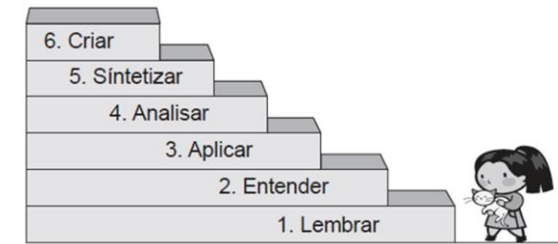


Webscrapy



Fazer ou Pedir: o trade-off da próxima década

- Ler, escrever, praticar, aprender vocabulário (do português ou de uma linguagem de programação)
 - É um exercício de aquisição de habilidades
- Delegar para a IA
 - a leitura, escrita e o domínio da linguagem
 - ✓ É equivalente a ter uma meta de fazer exercício físico
 - E na hora de treinar a corrida você vai de carro
 - Não ganhará o condicionamento incremental para chegar mais longe ou mais rápido
- Usar a IA para aprender ou usar a IA para evitar aprender?
 - Andrej Karpathy cunhou o termo vibe coding



Prompt do Agent mode de ChatGPT com o GPT-5

- Prompt

- faça um código de web scrapy utilizando o selenium em python para fazer o scrapy desta url <https://github.com/orgs/python/people>. Colete a URL, username e foto de cada pessoa da tabela que está nesta página. Crie um código para percorrer as páginas. O código deve ser robusto para receber uma especificação do usuário de quantas páginas deseja coletar (por exemplo, 10 páginas).

- Fez corretamente sem nenhum erro

- É um webscrapy simples,
 - ✓ mas que tomaria uns 10min de uma pessoa experiente

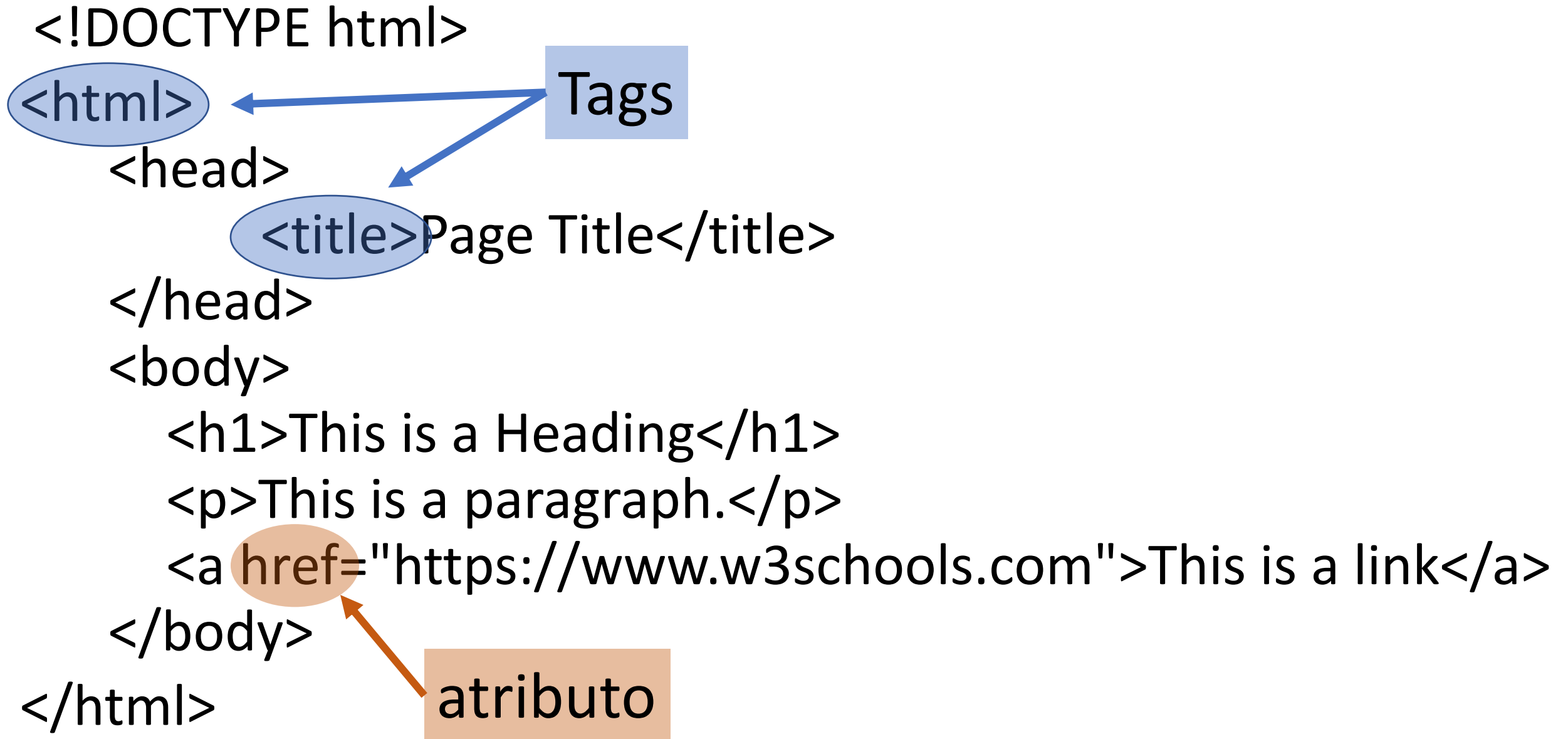
Objetivo da Aula

- Aprender a usar uma IDE Desktop (Vscode / Windsurf)
 - Se tornou um ambiente de prototipação e vibe coding
- Estudo de caso
 - Webscrapy

Webscrapy

- Coleta em larga escala de dados contidos em páginas web
- Legalidade
 - tem sido bastante discutida nos EUA no caso [HiQ Labs vs LinkedIn](#)
 - ✓ A decisão final foi em favor do LinkedIn
- Tenha
 - Bom senso, cuidado e responsabilidade
 - ✓ "With great power, comes great responsibility" (Uncle Ben/Stark)
 - Pequenos loops, limitado a uma quantidade que valide seu experimento.

Introdução à HTML



Introdução à HTML

- Atributos mais comuns no webscrapy
 - **id**
 - **class**
- Tags mais comuns no webscrapy
 - **div** – Define uma divisão ou uma seção num documento HTML
 - **a** – Define um hyperlink
 - **span** – É um container in-line usado para destacar parte de um texto.
 - **ul** e **li** – Definem uma lista não ordenada
 - **table**, **td** e **tr** – Definem uma tabela, as colunas e as linhas

XPATH (XML Path Language)

- Define sintaxe de "caminho" para identificar e navegar
 - em nós em um documento XML
 - ✓ Semelhante a caminhos numa estrutura de pastas de computador

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd country="USA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

A que se refere os
caminhos XPATH ?
`/catalog/cd/price`

XPATH – Seleção de Nós

- Sintaxe para criar uma expressão XPATH

Expressão	Descrição
começar com /	Seleciona o nó raiz
começar com //	Seleciona os nós no document do nó que satisfaz o critério de seleção, não importando onde ele esteja na árvore
.	Seleciona o nó atual
..	Seleciona o pai do nó atual
@	Seleciona os atributos

Exemplos de Seleção de Nós

- /catalog/cd/price
- //cd
- //@country
- //cd[@country='UK']
- //cd[@country='UK']/price
- //cd[@country='UK']/price/..

```
<?xml version="1.0" encoding="ISO-8859-1"
<catalog>
  <cd country="USA CA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

XPATH – Predicados

- Predicados são colocados entre colchetes e filtram valores

Exemplo	Descrição
<code>//cd[1]</code>	[1] seleciona o primeiro nó que satisfaz o critério
<code>//cd[last()]</code>	[last()] seleciona o último nó que satisfaz o critério
<code>//cd[@country]</code>	[@country] seleciona todos os nós cd que contêm o atributo country
<code>//cd[@country='UK']</code>	<code>//cd [@country='en']</code> seleciona todos os nós cd cujo atributo country é igual a 'UK'
<code>//cd[contains(@country,'USA')]</code>	<code>//cd[contains(@country,'USA')]</code> seleciona todos os nós cd cujo atributo country contém USA

Selenium

- Selenium automates browsers. That's it!
 - What you do with that power is entirely up to you.
- Selenium automatiza os Navegadores
 - O que você faz com esse poder é por sua conta.
- Qualquer interação do usuário com o Navegador
 - pode ser automatizada pelo Selenium. Exemplos:
 - ✓ Cliques;
 - ✓ Digitar texto;
 - ✓ Minimizar/Fechar janela;
 - ✓ Mouse over (passar o mouse sobre um elemento html)

Métodos do Selenium para retornar Elementos HTML

- `find_element(By.ID, "id")`
- `find_element(By.NAME, "name")`
- **`find_element(By.XPATH, "xpath")`**
 - Domine esse
- `find_element(By.LINK_TEXT, "link text")`
- `find_element(By.PARTIAL_LINK_TEXT, "partial link text")`
- `find_element(By.TAG_NAME, "tag name")`
- `find_element(By.CLASS_NAME, "class name")`
- `find_element(By.CSS_SELECTOR, "css selector")`
 - Retorna multiplos elementos numa lista

Webscrapy com Selenium

- Exemplo Básico


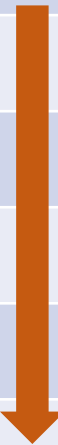
- Coletar dados de uma organização no github
 - ✓ <https://github.com/python>
- Para cada repositório, vamos coletar:
 - ✓ Nome;
 - ✓ URL;
 - ✓ Forks; e
 - ✓ Estrelas.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(ChromeDriverManager().install())
url = 'http://books.toscrape.com/'
driver.get(url)
print(driver)
```

