How to handle Data present in the Web? (1)

Web Scraping = using data available through regular web pages

- Web scraping is a technique allowing to fetch data from web page
- There is no common "format" that drives how data are contained in web pages
- The web page may change at any time, so it does not represent a stable data source

How to handle Data present in the Web? (2)

Html

- Content displayed from your web browser is contained in `html` (Hyper Text Markup Language) files
- `html` is a markup language, close to xml, but with less constraints
- the browser is highly **fault tolerant**, not many websites would display at all if the browser was strictly enforcing standards

How to handle Data present in the Web? (3)

Html - example

what you have seen in the previous slide is coded this:

```
<section class="slide">
   <h2>How to handle Data present in the Web? (2)</h2>
   <p>Html</p>
   <ul>
      <li>Content displayed from your web browser is contained in <code>html</code> (Hyper Text Markup Language) files</li>
      <li><code>html</code> is a markup language, close to xml, but with less constraints</li>
      <li>the browser is highly <b>fault tolerant</b>, not many websites would display at all if the browser was strictly enforcing stanc
   </ul>
</section>
```

How to handle Data present in the Web? (3)

Difference between DOM and html

- When there's a "fault" in the html file, the browser will do its best to correct it
- What you see is the **rendered DOM** (Document Object Model) and not the raw html anymore
- When a JavaScript action occurs, it works on this DOM object (in memory), not on the html file anymore

This is why there is quite always a difference between what you see in your browser, and what is fetched from the html page (and this is really important

How to handle Data present in the Web? (4)

DOM vs html: example

- DOM can be accessed via the developers tools (on chrome F12 key)
- navigate to : [redacted] (Single Page Application)
- Right click on the page, and select "view source"
- Now go back to the original page, and press F12 to open the developers tools
- compare the raw html (view source) vs the rendered DOM (F12), quite different isn't it?

How to get Data present in the Web?

Exercise: working on a manual extract

- Navigate [redacted]

- Think about the best way to extract the table of Human Development Index (whatever technique is acceptable)
- produce a csv from it as fast as you can, it does not have to be automated! (+1 for the 3 first valid submissions)

How to get Data present in the Web?

Exercise: fetch data from a web page and convert it to an Orange Data Table

- Go to this web page
- use the provided snippet to extract from the xml document a list of Regions (h2 tags)

```python
from lxml import etree
import requests
from bs4 import BeautifulSoup as bs


def getContent(link: str) -> str:
    webPage = requests.get(link)
    return str(bs(webPage.content, "html.parser"))
```

How to get Data present in the Web? (2)

- Now a bit harder task : extract the list of airports with their cities, countries, regions and IATA Code
- The target data table rows will look like this row:

```
['Americas', 'Caribbean', 'Cuba', 'Holguín', 'Frank País Airport', 'HOG']
```

  following this format `[Region, SubRegion, Country, City, Airport name, IATA Code]`

- You will probably need this snippet

```
allNodes: list[etree.ElementBase] = html.xpath(RELATIVE_ROOT+"/*")
length: int = len(allNodes)
#...
```


Limits of direct html fetching

- Fetching the html page is more reliable when it works
- Sometimes javascript is required
- In that case you need a "ghost" browsing, meaning you simulate the interaction between the browser and the website
- an example of this, using selenium, here:

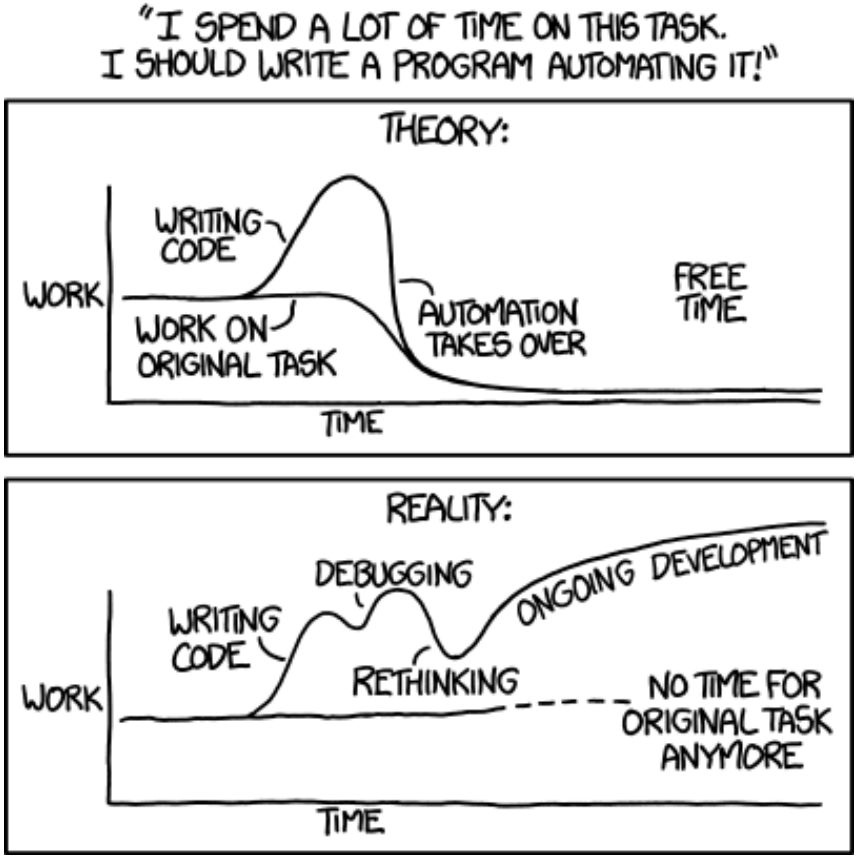
When to use scraping?

Short answer: when you cannot do otherwise

Longer answer: beware of the automation cost, verify that it is worthing it!

- You can use scraping when data volume is too large (to do it manually)
- Limit the effort (when possible) to reference data that will not move too much (web pages can be updated at any time!)

Automation: Theory vs Reality



Other web data you can fetch

REST and XML

curl -i -H "Accept:application/xml" -H "Content-Type:application/xml" -XGET "https://gorest.co.in/public-api/users.xml" GET https://gorest.co.in/public-api/users.xml Accept: application/xml Content-Type: application/xml

aggregating different sources with merge and concatenation

- Merge is working like SQL joins
- Example : with the orange zoo dataset

Exercise: Build an Orange Data table from those two tables (users and posts)

Merging dataframes