

## Relatório <008> - <Prática: Web Scraping com Python p/ Ciência de Dados (II)>

<Kawan Machado>

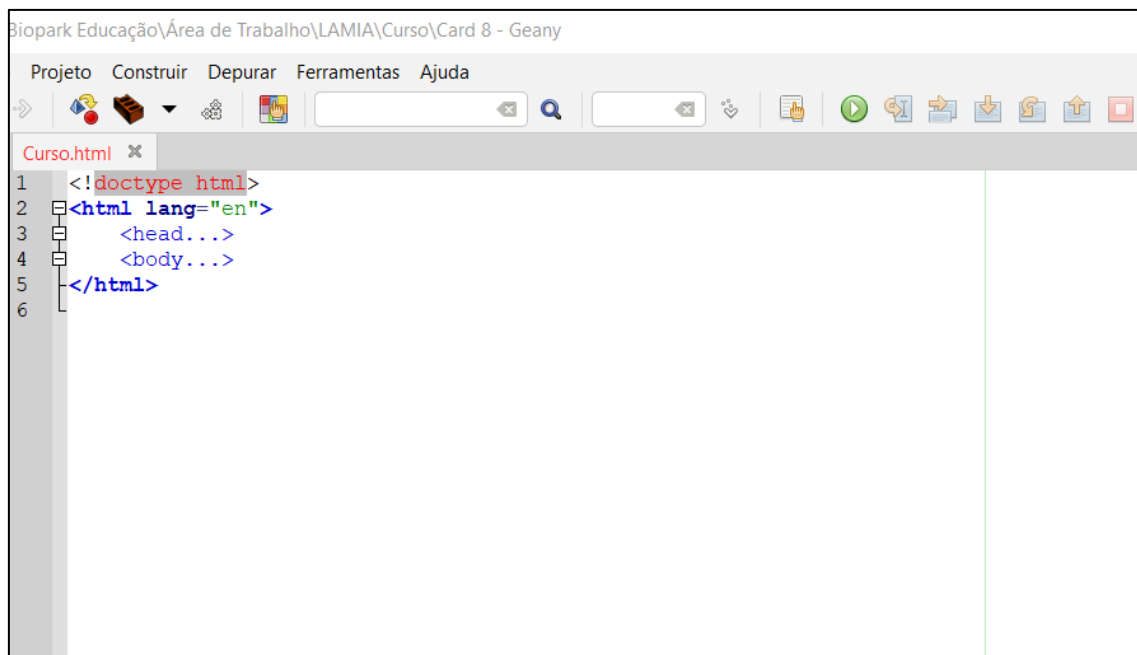
Descrição da atividade

### 1. Local HTML Scraping

#### 1.1 Basic HTML Structure, HTML Tags Explanation

A estrutura padrão do HTML é primeiro definindo o tipo do documento, através do código ‘<!doctype html>’. Depois ele se subdivide, primeiro definindo onde estará contido o código html e sua língua (Português, Inglês, etc).

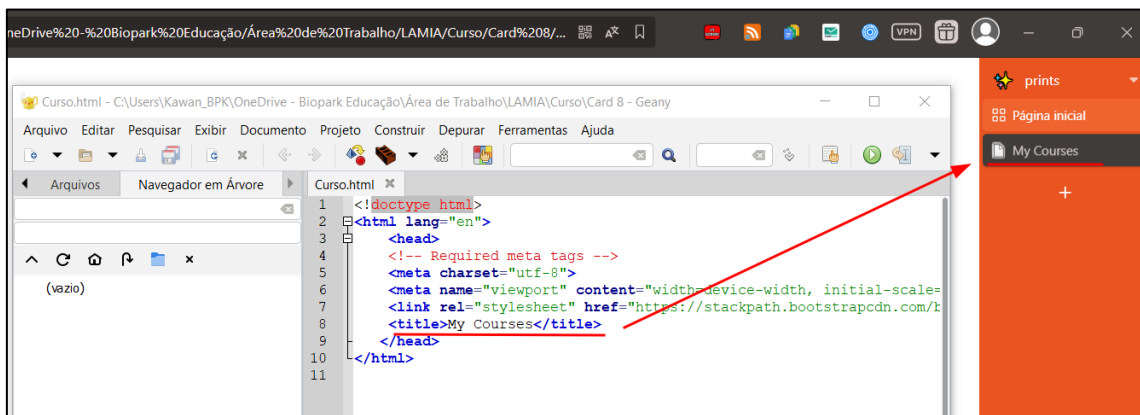
IMAGEM 1 – ESTRUTURA BASE



Elaborado pelo autor, 2025.

Como pode ser notado, o html funciona fazendo uso de tags, elas podem estar abrindo ‘<>’ e fechando ‘</>’. Tudo que está dentro daquele conjunto de tags irá executar naquele campo específico da página, por exemplo, a tag <tittle...> refere-se ao título que aparece em uma aba do site.

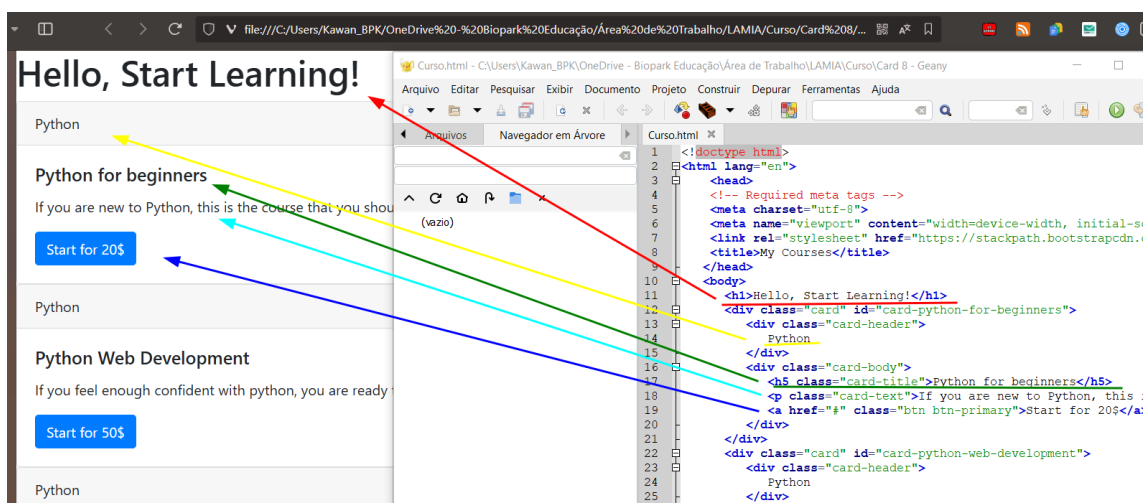
## IMAGEM 2 – FUNCIONAMENTO DE TAGS



Elaborado pelo autor, 2025.

A tag ‘<body>’ é responsável por mostrar tudo que é visível no site, dentro do ‘body’ encontramos as tags ‘<div>’, elas basicamente vão definir a forma do site através do atributo ‘class’. Dentro da ‘div’ podemos destacar a tag ‘<a>’ que permite linkar outras páginas ou seja, ela é basicamente uma referência a outra página.

## IMAGEM 3 – BODY E DIV

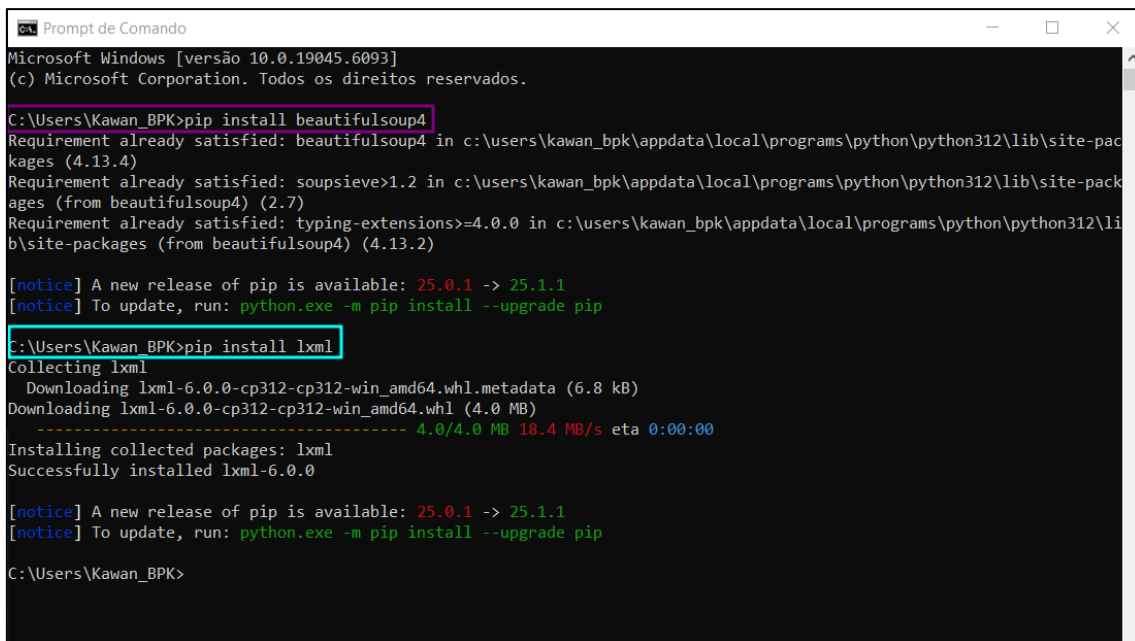


Elaborado pelo autor, 2025.

### 1.2 Packages Installation

Para realizar um bom scrap, precisamos instalar duas bibliotecas para isso, a primeira é o BeautifulSoup4 e a segunda é o lxml. Após a instalação, estamos prontos para iniciar um scrap.

## IMAGEM 4 – INSTALAÇÃO DE BIBLIOTECAS



```
Prompt de Comando
Microsoft Windows [versão 10.0.19045.6093]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Kawan_BPK>pip install beautifulsoup4
Requirement already satisfied: beautifulsoup4 in c:\users\kawan_bpk\appdata\local\programs\python\python312\lib\site-packages (4.13.4)
Requirement already satisfied: soupsieve>1.2 in c:\users\kawan_bpk\appdata\local\programs\python\python312\lib\site-packages (from beautifulsoup4) (2.7)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\kawan_bpk\appdata\local\programs\python\python312\lib\site-packages (from beautifulsoup4) (4.13.2)

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Kawan_BPK>pip install lxml
Collecting lxml
  Downloading lxml-6.0.0-cp312-cp312-win_amd64.whl.metadata (6.8 kB)
  Downloading lxml-6.0.0-cp312-cp312-win_amd64.whl (4.0 MB)
    ----- 4.0/4.0 MB 18.4 MB/s eta 0:00:00
Installing collected packages: lxml
Successfully installed lxml-6.0.0

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Kawan_BPK>
```

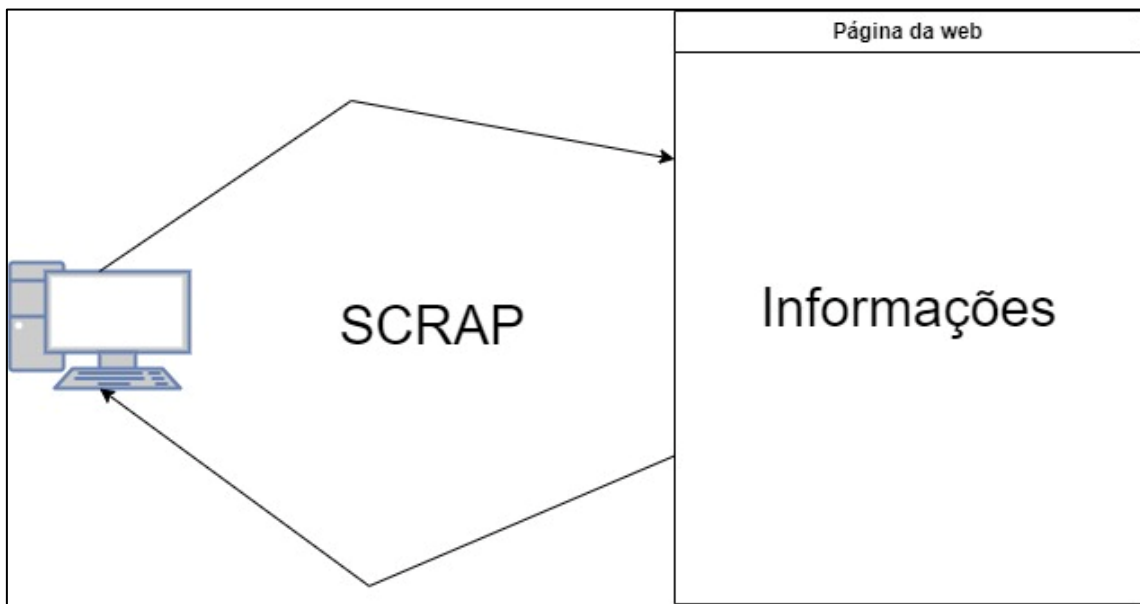
Elaborado pelo autor, 2025.

### 1.3 Scraping Usage, Local files and Web Browser Inspect Tool

Depois de instalar a biblioteca, podemos baixar o arquivo html e trabalhar com ele de forma local, para isso precisamos importar o arquivo usando ‘with open()’, definir o comando ‘r’ para apenas leitura e atribuir um alias. Podemos usar o comando .read() para ler o html e depois disso aplicar o comando ‘BeautifulSoup()’ para visualizar o código html e conseguir trabalhar com o mesmo.

Com a página importada no código, começamos o scrap, para encontrar um elemento, utilizamos o comando .find() e .find\_all(). O primeiro retorna o primeiro item da busca enquanto o segundo retorna todos os itens da busca.

IMAGEM 5 – SCRAP



Elaborado pelo autor, 2025.

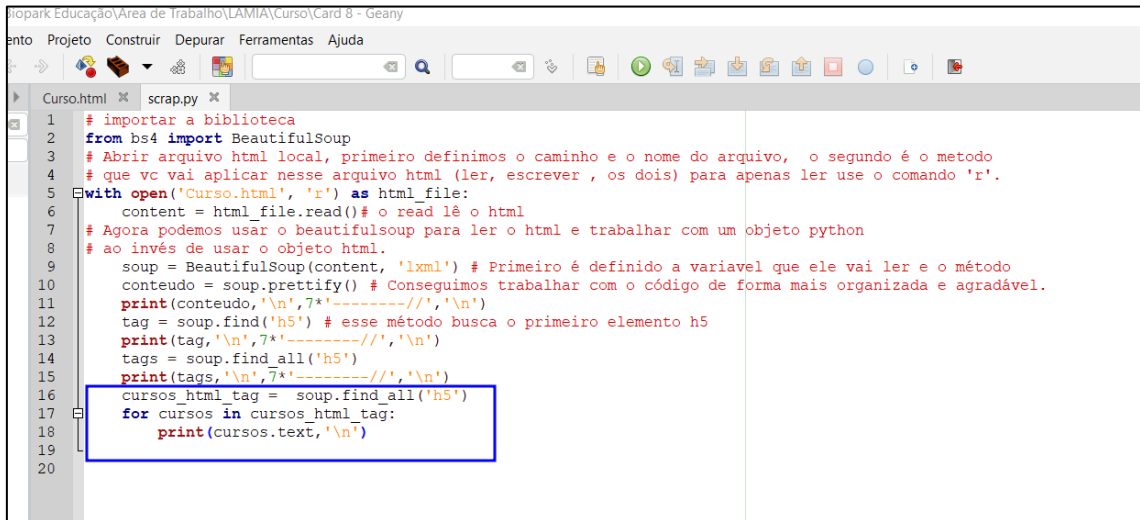
IMAGEM 6 – FIND E FIND\_ALL

```
ark Educação\Área de Trabalho\LAMIA\Curso\Card 8 - Geany
Projeto Construir Depurar Ferramentas Ajuda
Curso.html x scrap.py x
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 # Abrir arquivo html local, primeiro definimos o caminho e o nome do arquivo, o segundo é o metodo
4 # que vc vai aplicar nesse arquivo html (ler, escrever, os dois) para apenas ler use o comando 'r'.
5 with open('Curso.html', 'r') as html_file:
6     content = html_file.read() # o read lê o html
7     # Agora podemos usar o BeautifulSoup para ler o html e trabalhar com um objeto python
8     # ao invés de usar o objeto html.
9     soup = BeautifulSoup(content, 'lxml') # Primeiro é definido a variavel que ele vai ler e o método
10    conteudo = soup.prettify() # Conseguimos trabalhar com o código de forma mais organizada e agradável.
11    print(conteudo, '\n', 7*'-')
12    tag = soup.find('h5') # esse método busca o primeiro elemento h5
13    print(tag, '\n', 7*'-')
14    tags = soup.find_all('h5')
15    print(tags)
16
```