Pequenas iterações e incrementos

- Menor
 - Risco, retrabalho e desperdício
- 6 entradas (equivalentes) e 4 etapas. O que fazer primeiro?
 - No caso de entradas equivalentes

	Etapa 1	Etapa 2	Etapa 3	Etapa 4
Entrada 1				
Entrada 2				
Entrada 3				
Entrada 4				
Entrada 5				
Entrada 6				

Webscrapy com Selenium

- Comece simples e evolua iterativamente e interativamente
 - Exemplo: scrapy dos livros de <http://books.toscrape.com/>
- Execute esse trecho de código
 - Coloque um breakpoint na linha do print,
 - Inspecione o código HTML da página alvo (Debugger do Navegador)
 - Teste suas expressões XPATH com a ajuda da função Watches do Windsurf

```
driver = webdriver.Chrome(ChromeDriverManager().install())  
url = 'http://books.toscrape.com/'  
driver.get(url)  
print(driver)
```

- Dependendo do sistema operacional, você precisará especificar
 - o path do driver do Firefox com o argumento `executable_path`

Estratégia de Desenvolvimento

- Primeiro colete os dados de uma página de resultados
 - Depois desenvolva uma estratégia para buscar várias páginas
 - ✓ Por exemplo, clicando em "Próximo" ou nos links das próximas páginas
- Tente não executar a função *sleep* no seu código de scrapy
 - quando executado em larga escala, um prazo de sleep nunca será ao mesmo tempo
 - ✓ pequeno o suficiente para você coletar os seus dados de forma rápida, e
 - ✓ grande o suficiente para evitar buscar um dado ainda não carregado na página
 - Em vez disso use a função `WebDriverWait`
 - ✓ Ou a função disponibilizada pelo professor, que usa a `WebDriverWait`

Estratégia de Desenvolvimento (2)

- Expressões XPATH simples (que dependam de poucos nós)
 - Deixarão seu código eficaz por mais tempo
- Planeje antes de fazer!!!
 - "Dividir para conquistar"
 - Quais dados vai coletar?
 - Como vai fazer a iteração sobre as páginas
- Salve a evolução do scrape
 - Salve/imprima o número das páginas
 - ✓ que já foram coletadas com sucesso
- Dê preferência à tarefas de remodelagem e organização dos seus dados
 - à atividades de web scraping
 - ✓ O esforço se justifica no longo prazo

Qual expressão XPATH usar?

