of webpages for all runners that match the criterion.

In [14]:

```
runner_list
```

Out[14]:

```
['/community/kilian.jornet burgada/2704/9736/']
```

Now we can iterate through the list.

In [15]:

```
for webpage in runner_list:
    record, performance = sc.scrape_history(webpage, browser)
```

This reads from scrape_history.py. It returns lists for all races (results section) and performance (performance section).

In [16]:

```
performance
```

Out[16]:

```
[['/community/kilian.jornet burgada/2704/9736/',
  '2018-09-15',
  'Salomon Skyline Scotland 2018 - Ring Of Steall Skyrace',
  '26.5km / 2525m+',
  '03:04:34',
  '1 / 672',
  '1',
  '946'],
 ['/community/kilian.jornet burgada/2704/9736/',
  '2017-09-17',
  'Salomon Skyline Scotland 2017 - Glen Coe Skyline',
  '55.1km / 4750m+',
  '06:25:39',
  '1 / 167',
  '1',
  '940'],
 ['/community/kilian.jornet burgada/2704/9736/',
  '2017-09-01',
  'Utmb® 2017',
  '168.6km / 9975m+',
  '19:16:59',
  '2 / 1687',
  '2',
  '932'],
 ['/community/kilian.jornet burgada/2704/9736/',
  '2018-07-29',
  'Skyrun Comapedrosa 2018',
  '21km / 2300m+',
  '02:33:18',
```

```
   '02:33:18',
   '1 / 536',
   '1',
   '930'],
  ['/community/kilian.jornet burgada/2704/9736/',
   '2018-08-12',
   'Sierre-Zinal 2018',
   '30km / 2300m+',
   '02:31:40',
   '1 / 1452',
   '1',
   '926']]
```

Scrape races information

From the get_races.py file we can obtain the list of runners. This module is scraping data from https://itra.run/page/328/Results.html with personalized initial and end dates.

In [17]:

```
races_info = sc.get_races(browser, '20/12/2018', '01/01/2019', path=None)
```

In [22]:

```
races_info[0][:10]
```

Out[22]:

```
[['9-PEAK GLOBAL TRAIL 2018 - 30K',
  'China - Sichuan',
  '27.4km 1500M+',
  '31 December 2018',
  'Rk Name'],
 [],
 ['Shilei XIE', '03:45:22', '1', 'Man', '580', 'China'],
 ['Dong WANG', '03:53:56', '2', 'Man', '559', 'China'],
 ['Yijian LIU', '04:04:08', '3', 'Man', '535', 'China'],
 ['Xiao LU', '04:07:16', '4', 'Man', '529', 'China'],
 ['Ping ZHANG', '04:18:04', '5', 'Man', '506', 'China'],
 ['Bo JIANG', '04:18:13', '6', 'Man', '506', 'Singapore'],
 ['Wei HU', '04:18:28', '7', 'Woman', '506', 'China'],
 ['Ling YUAN', '04:20:21', '8', 'Woman', '502', 'China']]
```

Scripts

This write_races.py file allows you to download races into some txt file from the command line.

from ..src import * import argparse parser = argparse.ArgumentParser() parser.add_argument('--start_date', type=str, default="01/01/2010", help='initial date') parser.add_argument('--end_date', default="01/01/2011", type=str, help='end date') parser.add_argument('--path', default="race", type=str, help='path') args = parser.parse_args() browser = get_browser('https://itra.run/community/', False) try: from login_credentials import * login(username, password, browser) except ImportError: pass races_info = get_races(browser, args.start_date, args.end_date, args.path)

Processing the information

This is some code to process the information previosuly obtained during the scraping process for the races. This is not part of the package.

This assumes that the the files have been written to some .csv documents.

import pandas as pd df = pd.read_csv('races.txt', sep=",", header=None) df = df.drop_duplicates() # create index of races available position_race = [] for row in range(df.shape[0]): if '+' in df.iloc[row][2]: position_race.append(row) # create index for races last_position = df.shape[0] position_race.append(last_position) key_race = [] key = 0 for ind in range(len(position_race)-1): key_race.extend([key]*(position_race[ind+1]-position_race[ind])) key += 1 # create dataframe of races information position_race = position_race[:-1] race_info = df.iloc[position_race] # drop races from the first dataframe df = df.drop(df.index[position_race]) # separate distance & elevation race_info['Distance'], race_info['Elevation'] = race_info[2].str.split(' ', 1).str # delete previous useless columns race_info = race_info.drop([2, 4], axis=1) df = df.drop('level_0', axis = 1) race_info = race_info.drop('index', axis = 1) df = df.drop('index', axis = 1) df.columns = ['Name', 'Time', 'Ranking', 'Sex', 'Nationality', 'key_race'] race_info.columns = ['Race', 'Location', 'Date', 'key_race', 'Distance', 'Elevation'] df.to_csv('race_runner', index=False) race_info.to_csv('race_info', index=False)

Some final useful advices for myself for future implementation

Selenium can be pretty challenging to handle, and most of the time is spent on choosing WHAT you should target (be it a tag, a class, an id), and HOW you should target (css selector, xpath, name). The time.sleep() commands are necessary, as Selenium might perform operations ahead of the browser. For instance, if you input some word in a cell, it takes some time for the automated browser to actually do it; as a consequence, Selenium will already be executing the following lines of code while the page is still uploading. The implicit wait commands in Selenium can help solve some of the trobules, but not all of them.

Important: Xpath differs across browsers, CSS does not. For this reason, as I switched from Chrome to Firefox I found that the CSS implementation of my lookups was more effective.

Important x2: call all the function from outside the ScrapITRA folder with the command python -m ScrapITRA.blablabla.

In [ ]: