

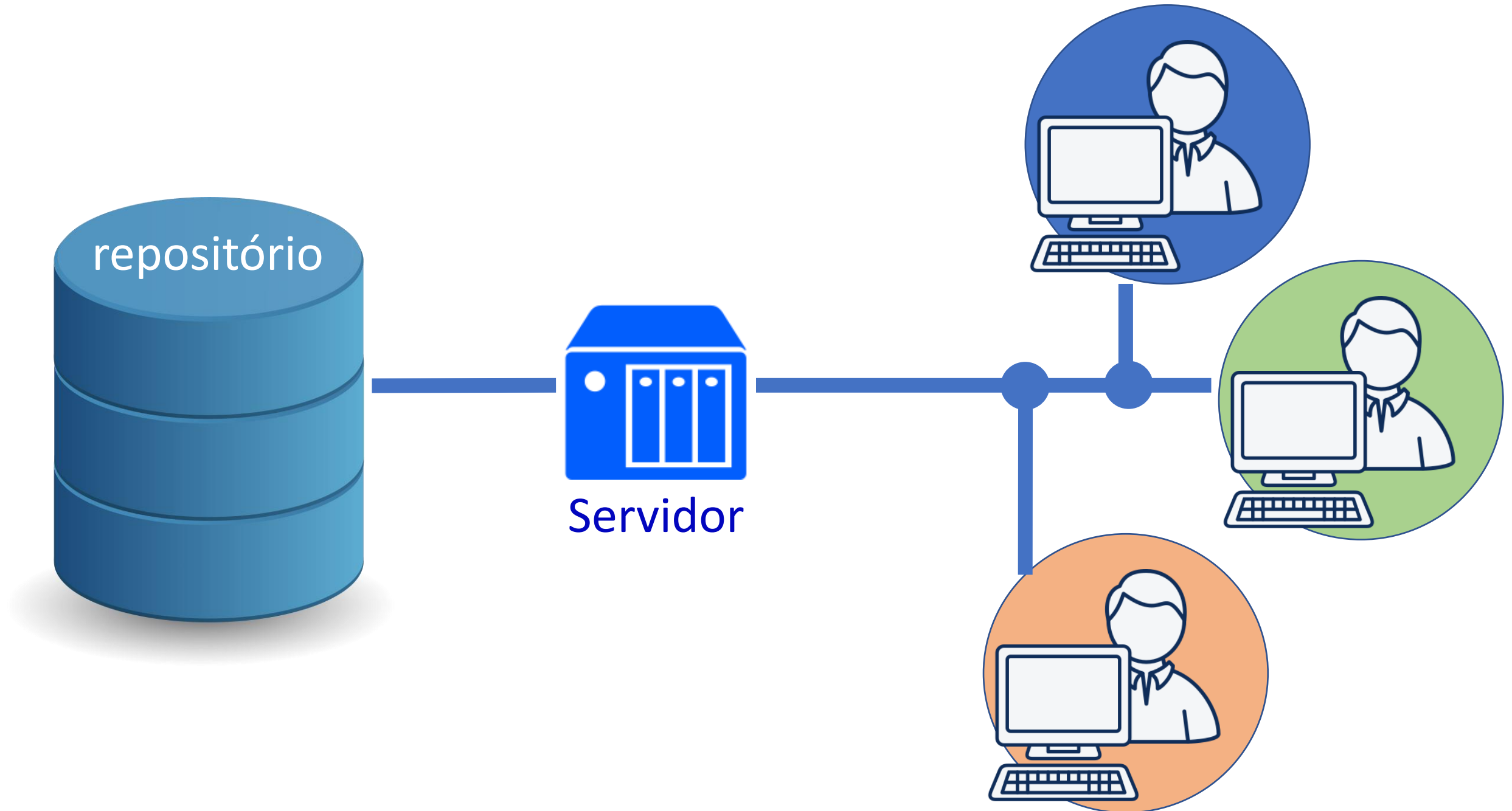
Git, Ambiente de Desenvolvimento e Webscrapy



