



INTENSIVÃO DE PYTHON {#}

100% ONLINE & GRATUITO

Apostila Completa Aula 3

Aprenda como automatizar processos que
tenham interface com a internet
Impressionador do absoluto zero!



Parte 1

Introdução

O que vamos aprender

Na terceira aula da Semana do Python, você vai aprender a criar um código para automação de processos. No dia a dia das empresas, é muito comum que existam operações manuais que além de extremamente repetitivas (chatas) são suscetíveis a erro visto que são feitas de forma manual. Vamos aprender como criar um código com o qual você possa resolver esse problema sem nem tocar no mouse 😊. Aprenda como **fazer uma automação com integração web** com os conceitos abaixo:

Importando bases
de dados do Excel

Jupyter Notebook

Importando
bibliotecas

Webdriver

Usando Selenium

Após todos esse aprendizado, seremos capazes de transformar um processo extremamente repetitivo

	A	B	C	D
1	Produto	Preço Ideal	Preço Atual	Comprar
2	Milho	85,32		
3	Soja	163,59		
4	Boi	282,2		
5	Petróleo	424,37		
6	Algodão	497,76		
7	Açúcar	136,23		
8	Café	1092,87		
9	Ouro	321,77		
10	Trigo	1549,11		
11	Tilápia	9,05		



... em processo automático e sem erros! Tudo graças a você! 😊

	A	B	C	D
1	Produto	Preço Ideal	Preço Atual	Comprar
2	Milho	85,32	85,61	FALSO
3	Soja	163,59	163,07	VERDADEIRO
4	Boi	282,2	280,2	VERDADEIRO
5	Petróleo	424,37	407,76	VERDADEIRO
6	Algodão	497,76	498,48	FALSO
7	Açúcar	136,23	133,46	VERDADEIRO
8	Café	1092,87	1107,71	FALSO
9	Ouro	321,77	322,58	FALSO
10	Trigo	1549,11	1542,87	VERDADEIRO
11	Tilápia	9,05	9,05	FALSO

Introdução

Entendendo o problema

Na nossa empresa fictícia somos uma importadora e vamos comprar commodities.

Nosso trabalho como analista de compras é verificar quando essas commodities estão em um valor “ideal” para que possamos fazer a compra.

É uma tarefa muito simples, mas se você parar para analisar é uma tarefa que feita de forma manual pode levar muito tempo.

Principalmente, porque vamos ter que ficar analisando os preços e comparando com o nosso valor ideal de cada uma das commodities.

	A	B	C	D
1	Produto	Preço Ideal	Preço Atual	Comprar
2	Milho	85,32		
3	Soja	163,59		
4	Boi	282,2		
5	Petróleo	424,37		
6	Algodão	497,76		
7	Açúcar	136,23		
8	Café	1092,87		
9	Ouro	321,77		
10	Trigo	1549,11		
11	Tilápia	9,05		



Entendendo a solução final

Nossa solução final será:

- 1) Pesquisar os valores da commodities;
- 2) Armazenar as informações;
- 3) Preencher a coluna de preço atual no Excel;
- 4) Comparar com o preço ideal;
- 5) Por fim, verificar se devemos comprar ou não.

Tudo isso automaticamente!! Apenas rodando o código que vamos criar.

Milho Hoje

Milho 1,00

VALOR DA SACA DO MILHO ▾

R\$ 85,42

Quer investir em Milho?

Salva como

[Ver valor do Dólar Hoje](#)

Valor da Saca do Milho hoje - 15/03/23 às 09:02

Uma saca de milho hoje vale R\$85,42



	A	B	C	D
1	Produto	Preço Ideal	Preço Atual	Comprar
2	Milho	85,32	85,42	
3	Soja	163,59		
4	Boi	282,2		
5	Petróleo	424,37		
6	Algodão	497,76		
7	Açúcar	136,23		
8	Café	1092,87		
9	Ouro	321,77		
10	Trigo	1549,11		
11	Tilápia	9,05		

Parte 2

Importando o Selenium

Importando o Selenium

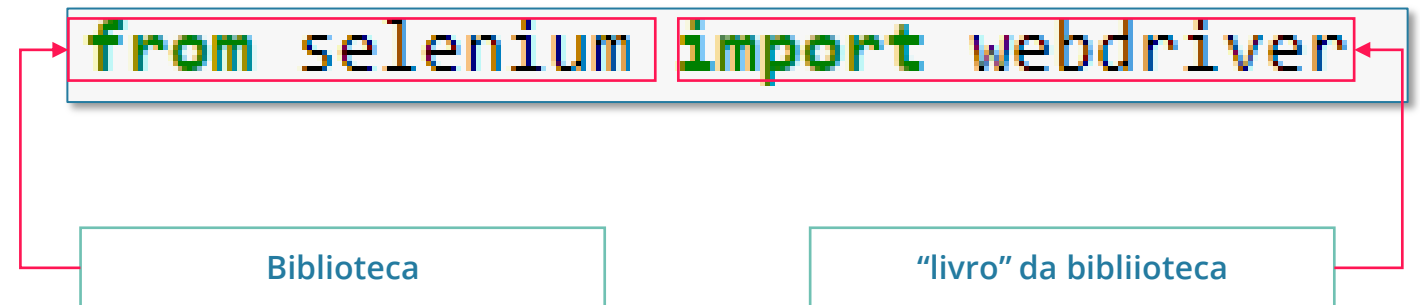
Como vimos na aula 1 da Semana do Python, vamos usar bibliotecas que nos facilitem importar dados de planilhas Excel, arquivos .csv, etc.

No entanto, além do pandas iremos importar o **selenium**.

Antes de entendermos no detalhe o que elas fazem e para que servem, vamos nos atentar a uma diferença na hora da importação.

Podemos perceber que na primeira linha importamos o selenium utilizando a estrutura **from** antes do import.

Essa estrutura significa dizer “dentro da biblioteca selenium importe o livro webdriver”.



Importando o webdriver

Como assim, um livro de uma biblioteca?

Antes de entendermos o código, imagine uma biblioteca de fato.

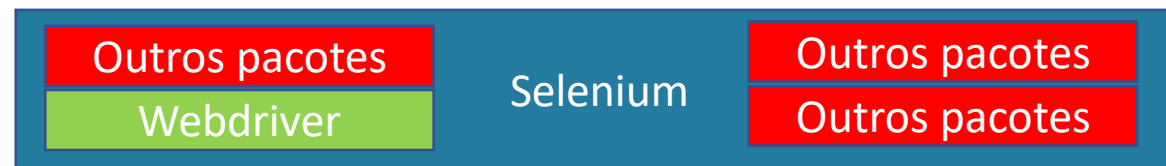
Grande, correto? Muitas das informações ali, são úteis mas não naquele momento...

Possivelmente só vamos conseguir ler 1, 2 ou 3 livros de uma vez. Não faz sentido alugar toda a biblioteca, apenas o que vamos precisar.

Aqui é exatamente a mesma coisa! Não temos que importar toda uma biblioteca se apenas uma parte dessa biblioteca nos interessa.

No nosso caso, o que nos interessa é o **webdriver**.

Isso torna nosso código mais simples e rápido!



Instalando o Selenium

Para essa aula estamos usando o Jupyter e nele já possuímos uma série de pacotes “pré-instalados” mas as vezes precisamos instalar pacotes adicionais.

O Python, possui um “instalador embutido” que se chama **pip**.

Por ele, é possível, instalar e desinstalar pacotes.

Caso você queira saber se o selenium está instalado basta usar o comando abaixo em uma das células do Jupyter:

pip freeze

Caso não encontre na lista o selenium, use o comando abaixo para instalar:

pip install -U selenium

```
pip freeze
argh==0.26.2
asn1crypto==1.3.0
astroid==2.4.2
astropy==4.0.1.post1
atomicwrites==1.4.0
attrs==19.3.0
autopep8 @ file:///tmp/build/80754af9/autopep8_1592412889138/work
Babel==2.8.0
backcall==0.2.0
backports.functools-lru-cache==1.6.1
...
scipy @ file:///C:/ci/scipy_1592916963468/work
seaborn==0.10.1
selenium==3.141.0
Send2Trash==1.5.0
simplegeneric==0.8.1
singledispatch==3.4.0.3
sip==4.19.13
six==1.15.0
snowballstemmer==2.0.0
sortedcollections==1.2.1
sortedcontainers==2.2.2
soupsieve==2.0.1
Sphinx @ file:///tmp/build/80754af9/sphinx_1594223420021/work
```

Comando pip freeze

Lista de pacotes instalados no Python

Parte 3

Interface com uma página na web

Interface com uma página na web

Selenium

Conforme explicamos anteriormente o **selenium** é uma biblioteca assim como o pandas.

Essa biblioteca é muito utilizada como interface com o navegador. Ela funciona como um robô que clica, insere dados, etc em páginas WEB: interagindo com os sites como se fosse você.

Muito útil para processos repetitivos como este que temos aqui.

Aqui temos links de documentação para aqueles que gostariam de se aprofundar no tema e funcionalidades:

<https://selenium-python.readthedocs.io/>

<https://pypi.org/project/selenium/>

<https://www.selenium.dev/documentation/en/>



Selenium – webdriver (1/3)