Elaborado pelo autor, 2025.

Nesse exemplo, sabemos que todos os elementos h5 contém os títulos dos cursos ofertados, podemos criar uma lista com todos os cursos ofertados, para isso armazenamos todos os resultados da busca 'h5' e iteramos sobre esse resultado um loop 'for', trazendo desse loop apenas os valores do tipo texto.

## IMAGEM 7 – EXTRAINDO H5

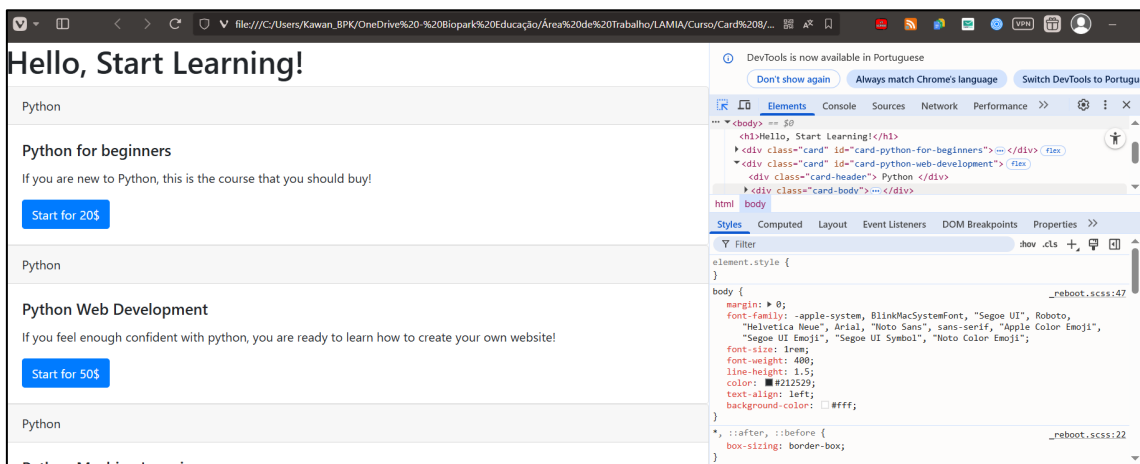


```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 # Abrir arquivo html local, primeiro definimos o caminho e o nome do arquivo, o segundo é o metodo
4 # que vc vai aplicar nesse arquivo html (ler, escrever, os dois) para apenas ler use o comando 'r'.
5 with open('Curso.html', 'r') as html_file:
6     content = html_file.read() # o read lê o html
7     # Agora podemos usar o BeautifulSoup para ler o html e trabalhar com um objeto python
8     # ao invés de usar o objeto html.
9     soup = BeautifulSoup(content, 'lxml') # Primeiro é definido a variavel que ele vai ler e o método
10    conteudo = soup.prettify() # Conseguimos trabalhar com o código de forma mais organizada e agradável.
11    print(conteudo, '\n', 7*'-----//', '\n')
12    tag = soup.find('h5') # esse método busca o primeiro elemento h5
13    print(tag, '\n', 7*'-----//', '\n')
14    tags = soup.find_all('h5')
15    print(tags, '\n', 7*'-----//', '\n')
16    cursos_html_tag = soup.find_all('h5')
17    for cursos in cursos_html_tag:
18        print(cursos.text, '\n')
19
20
```

Elaborado pelo autor, 2025.

Nesse exemplo, o arquivo html é simples e permite visualizar de forma fácil onde estão os elementos de cada código na página. No dia a dia é necessário fazer uso das ferramentas de análise das páginas para conseguir encontrar o elemento o qual se deseja extrair.

## IMAGEM 8 – ANÁLISE DE UMA PÁGINA



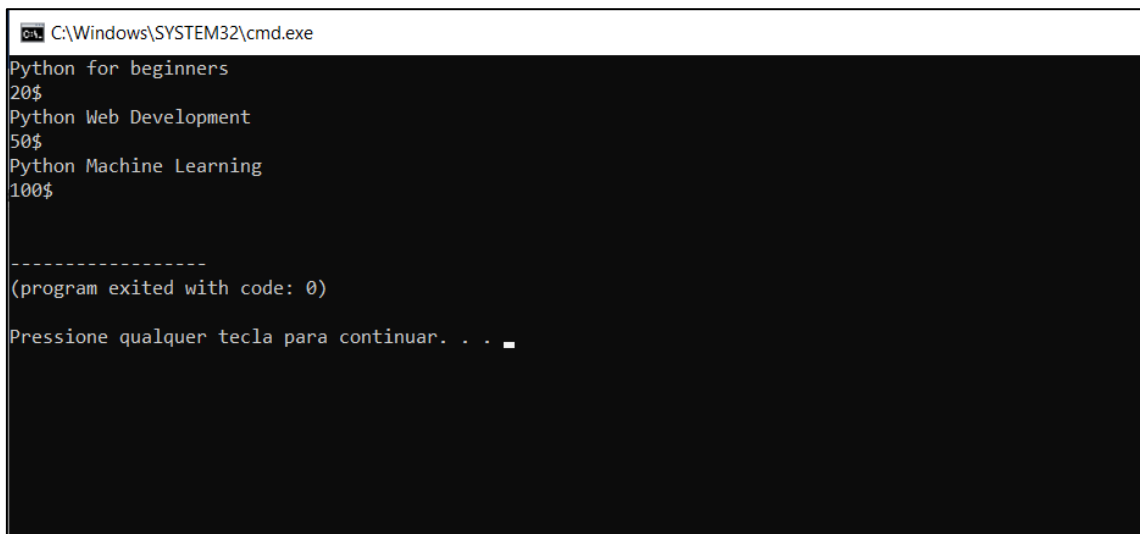
Elaborado pelo autor, 2025.

### 1.4 Grab all Prices, Basic Scraping Project

Ainda nesse exemplo, digamos que queremos extrair o nome do curso e os preços, para isso devemos selecionar todas as 'div' de uma mesma classe, fazendo uso do comando 'find\_all' filtrando todas as 'div' do tipo 'card'.

Com todas as ‘div’ contendo o valor que queremos extrair (curso e preço), podemos iterar sobre essa lista e extrair tanto os ‘textos’ contendo o nome do curso, quanto os textos contendo os valores de cada curso. O campo com o preço irá trazer todo o texto junto (exemplo, ‘*Start for 20\$*’), para evitar isso, podemos usar o comando `.split()[-1]`, o qual irá cortar o valor dos textos de trás para frente, dessa forma, mantendo apenas os valores numéricos.

## IMAGEM 9 – PREÇOS E NOMES



```
C:\Windows\SYSTEM32\cmd.exe
Python for beginners
20$
Python Web Development
50$
Python Machine Learning
100$

-----
(program exited with code: 0)
Pressione qualquer tecla para continuar. . . .
```

Elaborado pelo autor, 2025.

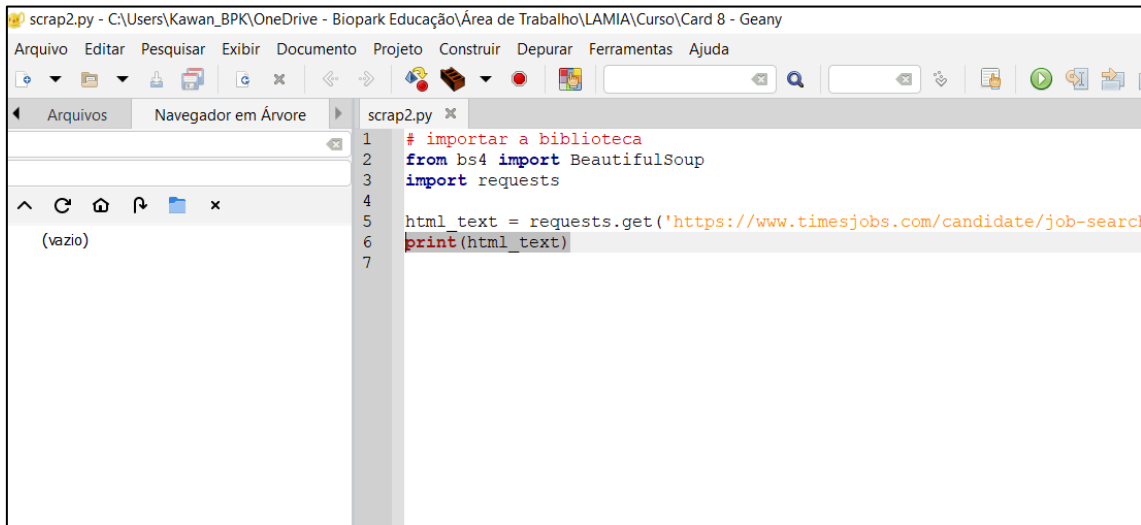
## 2. Website Scraping

### 2.1 Grab all Prices, Basic Scraping Project

Agora que conseguimos entender o básico, vamos começar a extrair informações de sites reais. Para isso vamos precisar da biblioteca ‘*requests*’, a qual podemos instalar usando o comando ‘*pip install requests*’.

Prosseguimos, o site que vamos realizar o scrap é o <https://www.timesjobs.com>, vamos usar o método ‘get’, com o qual conseguimos pegar uma informação específica de um website como se fosse uma pessoa normal acessando a página, o que no nosso caso será o conjunto de vagas para a pesquisa da palavra ‘python’.

## IMAGEM 10 – GET



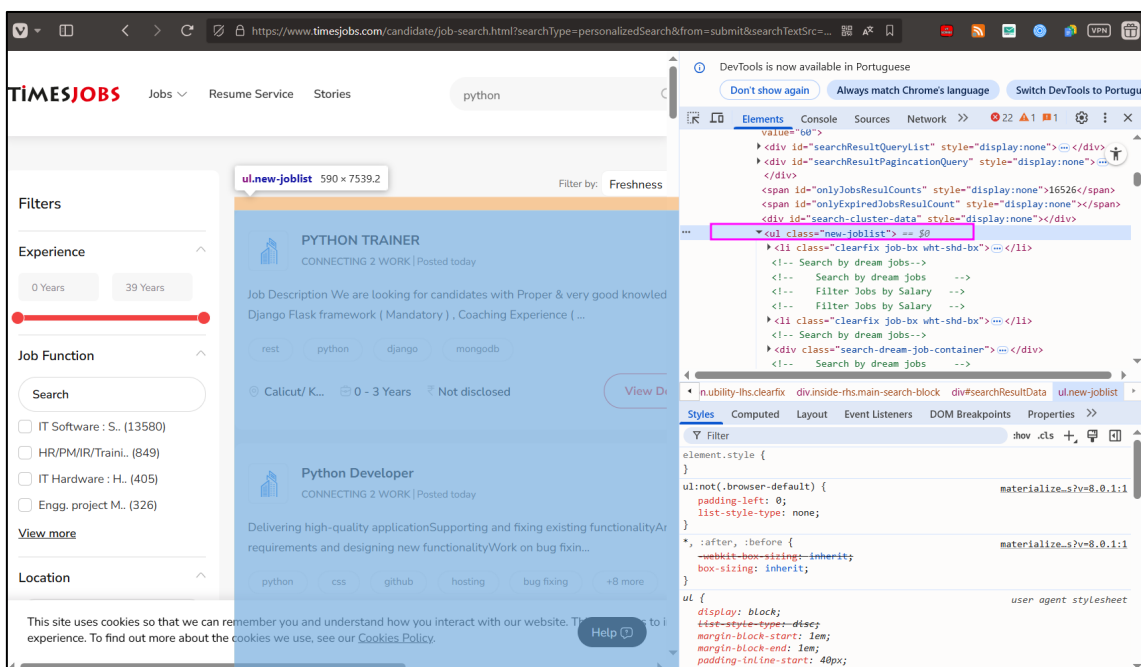
Elaborado pelo autor, 2025.

### 2.2 Scraping a Production Website, Best practices for pulling info

Para praticar como seria um scrap no mundo real, iremos usar o site <https://www.timesjobs.com> a qual se trata de um site de pesquisa de vagas, para o nosso exemplo iremos fazer um scrap da página com a pesquisa da palavra ‘python’ na busca.

Primeiro, precisamos acessar a ferramenta de desenvolvedor do navegador para analisar o html da página, e encontramos a seção que contém todos os cards de pesquisa da vaga.

## IMAGEM 11 – ENCONTRANDO ELEMENTOS



Observando o código conseguimos identificar que dentro do elemento ‘ul’ (unordered list) temos os elementos ‘li’ (list item) que representa cada card da busca de forma individual. Cada elemento ‘li’ pertence a classe ‘clearfix job-bx wht-shd-bx’, essa classe será inserida na busca para exibir todos os itens da lista.

The screenshot shows a Windows desktop with two applications open. On the left is a web browser displaying a search results page for 'PYTHON' jobs on the TimesJobs website. The page lists several job openings, including 'Python Developer', 'Python Backend Developer', and 'Python Frontend Developer'. On the right is a code editor window titled 'Selecionar C:\Windows\SYSTEM32\cmd.exe'. The editor contains a Python script named 'scrap2.py' that uses the BeautifulSoup library to scrape the job listings from the browser's HTML content. The script sends a GET request to the TimesJobs URL, parses the HTML, and prints the job details.

```

# Importar a biblioteca
from bs4 import BeautifulSoup
import requests

html_text = requests.get('https://www.timesjobs.com/candidate/job-se')
soup = BeautifulSoup(html_text, 'lxml')
jobs = soup.find_all('li', class_ = 'clearfix job-bx wht-shd-bx')
print(jobs)

```

The HTML content in the browser shows the following structure for the job listings:

```

<li class="clearfix job-bx wht-shd-bx">
  <div class="d-flex d-flex-l-r job-title__logo">
    <div class="d-flex d-flex-l-r">
      <span class="logo-container">
        <i class="default-company-logo"></i>
      </div>
      <h2 class="heading-trun" title="PYTHON TRAINER">
        <a href="https://www.timesjobs.com/job-detail/python-work-calicut-kozhikode-0-to-3-yrs-jobid-Z_PLUS_uAgZw=&source=srp" onclick="logViewUSBT('view', python , django , mongodb','Calicut/ Kozhikode', Software Products & Services','1','')">
          <strong class="blkclor">PYTHON</strong> TRAINER</a>
        <div class="d-flex d-flex-align-item">
          <h3 class="joblist-comp-name">
            CONNECTING 2 WORK
          </h3>
        </div>
      </div>
    </div>
    <span class="sim-posted">
      <span>Posted today</span>
    </span>
  </div>
</li>

```

Para facilitar o alteramos o código para usar apenas ‘find’, com isso, garantimos que retorne apenas o primeiro resultado da busca, em seguida localizamos onde se encontra a informação do nome da empresa e fazemos a pesquisa da classe dentro da variável ‘job’.