Git

- É um sistema de controle de versão
 - Utilizado para trabalho colaborativo,
 - ✓ Melhor ferramenta de gestão de desenvolvimento colaborativo.
 - Compartilhamento de código, dados e modelos.
 - É uma ferramenta de gestão de configuração de software.
 - Guarda o histórico de todas as alterações (arquivos, itens de config.)
- Desenvolvido por Linus Torvalds
- Download (Windows)
 - <https://gitforwindows.org/>
- Instalação
 - Windows: as opções padrão são apropriadas e suficientes
 - Linux (Ubuntu): `sudo apt install git`

Sistema de Controle de Versão



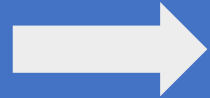
Principais funções e comandos do Git

- Cria um repositório
 - `git init <DIRETORIO>` (usaremos o github.com para criar)
 - ✓ este comando não será usado
- Clonar um repositório (fazer o download)
 - `git clone <ENDERECO_DO_REPOSITORIO>`
 - ✓ `git clone https://github.com/alexlopespereira/curso_ciencia_dados2020.git`
- Adicionar (marcar como adicionado) um arquivo ao repositório
 - `git add <ARQUIVO>`
- Confirmar a adição do arquivo
 - `git commit -m "Adicionei o arquivo XXX"`
- Enviar para o repositório
 - `git push origin master` (enviar para a branch master)
- Fazer o download de novas atualizações do repositório
 - `git pull`

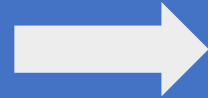
Principais fluxos no Git (Caminho feliz)

Fluxo de Criação e Envio de Arquivos

Cria um
arquivo



Adiciona
git add



Confirma
git commit



Envia
git push

Fluxo de Alteração e Envio de Atualizações

Altera um
arquivo



Confirma
git commit



Envia
git push

Fluxo de Download de Atualizações

Download
git pull

Exercício de Prática com git

- Criar um repositório público no github.com
- Clonar um repositório (fazer o download)
 - `git clone <ENDERECO_DO_REPOSITORIO>`
- Adicionar um arquivo ao repositório
 - `git add <ARQUIVO>`
- Confirmar a adição do arquivo
 - `git commit -m "Adicionei o arquivo XXX"`
- Enviar para o repositório
 - `git push origin master` (enviar para a branch master)
- Alterar um arquivo pelo website do github.com
- Fazer o download de novas atualizações do repositório
 - `git pull`

Configuração de Ambiente Python no Windows

- Download e instalação do Anaconda
 - <https://docs.anaconda.com/anaconda/install/windows/>
 - https://repo.anaconda.com/archive/Anaconda3-2020.07-Windows-x86_64.exe
- Download e instalação do Pycharm (Versão Community)
 - <https://www.jetbrains.com/pt-br/pycharm/download/#section=windows>

Ambiente Virtual (Virtual Environment)

- Instalador de pacotes – pip/conda (existem outros)
 - `conda/pip install <PACOTE>`
 - ✓ `pip install pandas`
 - ✓ `conda install pandas`
- Ambiente virtual
 - **é um ambiente isolado de pacotes python**
 - ✓ Elimina problema de diferentes projetos precisarem de várias versões do mesmo pacote
 - ✓ Minimiza risco de provocar instabilidade nas ferramentas baseadas em python utilizadas em outros contextos

Configuração do Ambiente Virtual com pip

- Instalar o pacote virtualenv
 - ✓ `pip install virtualenv`
- Criar e entrar um diretório
 - `mkdir virtual_envs && cd virtual_envs`
- Criar um ambiente virtual chamado env
 - `python -m venv env`
- Executar o script activate pra habilitar o ambiente env
 - `env\Scripts\activate`
 - ✓ a partir deste momento qualquer pacote será instalado no ambiente virtual
 - enquanto o ambiente estiver ativado: (env) no início do prompt
 - `env\Scripts\deactivate.bat`
- Instalar um pacote de teste (`pip install etlelk`)