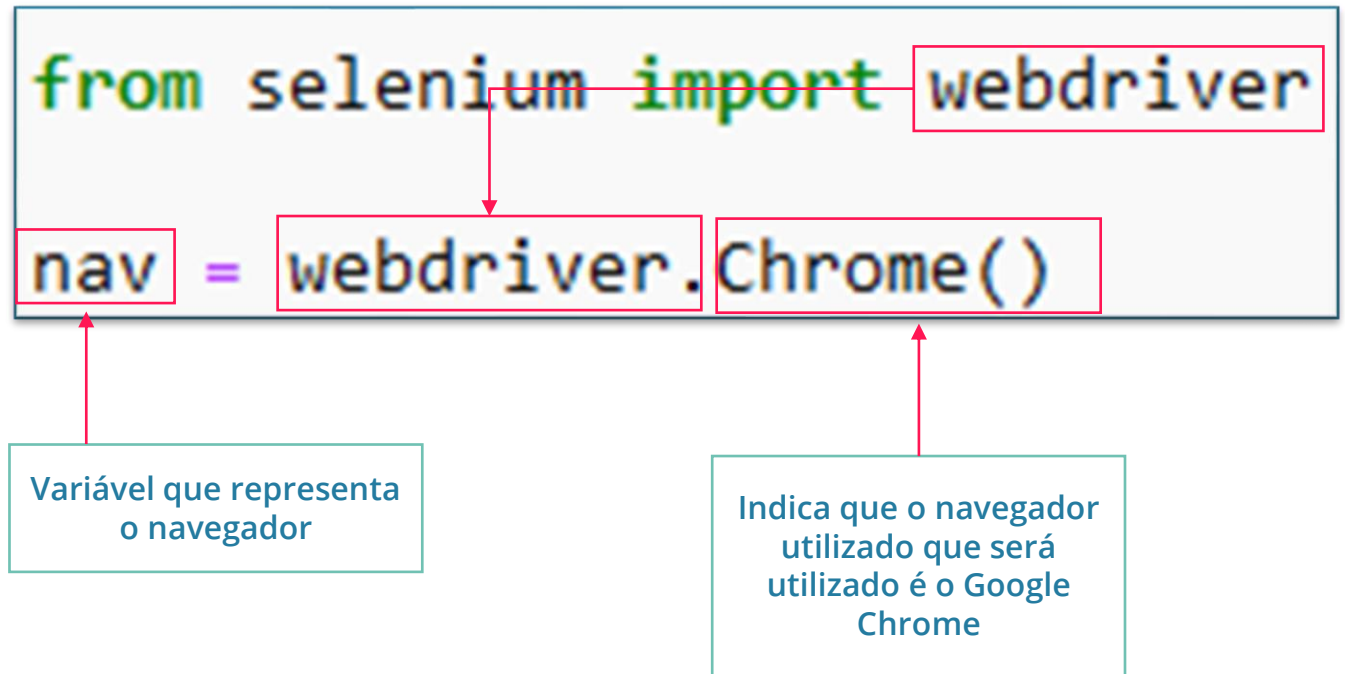
Vamos para nosso código. Devemos criar um código que nos permita acessar o site, e buscar os dados da nossa base.

Se você se lembra do nosso passo anterior, vai perceber que estamos usando o webdriver que importamos anteriormente.

O webdriver possui uma particularidade. Ele precisa ser baixado, extraído e colocado dentro da **MESMA pasta** que o Python possui seu executável. O próximo slide será só sobre isso, então não se preocupe.

Voltando para nosso código, podemos perceber que criamos uma variável **navegador**. Ela nos permitirá interagir com os diferentes elementos que encontraremos nas páginas WEB.

Outro ponto é que vemos **.Chrome()**. Isso nos indica que acessaremos essas páginas via Google Chrome.



Interface com uma página na web

Selenium – webdriver (2/3)

Antes de mais nada precisamos baixar o ChromeDriver, e o ChromeDriver precisa ser da mesma versão que o Chrome instalado no seu computador.

Passo 1: Abra o Chrome e acesse o endereço `chrome://settings/help`

Passo 2: Veja a versão do seu Google Chrome. Só os primeiros dígitos antes do primeiro ponto importam. Então, já que o navegador do exemplo é versão 97.0.4692.71, vamos baixar qualquer ChromeDriver que seja versão 97.

Passo 3: Acesse o [link](#) para baixar o driver

Passo 4: Escolha a versão mais próxima da sua versão, todas elas servem, mas versões mais próximas apresentam maior compatibilidade.

Passo 5: Escolha seu sistema operacional. No nosso caso Windows. Não existe chromedriver 64bits para Windows, então usaremos o win32.

The screenshot shows the Google Chrome interface. At the top, it says "Google Chrome". Below that, a status bar indicates "O Chrome está atualizado" (Chrome is updated) with a checkmark icon. The version "Versão 97.0.4692.71 (Versão oficial) 64 bits" is displayed and highlighted with a red box. Below the status bar, there are links for "Ajuda com o Chrome" and "Informar um problema".

Below the Chrome interface, the ChromeDriver download page is shown. The version "ChromeDriver 97.0.4692.71" is highlighted with a red box. The page lists the supported Chrome version (97) and provides a link to the release page. Below this, the version "ChromeDriver 97.0.4692.36" is listed, followed by another link to the release page. At the bottom, the version "ChromeDriver 97.0.4692.20" is listed.

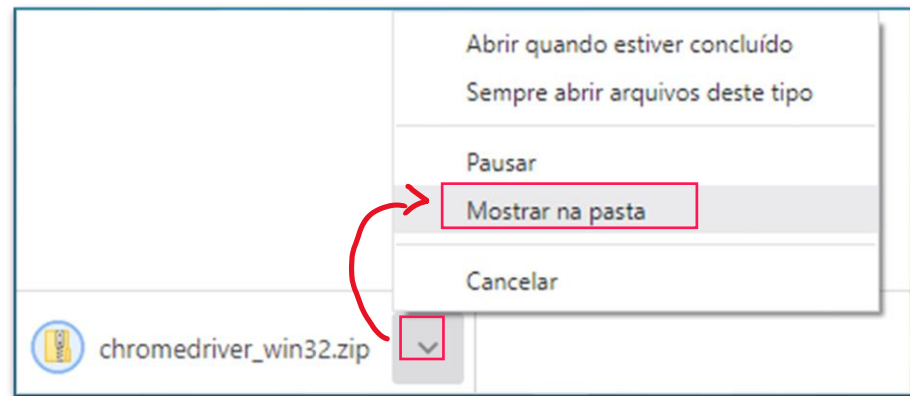
To the right of the ChromeDriver version list, a file explorer view shows the contents of the download directory. The files listed are:

File Name	Size	Modified
Parent Directory	-	-
chromedriver_linux64.zip	9.52MB	2022-01-05 05:45:10
chromedriver_mac64.zip	7.89MB	2022-01-05 05:45:13
chromedriver_mac64_m1.zip	7.48MB	2022-01-05 05:45:15
chromedriver_win32.zip	5.89MB	2022-01-05 05:45:17
notes.txt	0.00MB	2022-01-05 05:45:22

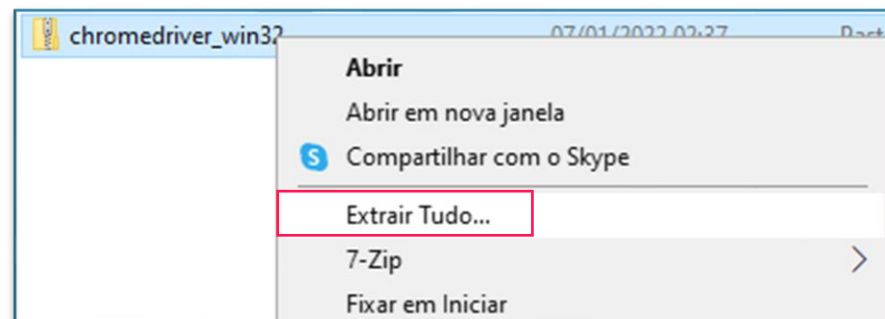
Interface com uma página na web

Selenium – webdriver (3/3)

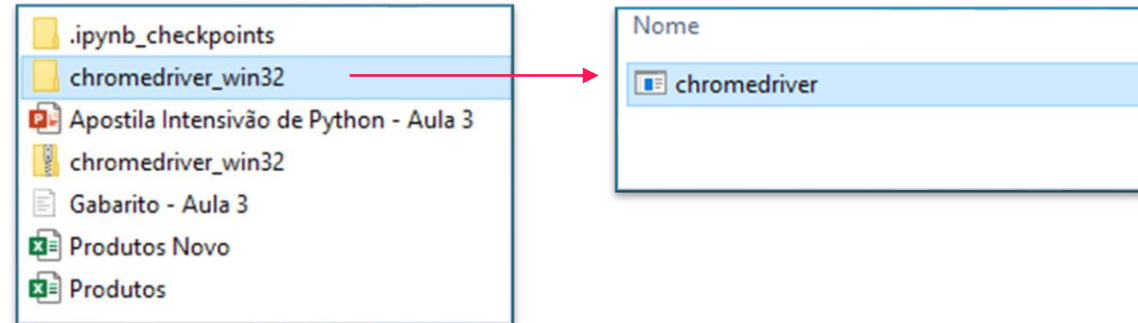
Passo 6: Abra o local do arquivo baixado.



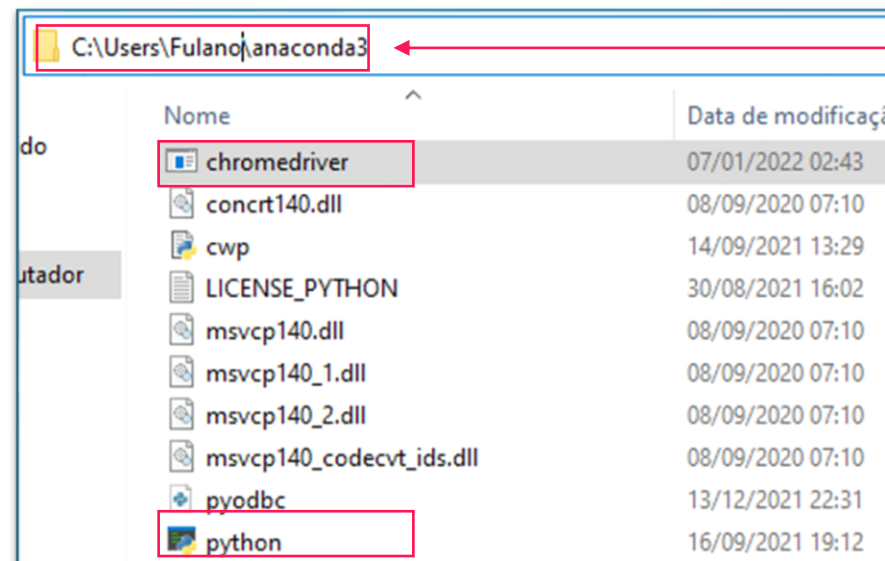
Passo 7: Clique com o botão direito no arquivo baixado e em seguida clique em Extrair Tudo... Abrirá uma nova janela. Clique em Extrair.



Passo 8: Será criada uma pasta que contém o arquivo chromedriver.exe



Passo 9: Coloque o arquivo **chromedriver.exe** na mesma pasta que o arquivo **python.exe**



Pasta que contém o arquivo python.exe. ATENÇÃO!!! Este caminho pode variar!!!

Selenium – webdriver - IMPORTANTE

É muito importante que você siga os passos anteriores de forma correta, pois se a sua versão do ChromDriver for diferente da versão do seu navegador o código vai dar um erro nessa parte inicial e não vai abrir o navegador.

```
from selenium import webdriver  
  
nav = webdriver.Chrome()  
nav.get("https://google.com")
```

Se essa parte inicial do código não funcionar provavelmente é porque a versão do ChromeDriver não está compatível com o seu navegador. Se estiver tudo certo, nesse primeiro bloco de código você vai notar que o Chrome vai abrir e vai entrar no site do Google!

Se isso acontecer você instalou corretamente o ChromeDriver e o restante do código vai funcionar, pois precisamos utilizar o ChromeDriver para acessar o Chrome e buscar as informações.

Parte 4

Importando a planilha Excel

Importando a planilha via Pandas

Assim como vimos nas demais aulas do Intensivão de Python, usaremos o Pandas para a importação da nossa base Excel.

Você já deve saber como essa importação é feita, mas nesse caso vamos utilizar o `pd.read_excel`, pois agora temos um arquivo de fato em Excel ao invés de um arquivo com a extensão csv.

Veja que ao importar a tabela temos duas colunas vazias, mas não vamos excluir essas colunas, pois precisamos inserir o preço atual de cada produto e verificar se vamos fazer ou não a compra de cada um desses itens.

```
import pandas as pd

tabela = pd.read_excel("commodities.xlsx")
display(tabela)
```

	Produto	Preço Ideal	Preço Atual	Comprar
0	Milho	85.32	NaN	NaN
1	Soja	163.59	NaN	NaN
2	Boi	282.20	NaN	NaN
3	Petróleo	424.37	NaN	NaN
4	Algodão	497.76	NaN	NaN
5	Açúcar	136.23	NaN	NaN
6	Café	1092.87	NaN	NaN
7	Ouro	321.77	NaN	NaN
8	Trigo	1549.11	NaN	NaN
9	Tilápia	9.05	NaN	NaN

Parte 5

Buscando as cotações na Web

Cotações das Commodities

Agora que já importamos nossas bibliotecas, vamos começar a utilizar o Selenium para acessarmos os sites que nos fornecerão as cotações que precisamos.

Essencialmente o que faremos será escrever códigos que reproduzam o que nós mesmos faríamos se estivéssemos realizando essa tarefa manualmente.

O passo a passo seria entrar no site onde temos a cotação do produto que será o <https://www.melhorcambio.com> e buscar pelo produto desejado.

Feito isso vamos poder pegar o valor e armazenar dentro da nossa planilha.

Em seguida temos que repetir todos esses passos para os demais produtos da nossa tabela.

Só que como isso pode ser um processo chato e demorado, nós vamos automatizar, até para deixá-lo mais rápido e eficiente evitando até uma análise errada por conta de algum descuido.

Buscando as cotações na Web

Cotações das Commodities

```
# import unicodedata
# novo_texto = unicodedata.normalize('NFKD', produto).encode('ascii', 'ignore')

for linha in tabela.index:
    produto = tabela.loc[linha, "Produto"]
    produto = produto.replace("ã", "a").replace(
        "ç", "c").replace("á", "a").replace("ó", "o").replace("ú", "u").replace("é", "e")
    link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
    print(link)
    nav.get(link)
    preco = nav.find_element("xpath", '//*[@id="comercial"]').get_attribute("value")
    preco = preco.replace(".", "").replace(",", ".")
    print(preco)
    tabela.loc[linha, "Preço Atual"] = float(preco)

display(tabela)
```

Aqui nós vamos utilizar a estrutura de repetição que vimos nas outras aulas para percorrer todas as linhas do `tabela.index`, que nada mais é do que o índice que temos na nossa tabela (são aqueles números que ficam a esquerda da tabela, começando do 0).

Em seguida nós vamos fazer um pequeno tratamento de dados nos nomes dos nossos produtos, isso é para que possamos acessar o site de cada um dos produtos de forma correta.

Qual o motivo desse tratamento? Se você observar o site onde vamos buscar o preço do milho por exemplo é assim:

<https://www.melhorcambio.com/milho-hoje>

Temos o nome padrão do site + / + nome do produto + -hoje

Isso nos traz o valor do milho no dia atual, isso quer dizer que precisamos tratar os nomes dos nossos produtos para que a busca seja feita de forma correta.

No produto Algodão por exemplo, não funcionaria o site escrito dessa forma:

<https://www.melhorcambio.com/Algodão-hoje>

O site vai retornar um erro porque não encontrou essa informação, então teríamos que ajustar de **Algodão** para **algodao**.

<https://www.melhorcambio.com/algodao-hoje>

Aqui você já consegue acessar o site normalmente.

Cotações das Commodities

```
# import unicodedata
# novo_texto = unicodedata.normalize('NFKD', produto).encode('ascii', 'ignore')

for linha in tabela.index:
    produto = tabela.loc[linha, "Produto"]
    produto = produto.replace("ã", "a").replace(
        "ç", "c").replace("é", "e").replace("ó", "o").replace("ú", "u").replace("ê", "e")
    link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
    print(link)
    nav.get(link)
    preco = nav.find_element("xpath", '//*[@id="comercial"]').get_attribute("value")
    preco = preco.replace(".", "").replace(",", ".")
    print(preco)
    tabela.loc[linha, "Preço Atual"] = float(preco)

display(tabela)
```

Então esse tratamento inicial logo depois da estrutura de repetição é para substituir os caracteres especiais que temos por caracteres normais.

Isso vai fazer com que o site possa ser acessado de forma correta e sem apresentar nenhum tipo de erro.

Feito o tratamento nós temos o seguinte código:

```
link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
```

Não se preocupe que é fácil de entender!

Primeiro temos um **f** antes de começar a escrever o nosso texto. Qual o motivo disso? Esse é chamado de fstring e serve para que você consiga colocar uma variável dentro do seu texto sem que tenha que “fatiar” seu texto para isso.

Então temos o **{produto.lower()}** dentro da composição do nosso link.

O produto é a variável que estamos utilizando no momento, que nada mais é do que o produto na primeira linha analisado (que é o Milho).

Já o **.lower** é um método para colocar todos os caracteres em minúsculo para manter o padrão do site.

Com isso vamos poder criar o site correto de cada um dos produtos para obter o preço de cada um deles!

Buscando as cotações na Web

Cotações das Commodities

```
# import unicodedata
# novo_texto = unicodedata.normalize('NFKD', produto).encode('ascii', 'ignore')

for linha in tabela.index:
    produto = tabela.loc[linha, "Produto"]
    produto = produto.replace("ã", "a").replace(
        "ç", "c").replace("é", "e").replace("ó", "o").replace("ú", "u").replace("ê", "e")
    link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
    print(link)
    nav.get(link)
    preco = nav.find_element("xpath", '//*[@id="comercial"]').get_attribute("value")
    preco = preco.replace(".", "").replace(",", ".")
    print(preco)
    tabela.loc[linha, "Preço Atual"] = float(preco)

display(tabela)
```

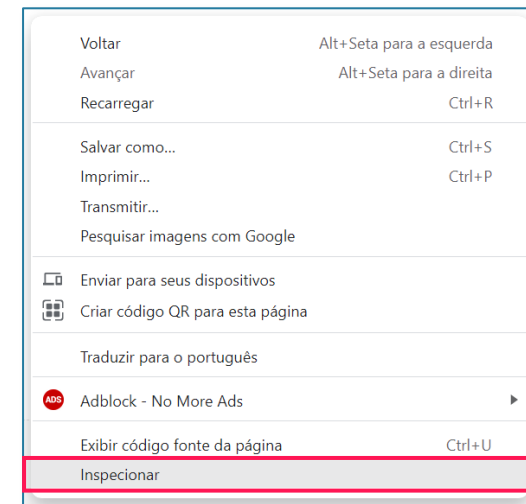
Com o link criado vamos fazer o print do link para que você possa acessá-lo depois e confirmar o valor se precisar.

Agora a partir do ponto mostrado na imagem, nós vamos fazer uma busca dentro do seu navegador usando o Python.

Aqui vamos buscar pelo **xpath**, que nada mais é do que a posição de um elemento em uma página.

E como você vai conseguir essa informação? É fácil, eu vou te mostrar um passo a passo que vai ter que fazer na página que vamos buscar a cotação.

Pode ser qualquer uma das páginas de cotação, pois todas possuem o mesmo padrão.

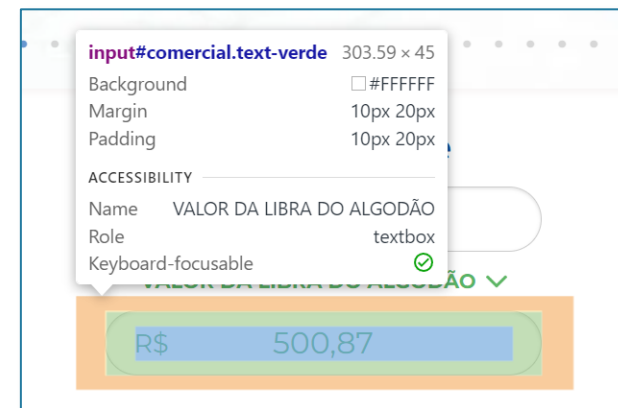
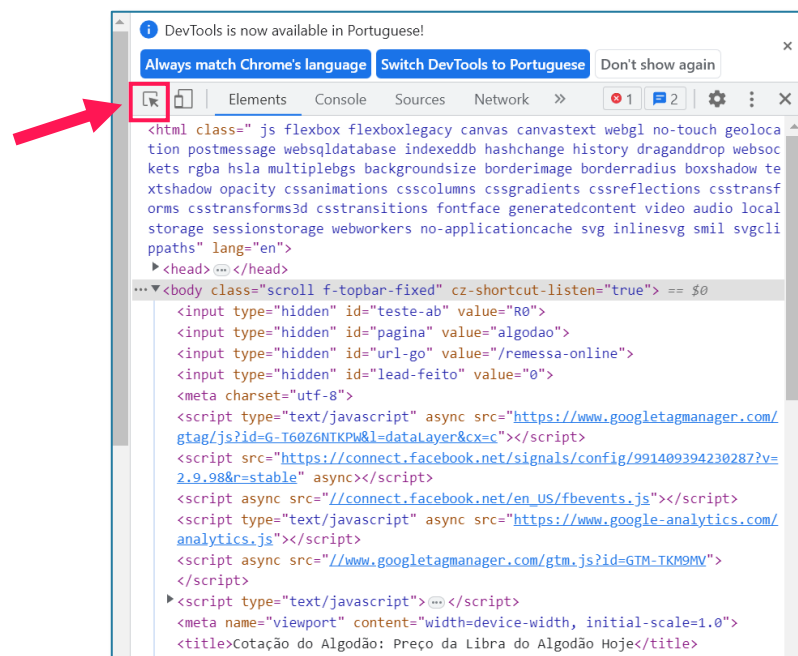


Você vai clicar em qualquer lugar da página com o botão direito do mouse e vai selecionar a opção **Inspecionar**, ou pode utilizar o **atalho F12**.

Buscando as cotações na Web

Cotações das Commodities

Feito isso você vai visualizar uma janela diferente dentro do seu navegador, ou à esquerda ou na parte inferior.



Com isso você vai estar selecionando o elemento onde é armazenado o valor da cotação dentro da página.

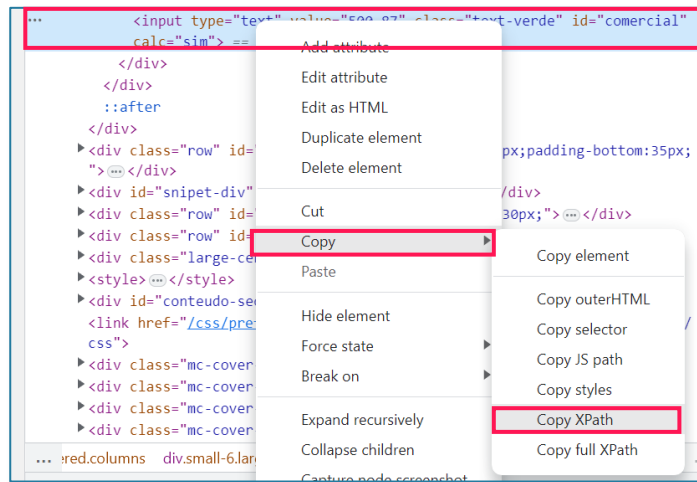
No canto superior esquerdo dessa janela você vai clicar no símbolo mostrado na figura acima.

Feito isso você vai passar o mouse por cima de onde temos o preço e depois vai clicar nessa informação.

Buscando as cotações na Web

Cotações das Commodities

Você vai notar que onde tinha aquelas informações de código que abriu quando clicou em inspecionar agora está selecionada.



Agora que tem a informação de preço realçada no código, você pode clicar com o botão direito, ir até **Copy** e depois **Copy XPath**.

Agora você tem a posição do elemento que mostra o preço dentro do site para que o Python possa acessar essa informação.

Note que essa informação será parecida com isso:

```
//*[@id="comercial"]
```

Essa é a informação que vamos colocar no nosso código! Veja que nessa linha de código estamos fazendo uma busca por um elemento (**.find_element**) e estamos informando que é o xpath.

Logo em seguida colocamos **.get_atribute("value")** que significa que estamos pegando o valor desse elemento.

```
# import unicodedata
# novo_texto = unicodedata.normalize('NFKD', produto).encode('ascii', 'ignore')

for linha in tabela.index:
    produto = tabela.loc[linha, "Produto"]
    produto = produto.replace("ã", "a").replace(
        "ç", "c").replace("á", "a").replace("ó", "o").replace("ú", "u").replace("é", "e")
    link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
    print(link)
    nav.get(link)
    preco = nav.find_element("xpath", '//*[@id="comercial"]').get_atribute("value")
    preco = preco.replace(".", "").replace(",", ".")
    print(preco)
    tabela.loc[linha, "Preço Atual"] = float(preco)

display(tabela)
```

```
<input type="text" value="500,87" class="text-verde" id="comercial"
calc="sim"> == $0
```

Buscando as cotações na Web

Cotações das Commodities

Para finalizar a parte de coleta de informações vamos apenas ajustar as casas decimais e o separador de milhar para deixar no formato americano para que possamos fazer as comparações dentro do Python.

```
# import unicodedata
# novo_texto = unicodedata.normalize('NFKD', produto).encode('ascii', 'ignore')

for linha in tabela.index:
    produto = tabela.loc[linha, "Produto"]
    produto = produto.replace("ã", "a").replace(
        "ç", "c").replace("á", "a").replace("ó", "o").replace("ú", "u").replace("é", "e")
    link = f"https://www.melhorcambio.com/{produto.lower()}-hoje"
    print(link)
    nav.get(link)
    preco = nav.find_element("xpath", '//*[@id="comercial"]').get_attribute("value")
    preco = preco.replace(".", "").replace(",", ".")
    print(preco)
    tabela.loc[linha, "Preço Atual"] = float(preco)

display(tabela)
```

Depois desse processo vamos “printar” o valor para o usuário e vamos adicionar essa informação dentro da tabela, na coluna de “Preço Atual” que até então estava vazia.

Veja que já estamos atribuindo com o **float**, ou seja, já estamos informando que esse é um valor decimal.

```
https://www.melhorcambio.com/milho-hoje
85.42
https://www.melhorcambio.com/soja-hoje
162.29
https://www.melhorcambio.com/boi-hoje
276.25
https://www.melhorcambio.com/petroleo-hoje
396.23
https://www.melhorcambio.com/algodao-hoje
500.87
https://www.melhorcambio.com/acucar-hoje
133.54
https://www.melhorcambio.com/cafe-hoje
1106.96
https://www.melhorcambio.com/ouro-hoje
327.19
https://www.melhorcambio.com/trigo-hoje
1549.23
https://www.melhorcambio.com/tilapia-hoje
9.05
```

	Produto	Preço Ideal	Preço Atual	Comprar
0	Milho	85.32	85.42	NaN
1	Soja	163.59	162.29	NaN
2	Boi	282.20	276.25	NaN
3	Petróleo	424.37	396.23	NaN
4	Algodão	497.76	500.87	NaN
5	Açúcar	136.23	133.54	NaN
6	Café	1092.87	1106.96	NaN
7	Ouro	321.77	327.19	NaN
8	Trigo	1549.11	1549.23	NaN
9	Tilápia	9.05	9.05	NaN

Parte 6

Criando uma planilha no Excel

Criando a planilha com os dados atualizados

Você vai notar que isso tudo vai acontecer forma bem rápida e o navegador que estava aberto vai passar em todos os links. Um para cada produto que temos na tabela.

Se você quisesse obter apenas os preços atuais desses produtos poderíamos finalizar o código aqui.

Só que o nosso objetivo é fazer uma comparação entre o **Preço Ideal** e o **Preço Atual** para saber se devemos ou não comprar esses produtos.

```
nav.quit()
tabela["Comprar"] = tabela["Preço Ideal"] > tabela["Preço Atual"]
tabela.to_excel("commodities_atualizado.xlsx", index=False)
```

Inicialmente vamos fechar o navegador, pois não precisar mais dele e em seguida vamos começar a preencher as informações da coluna **Comprar**.

Como isso vai ser feito? Vamos verificar se o Preço Ideal é menor do que o Preço Atual.

Como não estamos colocando mais nada, o nosso resultado será simplesmente VERDADEIRO (TRUE) ou FALSO (FALSE).

Isso quer dizer que se o preço atual for menor do que o preço ideal da tabela nós vamos querer comprar esse produto.






Com a tabela totalmente preenchida nós vamos utilizar o método **.to_excel** para transformar essas informações em um arquivo do Excel.

Basta inserir o nome que vamos querer no arquivo que ele será criado.

OBS: O **index=False** é apenas para que não seja levado para o Excel a coluna de índices que mostra o número das colunas.

Buscando as cotações na Web

Cotações das Commodities

 .ipynb_checkpoints	15/03/2023 09:02
 Apostila Intensivão de Python - Aula 3.pptx	15/03/2023 08:37
 commodities.xlsx	15/03/2023 08:36
 commodities_atualizado.xlsx	15/03/2023 10:45
 Projeto3.ipynb	15/03/2023 10:46

Agora você tem um arquivo novo de Excel além dos arquivos que já tinha na sua pasta com as informações atualizadas para verificar se deve comprar ou não esses produtos.

	A	B	C	D
1	Produto	Preço Ideal	Preço Atual	Comprar
2	Milho	85,32	85,42	FALSO
3	Soja	163,59	162,29	VERDADEIRO
4	Boi	282,2	276,25	VERDADEIRO
5	Petróleo	424,37	396,23	VERDADEIRO
6	Algodão	497,76	500,87	FALSO
7	Açúcar	136,23	133,54	VERDADEIRO
8	Café	1092,87	1106,96	FALSO
9	Ouro	321,77	327,19	FALSO
10	Trigo	1549,11	1549,23	FALSO
11	Tilápia	9,05	9,05	FALSO

Com essa tabela concluída você pode repassar para o seu time de compras e informar que pode comprar apenas os produtos com a informação **VERDADEIRA** na coluna comprar.

Os demais produtos você pode aguardar e fazer a verificação no dia seguinte para ver se será um bom negócio fazer a compra!

Com o código pronto você pode rodar ele de uma só vez e verificar quanto tempo leva para fazer todo esse processo.

Depois faça de forma manual para comparar o tempo que você levou e que o código levou para fazer isso.

Você vai notar que são poucos produtos, mas você levaria muito mais tempo mesmo com poucos produtos!

Agora imagine se você tem uma lista com 100, 200, 1.000 produtos, vai dizer que não facilitaria muito a sua vida usar esse tipo de automação?

INTENSIVÃO DE PYTHON {#}

100% ONLINE & GRATUITO

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!



@hashtagprogramacao



youtube.com/hashtag-programacao