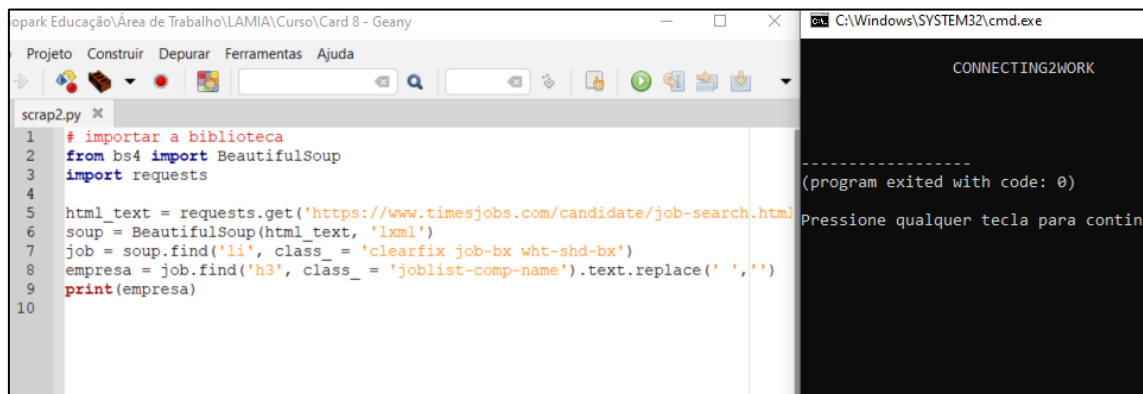
The image shows a Windows desktop with two windows. The left window is a web browser displaying a job listing page from timesjobs.com. The right window is a command prompt showing the output of a Python script. The script uses BeautifulSoup to parse the HTML of the job listing page and prints the company name, which is 'empresap'. The command prompt shows the script's output: 'empresap'.

Elaborado pelo autor, 2025.



Podemos aplicar o método ‘.text’ junto com ‘.replace’. O primeiro irá trazer apenas o texto ao invés do elemento html todo e o segundo irá remover, no nosso caso, espaços por nenhum espaço.

IMAGEM 14 – MÉTODO TEXT E REPLACE

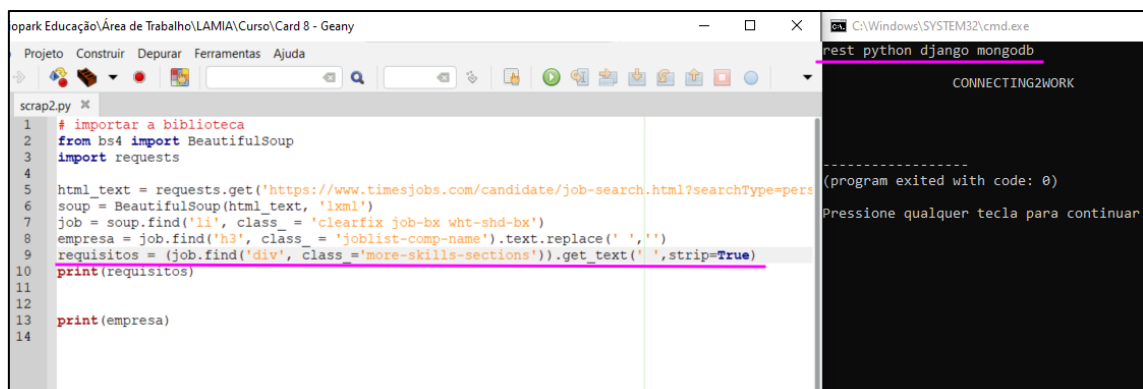


```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html')
6 soup = BeautifulSoup(html_text, 'lxml')
7 job = soup.find('li', class_='clearfix job-bx wht-shd-bx')
8 empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ','')
9 print(empresa)
10
```

Elaborado pelo autor, 2025.

Aplicamos a mesma forma para encontrar os requisitos necessário para a vaga, a diferença, é que foi necessário encontrar a classe que continha as informações dos requisitos e aplicar a função ‘.get\_text()’.

IMAGEM 15 – REQUISITOS

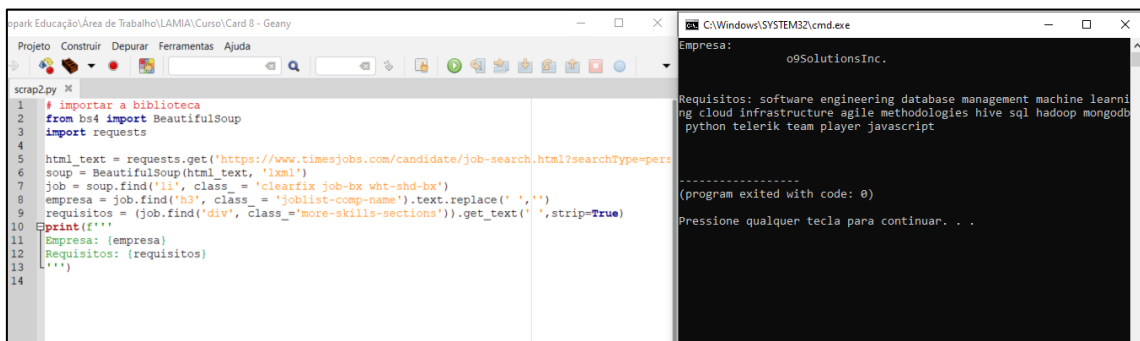


```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=per')
6 soup = BeautifulSoup(html_text, 'lxml')
7 job = soup.find('li', class_='clearfix job-bx wht-shd-bx')
8 empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ','')
9 requisitos = (job.find('div', class_='more-skills-sections')).get_text(' ',strip=True)
10 print(requisitos)
11
12
13 print(empresa)
14
```

Elaborado pelo autor, 2025.

Por fim, inserimos uma apresentação dos resultados usando f string com três aspas para imprimirmos as variáveis cada uma em uma linha. Abaixo o resultado final:

IMAGEM 16 – RESULTADO FINAL



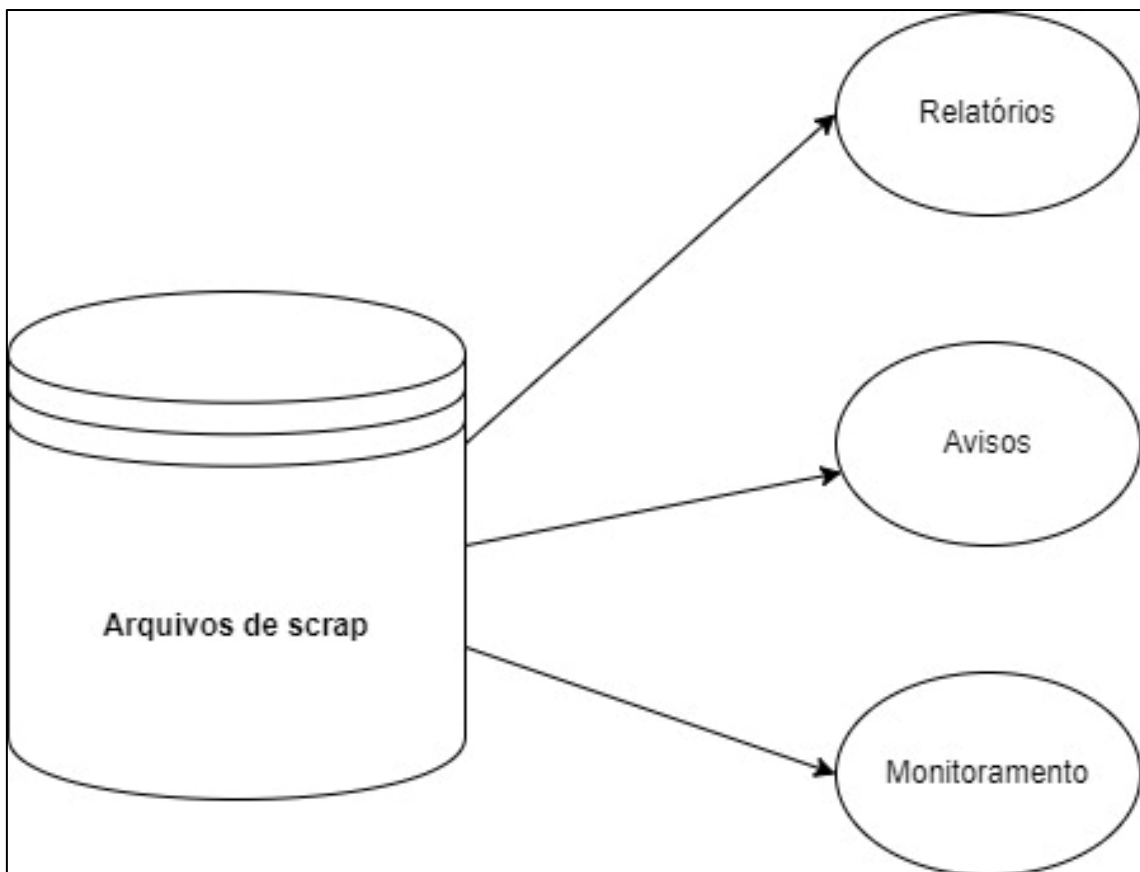
The image shows a code editor window on the left and a terminal window on the right. The code editor contains a Python script named 'scrap2.py' that uses 'requests' and 'BeautifulSoup' to fetch data from a job listing website. The script extracts the company name and a list of requirements. The terminal window shows the output of the script, displaying the company name 'o9SolutionsInc.' and a list of requirements including 'software engineering', 'database management', 'machine learning', etc.

```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=per
6 soup = BeautifulSoup(html_text, 'lxml')
7 job = soup.find('li', class_='clearfix job-bx wht-shd-bx')
8 empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ', '')
9 requisitos = (job.find('div', class_='more-skills-sections')).get_text(' ', strip=True)
10
11 print(f'''
12     Empresa: {empresa}
13     Requisitos: {requisitos}
14 ''')
```

```
Empresa:
o9SolutionsInc.
Requisitos: software engineering database management machine learning cloud infrastructure agile methodologies hive sql hadoop mongodb python telerik team player javascript
(program exited with code: 0)
Pressione qualquer tecla para continuar. . .
```

Elaborado pelo autor, 2025.

IMAGEM 17 – ALGUNS OBJETIVOS DO SCRAP



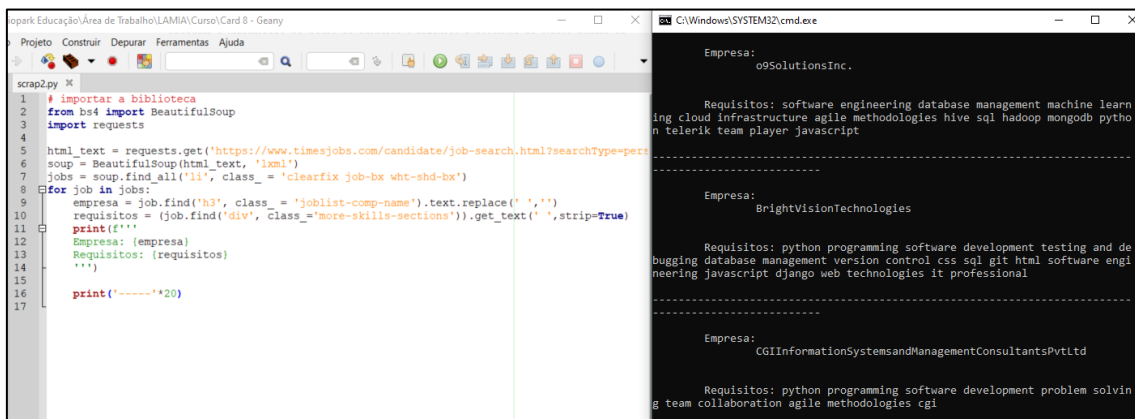
Elaborado pelo autor, 2025.

### 2.3 Looping through similar soup.find\_all() objects

Quando estamos tratando de um scrap, geralmente estamos buscando todas as informações contidas em determinado site, no nosso caso, queremos encontrar todos os trabalhos da página 1, para isso é necessário adaptar o código com um comando de loop 'for'.

Primeiro alteramos o nome da variável ‘job’ para ‘jobs’, depois modificamos o método de busca ‘find’ para ‘find\_all’ e depois executamos um loop ‘for’ para iterar sobre ‘jobs’.

IMAGEM 18 – ITERANDO SOBRE JOBS



```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=per
6 soup = BeautifulSoup(html_text, 'lxml')
7 jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')
8
9 for job in jobs:
10     empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ','')
11     requisitos = (job.find('div', class_='more-skills-sections')).get_text(' ',strip=True)
12     print(f'''
13     Empresa: {empresa}
14     Requisitos: {requisitos}
15     ''')
16
17 print('-----'*20)
```

```
Empresa:
o9SolutionsInc.

Requisitos: software engineering database management machine learn
ing cloud infrastructure agile methodologies hive sql hadoop mongodb pytho
n telerik team player javascript

-----

Empresa:
BrightVisionTechnologies

Requisitos: python programming software development testing and de
bugging database management version control css sql git html software engi
neering javascript django web technologies it professional

-----

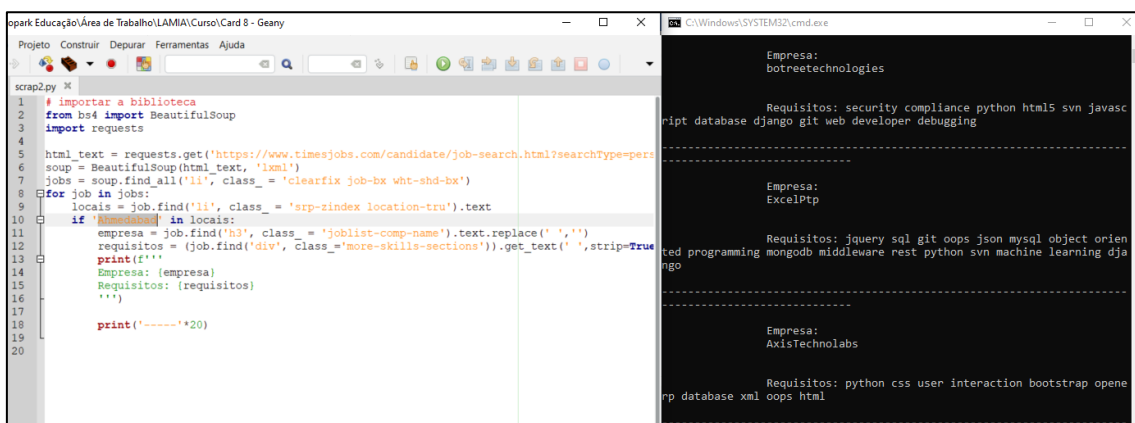
Empresa:
CGIInformationSystemsandManagementConsultantsPvtLtd

Requisitos: python programming software development problem solvin
g team collaboration agile methodologies cgi
```

Elaborado pelo autor, 2025.

Além disso, podemos aplicar um filtro para por exemplo, buscar os trabalhos de determinada cidade apenas, isso diminui a busca e é uma forma eficiente de focar o scrap em uma informação específica. No nosso caso, vamos buscar apenas vagas do país ‘Ahmedabad’.