Criar projeto no Pycharm

- File -> New Project
 - Crie um ambiente virtual
 - ✓ o pycharm faz isso pra você
- Crie um arquivo main.py
 - escreva um código de teste. Uma soma e um print.
- Crie um breakpoint e execute o arquivo em modo debug
 - Inspecione as variáveis
 - adicione uma expressão aos Watches
 - Avance uma linha com o botão step over
 - Aprenda a usar os outros botões da barra de debug
- Teste o console python

Mais funções do Pycharm

- Teste algumas opções do menu Code
- Teste a função Rename do menu Refactor
- Teste a funcionalidade de busca e de substituição
 - Find In Path (CTRL+SHIFT+F)
- Crie um diretório e um arquivo (teste.py) dentro dele
 - Adicione o diretório aos Sources Root (PYTHONPATH)
 - Crie uma função square(x) deste novo arquivo
 - Use esta função dentro do arquivo main.py
 - ✓ Use a ajuda do pycharm para encontrar e importar square

Git no Pycharm


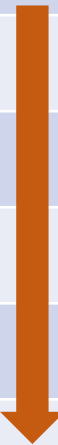
- VCS -> Get from Version Control
 - No campo URL preecha a URL de um repositório git
 - ✓ pode ser algum repositório do seu github
 - Isso vai fazer o clone do seu repositório
- Teste os comandos git: add, commit, push e pull
 - pela linha de comando,
 - e pelo menu de contexto do pycharm.
- Procure pacotes python pela interface do Pycharm
- Crie um arquivo requirements.txt e adicione-o ao repositório
 - `pip freeze > ./requirements.txt`
 - ✓ `pip install -r requirements.txt`

Webscrapy

- Coleta em larga escala de dados contidos em páginas web
- Legalidade
 - tem sido bastante discutida nos EUA
 - ✓ <https://newmedialaw.proskauer.com/2020/03/11/linkedin-files-petition-to-the-supreme-court-in-hiq-web-scraping-case/>
 - No caso:
 - ✓ hiQ Labs, Inc. vs. LinkedIn Corp.
 - A primeira decisão da suprema corte americana foi
 - ✓ Não criminalizar o acesso a dados não protegidos por login e senha
- Tenha
 - Bom senso, cuidado e responsabilidade
 - ✓ "With great power, comes great responsibility" (Uncle Ben)
 - Pequenos loops, limitado a uma quantidade que valide seu experimento.