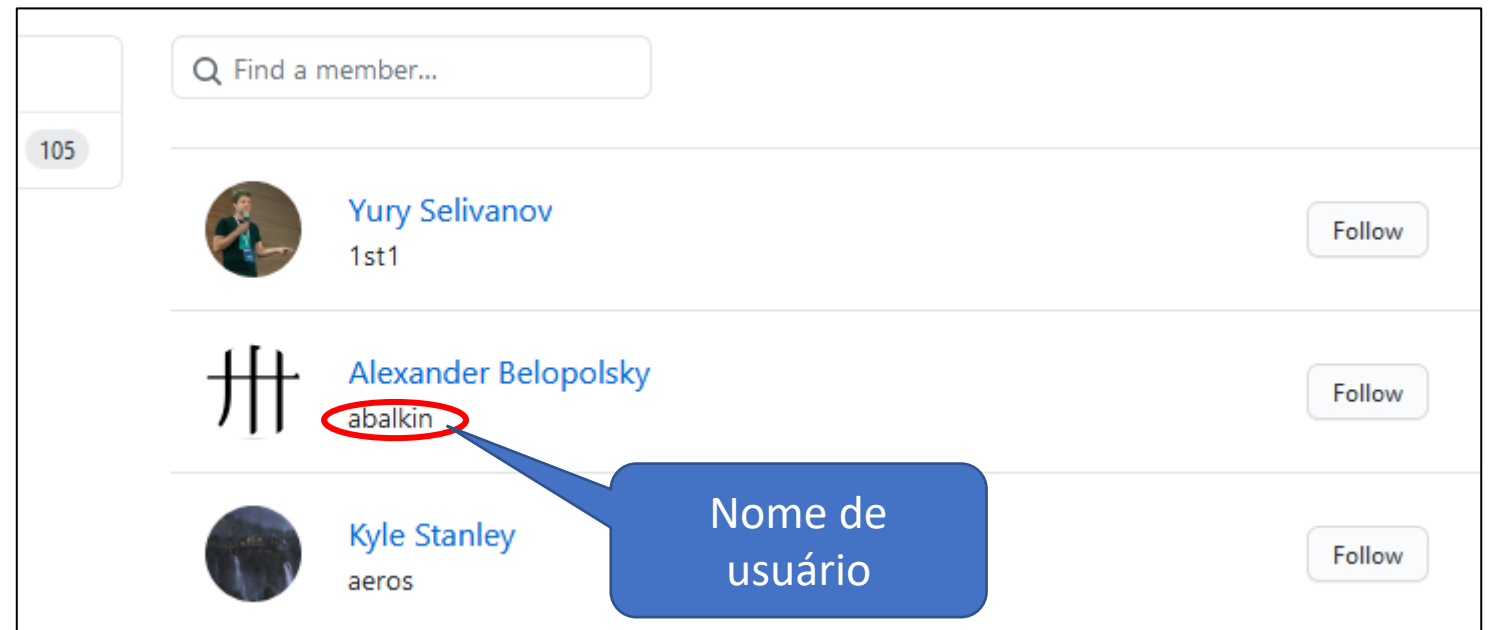
- Teste interativamente

```
▼ <section>
  ▶ <div class="alert alert-warning" role="alert"> ... </div>
  ▼ <div>
    ▼ <ol class="row">
      ::before
      ▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
        ::marker
        ▼ <article class="product_pod">
          ▶ <div class="image_container"> ... </div>
          ▶ <p class="star-rating Three"> ... </p>
          ▼ <h3>
            <a href="catalogue/a-light-in-the-attic_1000/index.html" title="A Li
            </h3>
            ▶ <div class="product_price"> ... </div>
          </article>
        </li>
        ▶ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3"> ... </li>
```

```
//a
//h3/a
//article/h3/a
//article[@class='product_pod']/h3/a
//ol/li/article/h3/a
```

Webscrapy com Selenium

- Atividade para praticar
 - Coletar dados das pessoas associadas a organização Python no github
 - ✓ <https://github.com/orgs/python/people>
 - Para cada pessoa, colete:
 - ✓ Nome;
 - ✓ Nome de usuário; e
 - ✓ URL.



Execução do Warmup

Até o W8.1

Tempo: 10 min

00 : 10 : 00

Duration: 00 10 00

TimeUp Reminder (Optional): -- -- --

Choose Sound Effect None

Choose TimeUp Sound Alarm

☐ Enable Count Up ☐ Combine With Bar Clock

Start

Pause

Stop

Reset

De volta ao exemplo books.toscrape.com

- Criar uma iteração sobre elementos de interesse resultantes de
 - `find_elements_by_xpath`
 - ✓ são tags a, clique nelas com o método `click()`
 - isso faz o driver do navegador abrir esse link (entrar nessa página)
- Usar a função `wait_element` disponibilizada pelo Professor
- Testar interativamente
 - Colocar um breakpoint na linha do método `wait_element`
 - ✓ E testar novas expressões XPATH para selecionar o conteúdo de interesse

```
a_tags = driver.find_element(By.XPATH, '//article/h3/a')
for a in a_tags:
    a.click()
    wait_element(driver, '//tr/td', by=By.XPATH)
```