IMAGEM 19 – VAGAS APENAS DE UM LOCAL



```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search.html?searchType=per
6 soup = BeautifulSoup(html_text, 'lxml')
7 jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')
8
9 for job in jobs:
10     localis = job.find('li', class_='srp-zindex location-tru').text
11     if 'Ahmedabad' in localis:
12         empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ','')
13         requisitos = (job.find('div', class_='more-skills-sections')).get_text(' ',strip=True)
14         print(f'''
15         Empresa: {empresa}
16         Requisitos: {requisitos}
17         ''')
18
19 print('-----'*20)
```

```
Empresa:
botreetechnologies

Requisitos: security compliance python html5 svn javasc
ript database django git web developer debugging

-----

Empresa:
ExcelPtp

Requisitos: jquery sql git oops json mysql object orien
ted programming mongodb middleware rest python svn machine learning dja
ngo

-----

Empresa:
AxisTechnolabs

Requisitos: python css user interaction bootstrap opene
rp database xml oops html
```

Elaborado pelo autor, 2025.

### 3. Features addition

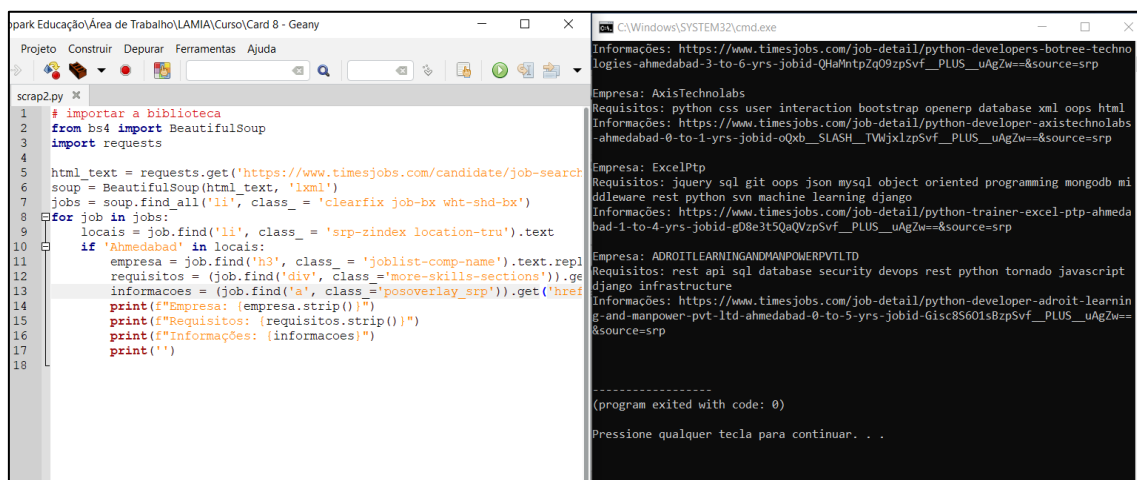
#### 3.1 Looping through similar soup.find\_all() objects

Agora que nosso código está organizado precisamos melhorar a exibição dos resultados, primeiro ajustamos a forma como os resultados são exibidos alterando o

código *'print'* e depois iremos inserir o link para a pessoa acessar as vagas diretamente do scrap.

Para ajustar o print apenas criamos um print para cada variável e inserimos o método *'strip()'*, o qual elimina espaços em branco antes e depois do texto. Depois buscamos o elemento que contenha o link e extraímos da mesma maneira que os anteriores, usando *'find'* e o método *'get()'* para podermos extrair o link contido em *'href'*.

IMAGEM 20 – PRINT E LINK



The image shows a screenshot of a code editor and a terminal window. The code editor on the left displays a Python script named `scrap2.py` that uses `requests` and `BeautifulSoup` to scrape job listings from `timesjobs.com`. The script iterates through job listings, finds specific elements like company names and requirements, and prints them with `strip()` to remove extra spaces. The terminal window on the right shows the output of the script, displaying details for three different jobs, including company names, requirements, and full URLs.

```
1 # importar a biblioteca
2 from bs4 import BeautifulSoup
3 import requests
4
5 html_text = requests.get('https://www.timesjobs.com/candidate/job-search')
6 soup = BeautifulSoup(html_text, 'lxml')
7 jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')
8 for job in jobs:
9     locals = job.find('li', class_='srp-zindex location-tru').text
10    if 'Ahmedabad' in locals:
11        empresa = job.find('h3', class_='joblist-comp-name').text.replace(' ', '')
12        requisitos = (job.find('div', class_='more-skills-sections')).get('text')
13        informacoes = (job.find('a', class_='posoverlay_srp')).get('href')
14        print(f"Empresa: {empresa.strip()}")
15        print(f"Requisitos: {requisitos.strip()}")
16        print(f"Informações: {informacoes}")
17        print('')
```

Informações: https://www.timesjobs.com/job-detail/python-developers-botnee-technologies-ahmedabad-3-to-6-yrs-jobid-QHaMtpZq09zpSvf\_\_PLUS\_uAgZw==&source=srp  
Empresa: AxisTechnolabs  
Requisitos: python css user interaction bootstrap openerp database xml oops html ddleware rest python svm machine learning django  
Informações: https://www.timesjobs.com/job-detail/python-developer-axistechnolabs-ahmedabad-0-to-1-yrs-jobid-oQxb\_SLASH\_TVWjxlzpSvf\_\_PLUS\_uAgZw==&source=srp  
Empresa: ExcelPtp  
Requisitos: jquery sql git oops json mysql object oriented programming mongodb mi  
Informações: https://www.timesjobs.com/job-detail/python-trainer-excel-tp-ahmedabad-1-to-4-yrs-jobid-gD8e3t5QaQVzpSvf\_\_PLUS\_uAgZw==&source=srp  
Empresa: ADROITLEARNINGANDMANPOWERPVTLTD  
Requisitos: rest api sql database security devops rest python tornado javascript django infrastructure  
Informações: https://www.timesjobs.com/job-detail/python-developer-adroit-learning-and-manpower-pvt-ltd-ahmedabad-0-to-5-yrs-jobid-Gisc8S601sBzpSvf\_\_PLUS\_uAgZw==&source=srp  
-----  
(program exited with code: 0)  
Pressione qualquer tecla para continuar. . .

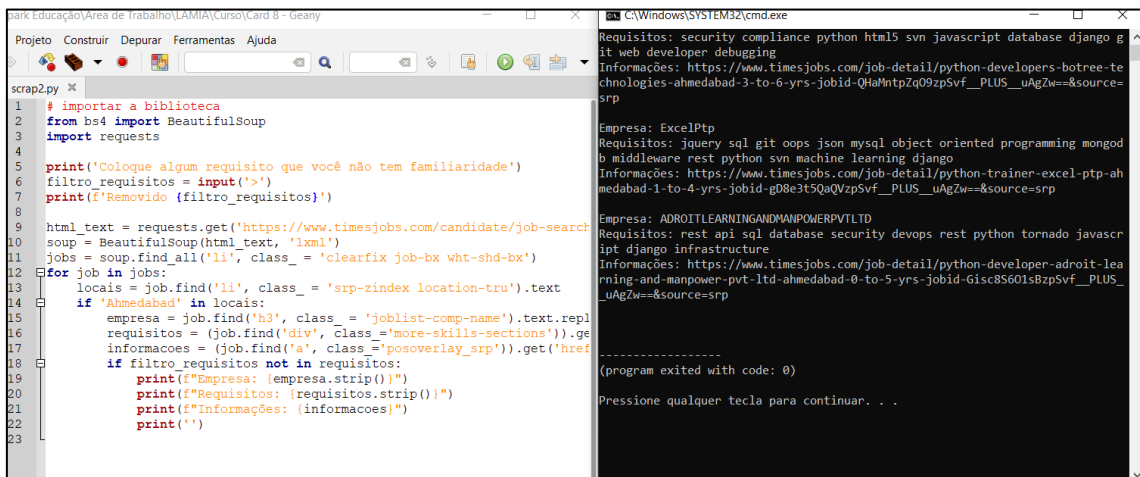
Elaborado pelo autor, 2025.

### 3.2 Jobs Filtration by owned skills

Agora implementaremos a opção de o usuário poder remover resultados do scrap, para isso, precisamos implementar um comando *'input()'* a qual irá receber a variável que está sendo removida da pesquisa de vagas, no nosso caso, um requisito.