Pequenas iterações e incrementos

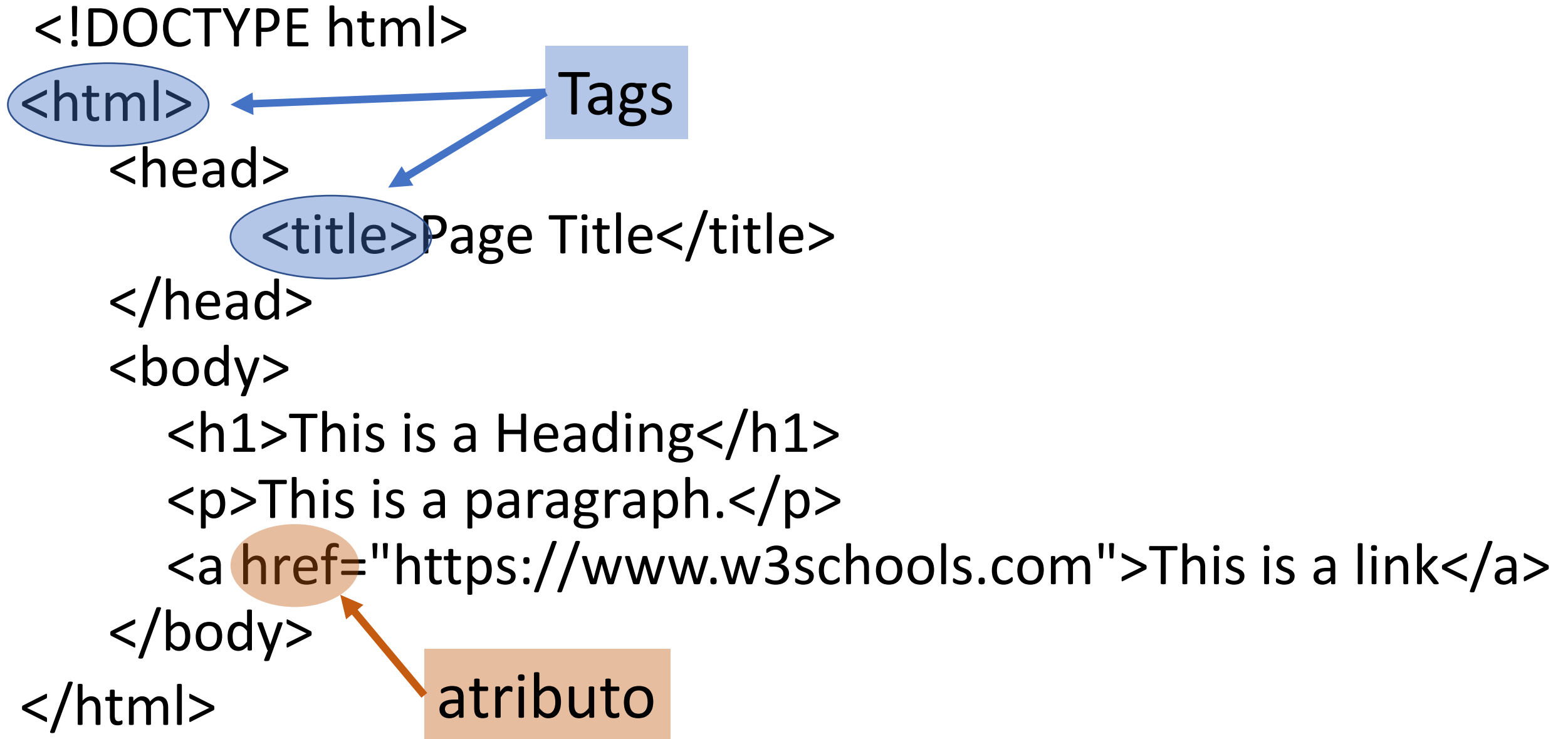
- Menor
 - Risco, retrabalho e desperdício
- 6 entradas e 4 etapas. O que fazer primeiro?
 - No caso de entradas equivalentes

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

Introdução à HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>This is a Heading</h1>
    <p>This is a paragraph.</p>
    <a href="https://www.w3schools.com">This is a link</a>
  </body>
</html>
```

Introdução à HTML



Introdução à HTML

- Atributos mais comuns
 - **id**
 - **class**
- Tags mais comuns
 - **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

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 pycharm

```
driver = webdriver.Firefox()  
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`

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 scrapy
 - Salve/imprima o número das páginas
 - ✓ que já foram coletadas com sucesso
- Dê preferencia à tarefas de remodelagem e organização dos seus dados
 - à atividades de webscrapy
 - ✓ O esforço se justifica no longo prazo

Métodos do Selenium para retornar Elementos HTML

- `find_element_by_id`
- `find_element_by_name`
- **`find_element_by_xpath`** (domine o uso desse método)
- `find_element_by_link_text`
- `find_element_by_partial_link_text`
- `find_element_by_tag_name`
- `find_element_by_class_name`
- `find_element_by_css_selector`
- `find_elements_*` (os mesmo acima)
 - Retorna multiplos elementos numa lista

Manual: <https://selenium-python.readthedocs.io/locating-elements.html>

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
```

Crie uma iteração sobre elementos de interesse

- Crie 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)
- Use a função `wait_element` disponibilizada pelo Professor
- Teste interativamente
 - Coloque um breakpoint na linha do método `wait_element`
 - ✓ E teste novas expressões XPATH para selecionar o conteúdo de interesse

