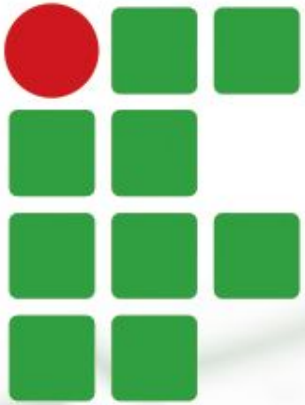

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Sistemas Distribuídos

- Requests && Scraping -

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Scraping

- ***Scraping*** ou **Raspagem de Dados** é uma técnica de mineração que possibilita a extração de dados de sites na WEB, convertendo-os em informação útil e estruturada para fins diversos.

Scraping

- ***Por exemplo***...
- Resgatar as últimas notícias de diversos portais diferentes, automatizando o acesso um a um, e reunindo as informações em um único local.
- Reunir comentários de usuários sobre um determinado produto com base em diferentes *e-commerces*.
- Realizar o *download* de mídias (imagens/áudio, etc) de forma automatizada por meio de *script*.

Obtenção da Página

- Primeiro Passo: **Obtenção da Página**

- Alternativas no Python:
  - Módulo **`urllib`**
    - Módulo Nativo
  - Módulo **`Requests`**
    - Módulo Externo
    - Mais limpo, enxuto e moderno

      > Correção de Tags HTML

Requests

- ***Requests*** é uma biblioteca Python que permite obter páginas WEB a partir de métodos como **GET**.

- Instalação do Pacote:
  - `pip install requests`

Vamos ao nosso 'Hello World' de WebScraping

Requests

- ***Requests*** é uma biblioteca Python que permite obter páginas WEB a partir de métodos como **GET**.

```
from requests import get

pagina = get('http://www.pudim.com.br/')

print(pagina.text)
```

Vamos ao nosso 'Hello World' de WebScraping

Scraping

- ***Beautiful Soap*** é uma biblioteca Python que permite a análise e raspagem de dados provenientes de documentos HTML ou XML.

- Instalação do Pacote:
  - `pip install bs4`

Scraping

- ***Beautiful Soap*** é uma biblioteca Python que permite a análise e raspagem de dados

```
from requests import get
from bs4 import BeautifulSoup

pagina = get('http://www.pudim.com.br/')

bs = BeautifulSoup(pagina.text,'html.parser')

print(bs.title)
```

Principais Métodos

- **`find` *ou* `find_all('tag',{'':''})`**
  - Retorna um item (find) ou uma lista de itens (find_all), contendo o(s) objeto(s) da **tag**.
  - Opcionalmente, o segundo parâmetro do método pode filtrar atributos da tag, como: name, id ou class
- **`select_one` *ou* `select('seletorCSS')`**
  - Retorna um (select_one) ou todos (select) os objetos com seletor ***CSS-like***
- **`get('atributo')`**
  - Obtem o valor de um **atributo** HTML
- **`text()`**
  - Retorna o conteúdo **textual** de **um objeto** selecionado.

```
from requests import get
from bs4 import BeautifulSoup
url = 'https://www.imdb.com/chart/top/'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)\
    Chrome/111.0.0.0 Safari/537.36'}
pagina = requests.get(url,headers=headers)
bs = BeautifulSoup(pagina.text, 'html.parser')
linhas = bs.find_all('td',{'class':'titleColumn'})
filmes = []
for l in linhas:
    nome = l.find('a').text
    filmes.append(nome)
print('TOP 250 Filmes')
for f in filmes:
    print(f)
```

Laboratório Prático

- *Vamos capturar Pokémons?*
- Utilize o repositório abaixo como fonte do scrap:
  - https://pokemondb.net/pokedex/shiny
- Implemente um *script* para fazer o *download* da imagem representativa de todos os pokémons listados e salvar em um diretório local.
- Implemente um pool de processos para paralelizar a tarefa.
- Meça o tempo de processamento.