Depois, inserimos um comando *'if'* dentro do loop o qual funciona da seguinte maneira: caso o valor digitado no input não esteja dentro da variável *requisitos* imprime o resultado, caso contrário não imprime.

## IMAGEM 21 – FILTRO DE REQUISITOS

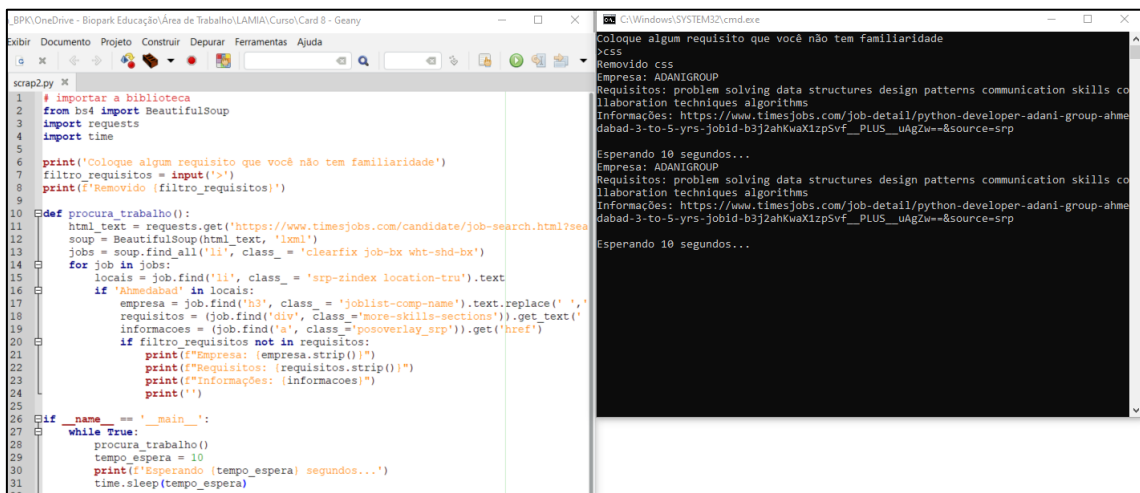


Elaborado pelo autor, 2025.

### 3.3 Setting up the Project to scrape every 10 minutes

Agora vamos ajustar para o código executar a cada 10 minutos uma busca no site. Para isso, precisamos importar a biblioteca `'time'`, criamos uma função para receber a parte do código que de fato faz o scrap no site, definimos esse arquivo `'.py'` como arquivo principal através de `'__name__' == '__main__'` e criamos um loop `'while True'`, o qual irá ficar executando infinitamente buscando de dez em dez segundos no site.

## IMAGEM 22 – EXECUÇÃO A CADA 10 SEGUNDOS



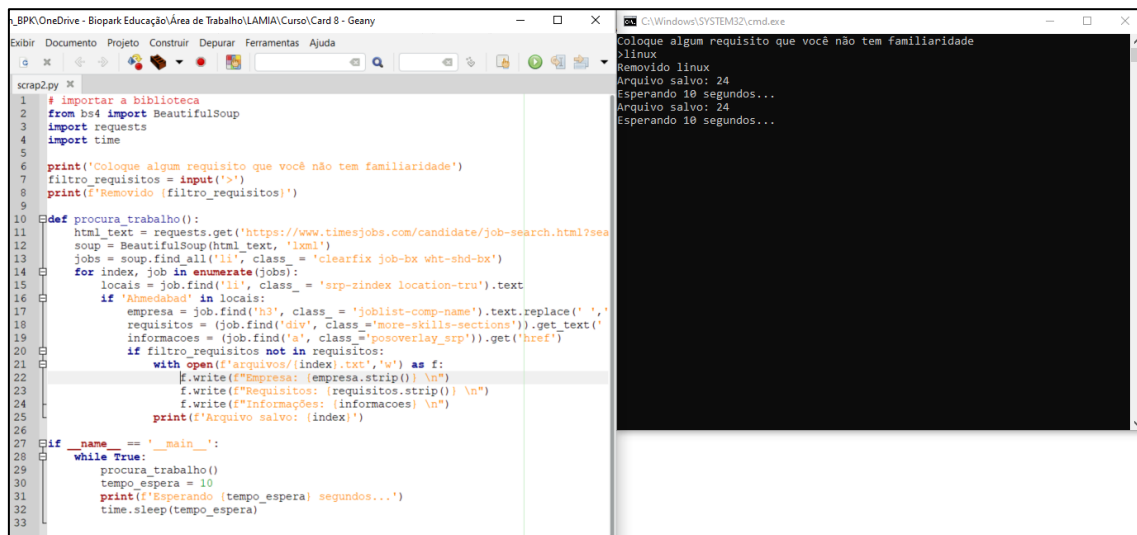
Elaborado pelo autor, 2025.

### 3.4 Storing the jobs paragraph in text files

Por último iremos armazenar os resultados da busca em um arquivo ‘.txt’, para isso precisamos alterar o loop ‘for’, para usar o método *enumerate()* e uma nova variável *index*. Depois, dentro do segundo *if*, inserimos o comando ‘*with open()*’, onde configuramos a pasta onde ele irá armazenar o arquivo *.txt* com o seu respectivo nome e a permissão para poder escrever no arquivo.

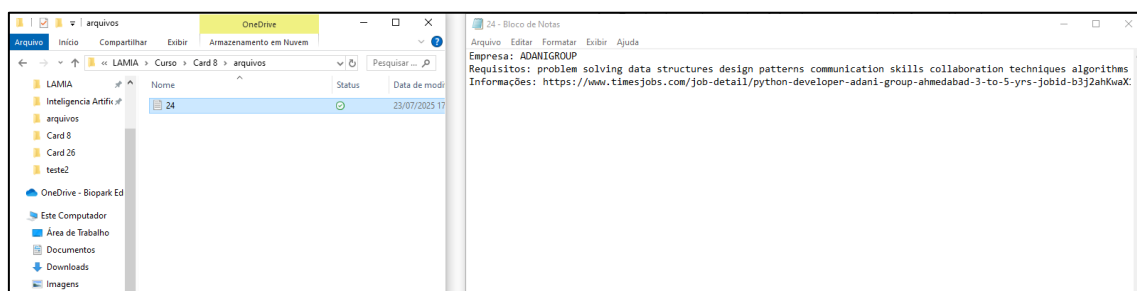
Por último usamos ‘*as f:*’ para definir que tudo isso é um arquivo, e colocamos dentro do *with* os prints configurados anteriormente, alterando para *f.write* e ao final exibimos uma mensagem de finalização do processo.

IMAGEM 23 – CRIANDO ARQUIVOS A CADA 10S



Elaborado pelo autor, 2025.

IMAGEM 24 – EXEMPLO DE ARQUIVO CRIADO



Elaborado pelo autor, 2025.

### Conclusões

O scrap é uma técnica muito importante no que tange trabalhos repetitivos, pois com ela conseguimos criar automações que vão diariamente buscar a informação em

determinado site, de forma contínua e persistente, liberando seu usuário para outros processos.

Para ter um scrap efetivo é preciso saber e entender o código da página, logo, ter um conhecimento de *html* é imprescindível para conseguir programar bons códigos de scrap. Além disso, saber como usar a ferramenta de análise de página web, é extremamente importante, pois é ela que vai facilitar o entendimento do código e possibilitar entender a estrutura de como foi construído o site.

Para além desses pontos, é importante sempre estar praticando a escrita de códigos, durante esse processo, é uma boa prática, iniciar a construção do código de forma gradual ao invés de já tentar construir algo muito complexo. Também é importante notar que o exemplo dado no vídeo já permite aplicar para uma gama alta de opções, por exemplo, monitoradores de preço de um determinado item, monitoramento de vagas, monitoramento de postagens, etc.

#### Referencias

As referências foram apenas os vídeos da atividade.