```
a_tags = driver.find_elements_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

```
product_main =  
current_page_driver.find_element_by_xpath('//div[contains(@class,"product_m  
ain")]')  
title = product_main.find_element_by_tag_name('h1').text  
product_main_ps = product_main.find_elements_by_tag_name('p')  
price = product_main_ps[0].text  
stock = re.findall('\d+', product_main_ps[1].text)  
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

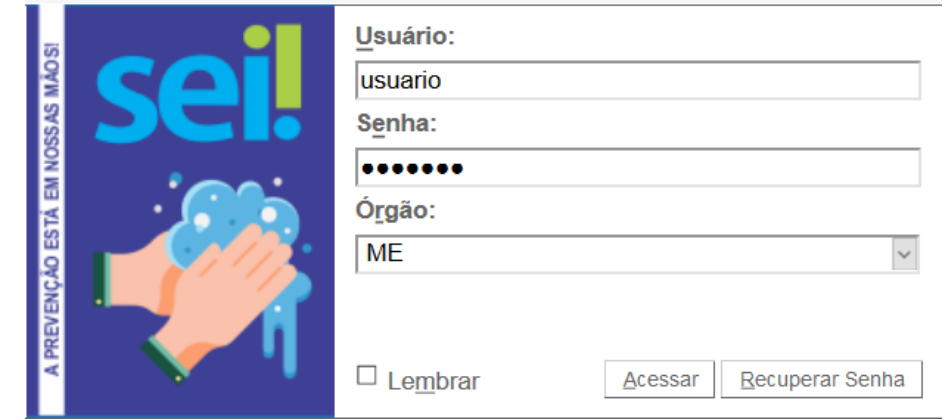
```
def scrapy_books(url):
    driver = webdriver.Firefox()
    driver.get(url)
    to_continue = True
    whole_dataset = []
    while to_continue:
        current_items = get_page_data(driver)
        whole_dataset.extend(current_items)
        try:
            nb = driver.find_element_by_xpath('//li[@class="next"]/a')
        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

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

Preencher campos de formulário com Selenium

- Encontre o elemento input
 - e use o método send_Keys(str)



The screenshot shows a login interface with the following elements:

- Usuário:** A text input field containing the text 'usuario'.
- Senha:** A password input field with masked characters (dots).
- Órgão:** A dropdown menu with 'ME' selected.
- Lembrar:** A checkbox.
- Acessar:** A button to submit the login form.
- Recuperar Senha:** A button to recover the password.

```
url = 'https://sei.fazenda.gov.br/'
driver = webdriver.Firefox()
driver.get(url)
ilogin = driver.find_element_by_xpath('//input[@id="txtUsuario"]')
ilogin.send_keys('login')
isenha = driver.find_element_by_xpath('//input[@id="pwdSenha"]')
isenha.send_keys('password')
option = driver.find_element_by_xpath('//option[@value="0"]')
option.click()
ba = driver.find_element_by_xpath('//button[@id="sbmLogin"]')
ba.click()
```


Trabalho Final

1. Estabeleça um objetivo de análise de dados.
2. Escolha uma ou mais bases de dados do seu interesse.
3. Faça análises estatísticas sobre esta base de dados e apresente argumentos, achados ou conclusões não triviais baseados nos parâmetros calculados por você.
4. O formato deve ser um Jupyter notebook contando uma história com as seguintes seções: Apresentação da Base de Dados, Motivação, Investigação (Com código fonte), e Principais achados ou conclusões.
5. Trabalhos utilizando uma base de dados entre 2 mil e 100 mil linhas, valem no máximo 8. Entre 100 mil e 500 mil linhas, 9. E mais do que 500 mil, 10.
6. A base de dados deve ser pública ou acessível ao Professor.
7. Prazo de Entrega: 06/09/2020 até as 23h59 no horário de Brasília. Atrasos terão descontos exponenciais a cada dia.

Entrega dos Exercícios

- Os exercícios poderão ser submetidos na plataforma do Jupyter hub até o dia **13/09/2020** às 23h59 no horário de Brasília.
- Depois desta data o ambiente do Jupyter Hub será reiniciado
 - Sem dados de exercícios submetidos.
 - ✓ **Salve suas soluções antes desta data!**
- O ambiente do classroom continuará disponível
 - E a solução dos exercícios será disponibilizada lá
- Nota Final
 - $(\text{Nota Exercicios} * 0.7 + \text{Nota Trabalho Final} * 0.3)$
- Nota de Aprovação
 - 70%

Prática no Jupyter Notebook

- Faça os exercícios da aula.