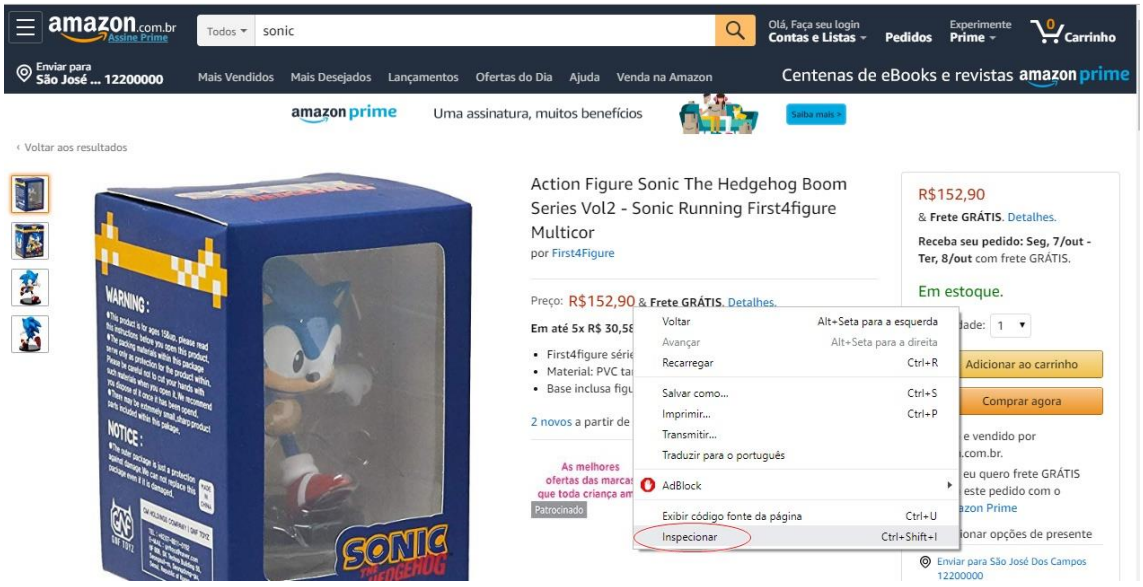
web scraping

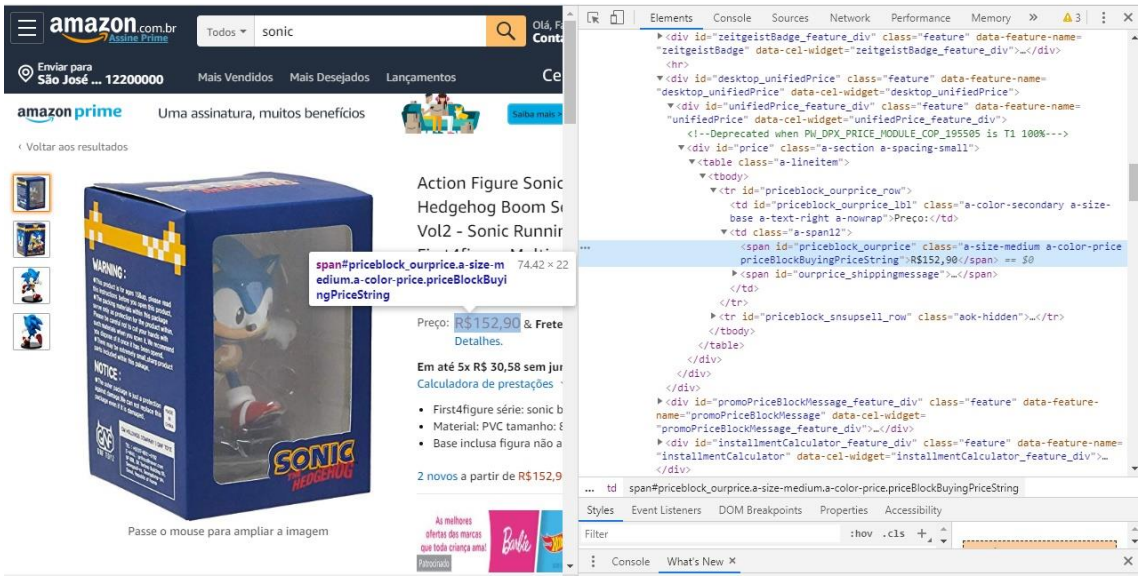
Web Scraping

1º - Verificar o que queremos no código da página:
Primeiramente precisamos definir um site o qual queremos fazer a raspagem de dados. Tendo a página aberta vamos analisar quais dados são de fato importantes para o funcionamento de nosso bot, ao indentificar estes elementos iremos clicar com o botão direito sobre ele e após isso iremos em “Inspecionar”.



Neste exemplo da foto iremos fazer uma raspagem no site da Amazon e nos limitanto a apenas este produto.

Ao clicar em Inspecionar abriremos uma pequena aba no navegador contendo as informações do código da página, é possível observar as tags utilizadas e toda a estrutura que será útil posteriormente.



Você terá uma tela semelhante a esta em seu computador, nota que ao passar o mouse sobre o código da página, alguns elementos do site ficarão destacados, este que são correspondentes a determinada tag na estrutura.

2º - Preparando o ambiente

Antes de começar a programar, temos que importar duas bibliotecas que serão muito importantes para o desenvolvimento de nosso código. Para instalar estas bibliotecas temos que abrir o “cmd.exe” (caso você esteja utilizando Windows) e digitar “pip install *nome_da_biblioteca*”. As duas bibliotecas que teremos que instalar são: **lxml** e **requests.**

Exemplo:

3º - Primeiros passos na programação

Este é o momento onde iremos criar um novo arquivo Python, pode ser utilizada a IDE de preferência para este processo.

As duas primeiras do nosso código será destinada a importação das bibliotecas, para fazer essa importação basta digitar “Import *nome_da_biblioteca*”. Teremos nossas duas importações assim:

```
import requests
import lxml.html
```

A biblioteca “requests” é responsável pelas solicitações em HTTP que iremos fazer e a biblioteca “lxml.html” será responsável pela raspagem de informações.

Após a importação de bibliotecas podemos definir o caminho onde efetuaremos a raspagem, por motivos de organização vamos atribuir uma variável para armazenar o

caminho pois algumas vezes este caminho é extenso e para não deixar uma linha muito grande atribuiremos a variável “url”.

```
url = 'https://www.amazon.com.br/Action-Figure-Sonic-Hedgehog-
Boom/dp/B01AKT1Y50/ref=sr_1_1?__mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&
keywords=sonic&qid=1569325718&s=gateway&sr=8-1'
```

O link armazenado nesta variável corresponde ao boneco do Sonic encontrado na Amazon.

Com o caminho definido podemos dar inicio as funções de nosso bot. O primeiro passo que devemos dar é realizar uma requisição na Internet, para isso utilizaremos a sintaxe: “*variavel* = requests.get(*caminho,* header={}*)”.*

```
res = requests.get(url, headers={'user-agent':'bot by quakcduck'})
```

Agora que temos nossa requisição feita temos que verificar se ela foi efetiva para que possamos dar continuidade. Alguns sites demandam um certo tempo para realizar essa requisição o que acarreta no status 503, este status nos impede de acessar o código do site pois na teoria a página não foi encontrada pelo Python, alguns motivos que podem causar esse erro são: caminho digitado erroneamente e o site ter diversas requisições simultâneas.

Uma maneira de contornar a segunda hipótese é realizando uma estrutura de repetição chegando o status de nossa requisição. Podemos verificar o status dela através do comando : “*variavel*.status_code”. Precisamos obter o status 200 para que nossa requisição seja efetiva.

Faremos nossa repetição da seguinte forma:

```
while(res.status_code == 503):
    html = requests.get(url)
```

Explicando essa repetição: enquanto o status de nossa repetição for 503 ele fará a requisição até apresentar um código diferente.
obs: é necessário ter a certeza que o caminho esta certo para fazer esta repetição.

Após esta repetição iremos verificar qual o status que diferente do 503 esta em nossa requisição, para isso faremos uma estrutura condicional:

```
if res.status_code == 200:
    print("Conexao bem sucedida")
else
    print(res.status_code)
```

Explicando mais uma vez a sintaxe, caso nosso status seja igual a 200 a requisição foi bem sucedida, agora se não for igual ele ira apresentar o status e devemos procurar o que aquela informação pode significar.

Caso nosso status seja positivo, podemos continuar para o acesso do site em si. O próximo comando que devemos inserir é o “lxml.html.fromstring(*variavel*.content)” este método nos possiblita uma consulta em nosso documento, possiblitando uma

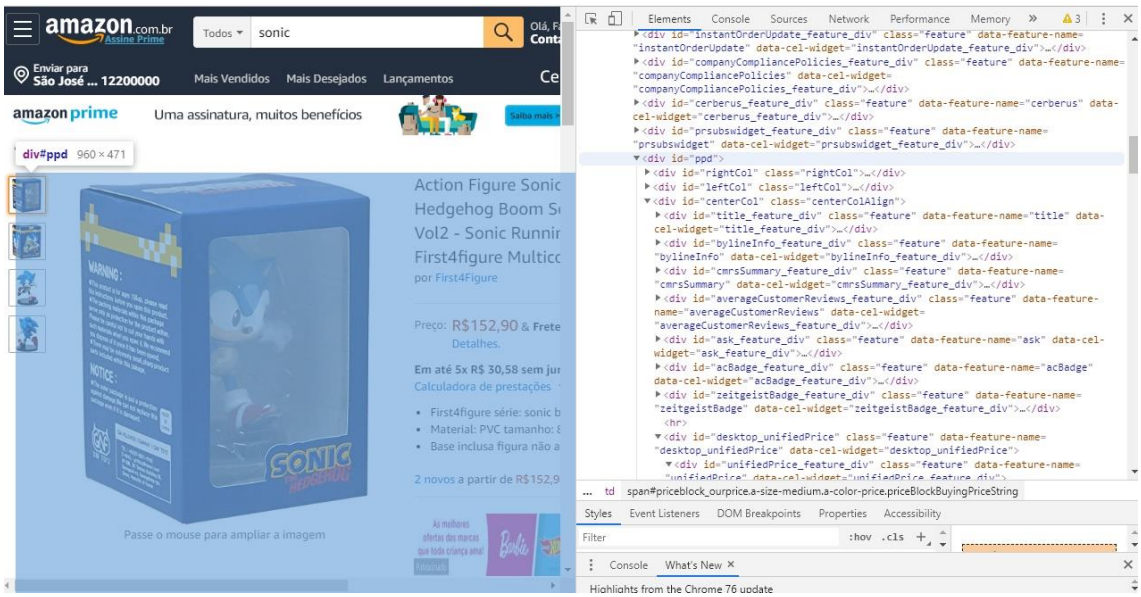
maneira estruturada de extrair informações. Utilizando este método para nosso exemplo ficaria desta maneira:

```
doc = lxml.html.fromstring(res.content)
```

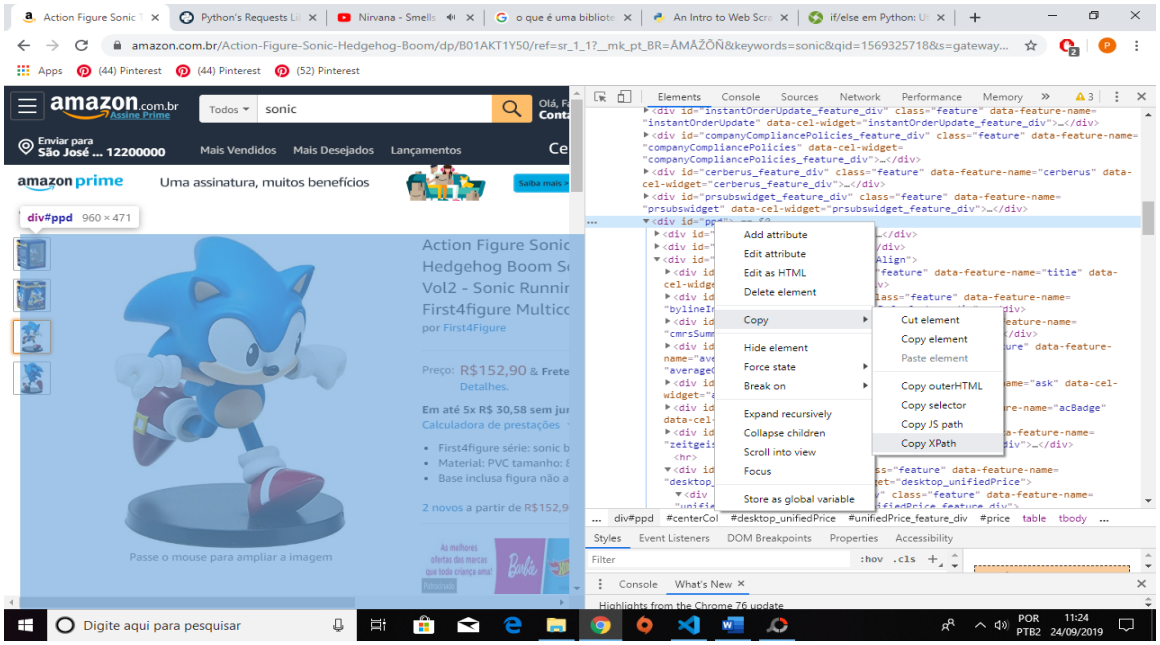
Com o acesso ao conteúdo do site, agora podemos dar início a raspagem de dados.

4º - Raspagem de dados

Para a organização de informações na estruturas, o primeiro conteudo que iremos buscar não será a informação propriamente dita mas sim um **tag** que abrange todas as informações que queremos. Dando o exemplo do site da Amazon:



No exemplo queremos apenas o nome e o preço do produto, note que ao passar o mouse na linha ** ** tanto o nome quanto o preço ficam grifados em azul, esta seria a tag citada anteriormente. Agora que sabemos qual a primeira informação, iremos clicar com o botão direito sobre ela, após isso passar o mouse em cima de “Copy” e por fim clicar em “Copy xPath.



Para trazer esta informação que indetificamos para nosso código iremos utilizar o comando: “*variavel* = doc.xpath(*informação*)[0]”

Adaptando esta linha para nosso exemplo teremos:

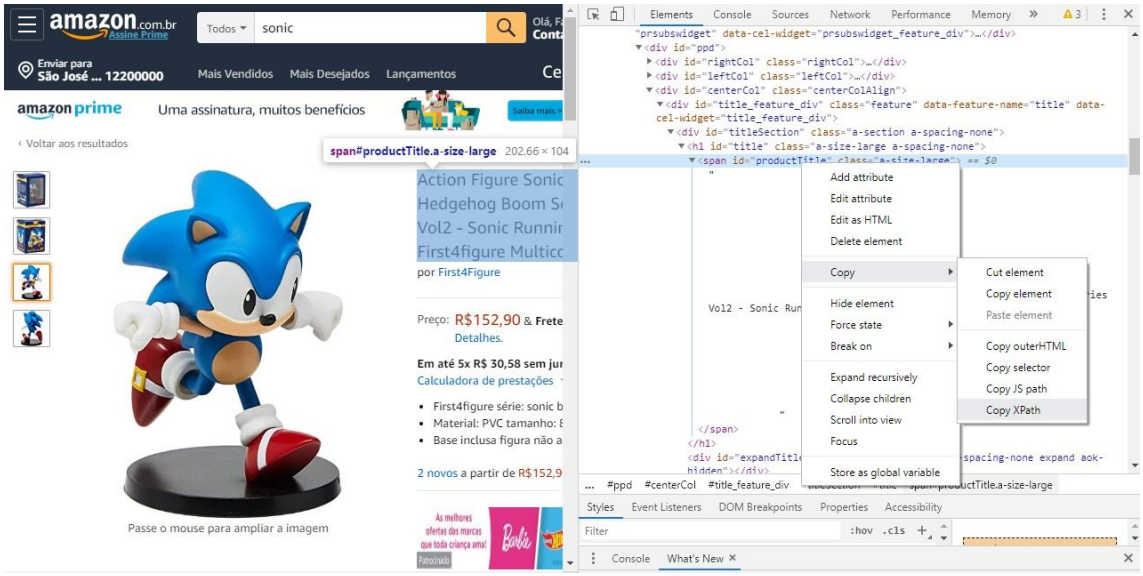
```
corpo = doc.xpath('//*[@id="dp"]')[0]
```

Note que *informação* ira receber o que acabamos de copiar de nosso site base.

Com o “corpo” da raspagem definido podemos apontar para onde estão as informações que queremos tirar do site. Para isso utilizaremos o comando “*variavel* = corpo.*xpath(informação)*”

Agora faremos o processo para identificar o xPath novamente, só que do conteúdo que iremos extrair.

Uma das informações que queremos no exemplo é o nome, para isso clicaremos sobre o nome do produto e faremos o mesmo processo descrito para pegar a informação do corpo da raspagem.

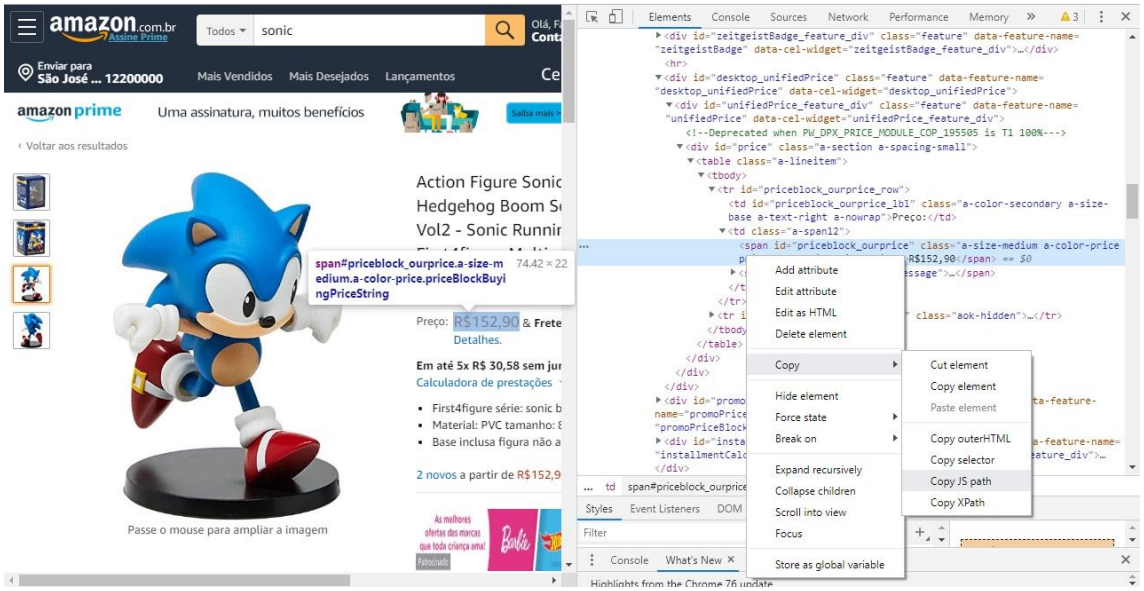


Adaptando o último comando apresentado para nosso exemplo:

```
nome_produto = corpo.xpath('.//*[@id="productTitle"]/text()')
```

Para fins de formatação, é necessário colocar “/text()” após o xPath copiado.

Faremos o mesmo processo para o preço do produto:




```
preco_produto = corpo.xpath('.//*[@id="priceblock_ourprice"]/text()')
```

Por fim teremos nossas duas informações armazenas em variáveis, para conseguirmos mostrar basta utilizar o comando básico de print:

```
print('Nome: ', nome_produto)
print('Preco: ', preco_produto)
```

5º - Código desenvolvido

```
import requests
import lxml.html

url = 'https://www.amazon.com.br/Action-Figure-Sonic-Hedgehog-
Boom/dp/B01AKT1Y50/ref=sr_1_1?__mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&
keywords=sonic&qid=1569325718&s=gateway&sr=8-1'

res = requests.get(url, headers={'user-agent':'bot by quakcduck'})

while(res.status_code == 503):
    html = requests.get(url)

if res.status_code == 200:
    print("Conexao bem sucedida")

doc = lxml.html.fromstring(res.content)

corpo = doc.xpath('//*[@id="dp"]')[0]

nome_produto = corpo.xpath('.//*[@id="productTitle"]/text()')
```

```
preco_produto = corpo.xpath('.//*[@id="priceblock_ourprice"]/text()')

print('Nome: ', nome_produto)
print('Preco: ', preco_produto)
```