Encontre e colete os campos de interesse (em cada iteração)

```
href = a.get_attribute('href')
current_page_driver.get(href)
wait_element(current_page_driver, '//tr/td', by=By.XPATH)
product_main = current_page_driver.find_element(By.XPATH,
//div[contains(@class,"product_main")]')
product_main_text = product_main.text.split('\n')
title = product_main_text[0]
price = product_main_text[1]
stock = re.findall('\d+', product_main_text[2])
product_main_ps = product_main.find_elements_by_tag_name('p')
stars_colors = [x.value_of_css_property("color") for x in
product_main_ps[2].find_elements_by_tag_name('i')]
stars = stars_colors.count('rgb(230, 206, 49)')
description = current_page_driver.find_element_by_xpath('//article/p').text
```

Transforme esse código numa função

```
def get_page_data(driver):
    a_tags = driver.find_elements_by_xpath('//article/h3/a')
    dataset = []
    current_page_driver = webdriver.Firefox()
    for a in a_tags:
        href = a.get_attribute('href')
        current_page_driver.get(href)
        wait_element(current_page_driver, '//tr/td', by=By.XPATH)
        ...
        record = {'title': title, 'price': price, 'stars': stars}
        dataset.append(record)
    current_page_driver.close()
    return dataset
```


Faça uma iteração sobre as páginas

```
def scrapy_books(url):
    driver = webdriver.Firefox()
    driver.get(url)
    to_continue = True
    whole_dataset = []
    while to_continue:
        current_items = get_page_data(driver)
        whole_dataset.extend(current_items)
        try:
            nb = driver.find_element_by_xpath('//li[@class="next"]/a')
        except NoSuchElementException:
            pass
            break
        nb.click()
        wait_element(driver, '//img[@class="thumbnail"]', by=By.XPATH)
    driver.close()
    return whole_dataset
```

Execute a função e imprima o resultado

- Código fonte completo

- https://github.com/alexlopespereira/curso_ciencia_dados2021/blob/master/src/scrapy/books_toscrapy.py

```
url = 'http://books.toscrape.com'
result = scrapy_books(url)
with open('data.json', 'w') as fp:
    json.dump(result, fp)
print(result)
```

Fake User Agent

- User Agent
 - Uma string gerada pelo navegador e entregue ao servidor web
 - ✓ Identifica seu browser e seu sistema operacional
- Usado por alguns sites para identificar bots
- Solução
 - Utilizar "user agents" gerados aleatoriamente

```
useragent = UserAgent()  
profile = webdriver.FirefoxProfile()  
profile.set_preference("general.useragent.override", useragent.random)  
firefox_driver = webdriver.Firefox(firefox_profile=profile)
```

- Outras soluções em
 - <https://github.com/elvesrodrigues/antiblock-scrapy-selenium>

Prática no Jupyter Notebook

- Faça os exercícios da aula.

Exercício 4.1

- Escolha uma das duas opções descritas nos proximos slides
 - a) quotes.toscrape.com
 - b) Webscrapy do Google

Exercício 4.1 a) Webscrapy do quotes.toscrape.com

- Faça um webscrapy dos dados da pagina <https://quotes.toscrape.com/>
 - Colete a citação (quote), o autor e as tags
- O resultado deve ser uma lista de dicionário
- Seu código deve necessariamente utilizar as boas práticas de navegação nas páginas ensinadas na aula
- **Faça no ambiente do seu PC numa IDE!!!**
- Submeta [aqui](#) uma url de um arquivo com o seu código fonte no formato de um arquivo texto
 - E um link de um vídeo explicando o webscrapy que você fez. Pode usar a IA desde que você saiba explicar o que ela fez.

Exercício 4.1 b) Webscrapy do Google

- Busque por:
 - gov.br
- Para cada resultado no primeiro nível, colete:
 - Título do resultado;
 - URL do resultado.
- Gere uma lista de dicionários como resultado.
 - [{"titulo": " Governo Federal - Governo do Brasil... — Português (Brasil)", "url": <https://www.google.com/url?sa=t&rct=..., ...>}]
- Colete os dados apenas da 1a página de resultados.
 - **Faça no ambiente do seu PC numa IDE!!!**
- Submeta [aqui](#) uma url para o seu código fonte no formato de um arquivo texto
- Submeta um link de um vídeo explicando o webscrapy que você fez