# Como coletar dados sobre opções da bolsa de valores automaticamente

Muitos investidores consultam sites na Internet para analisar empresas ou acompanhar informações que os auxiliam na tomada de decisão sobre realizar ou não um determinado investimento. Para o mercado de derivativos, mais especificamente de opções, há alguns sites que oferecem dados sobre as opções disponíveis no mercado, sua liquidez diária de negociação, preços de exercício (strike), datas de vencimento, dentre outras informações.

Todos esses dados são muito importantes porém ferramentas mais especializadas tendem a ser pagas e investidores iniciantes podem não estar dispostos a realizar esse tipo de desembolso mensal, ou mesmo não tem condições financeiras para tanto.

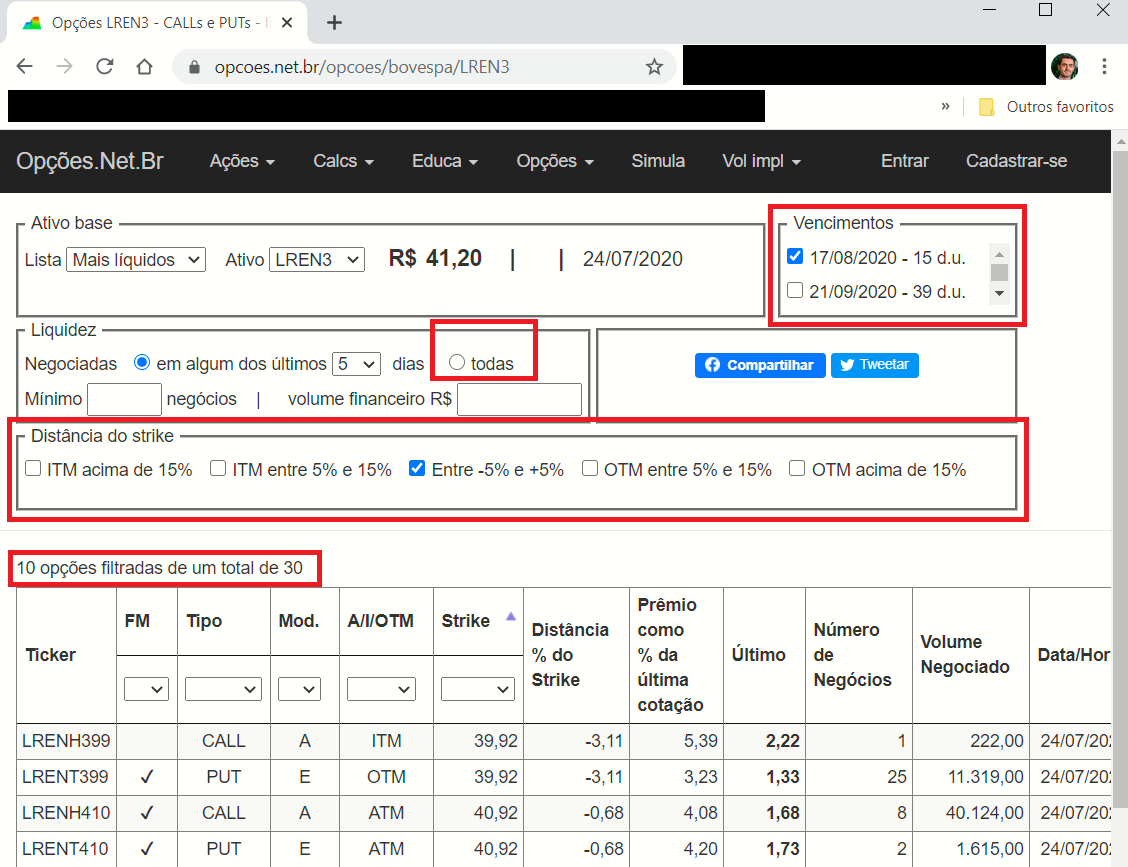
Dentre os sites mencionados anteriormente, há alguns grátis, o que já é uma vantagem, porém eles não permitem consulta das informações em tempo real. Assim, desenvolvi um programa que coleta todos os códigos de opções disponíveis no webiste <https://opcoes.net.br/> de modo automático e consolida as informações em um único arquivo. Os códigos das opções podem depois ser copiados para uma planilha de análise do investidor que possua link de dados com alguma plataforma, tal como o Tryd, disponibilizado gratuitamente pela corretora Modal Mais. Com os códigos das opções e o link de dados podemos acompanhar o comportamento das opções em tempo real e melhor, sem custo!

## Pré-requisitos

Antes de entrarmos na parte técnica, é preciso que você tenha instalado no seu sistema o python 3, o Chromedriver e os pacotes selenium, pandas e openpyxl. A instalação de todos esses componentes de software pode ser encontrada em tutoriais detalhados que já havia previamente escrito. Os artigos são:

## O website Opcoes.net.br

Para realizar a coleta de dados, precisamos primeiramente entender como funciona o site. Na imagem abaixo temos uma visão geral de quais dados são apresentados para a ação LREN3 (Lojas Renner).



Logo que acessamos a página, alguns filtros já estão selecionados, e conforme o retângulo vermelho mais para baixo na página, são apresentadas apenas 10 opções de um total de 30. Queremos que o site nos mostre todas as opções.

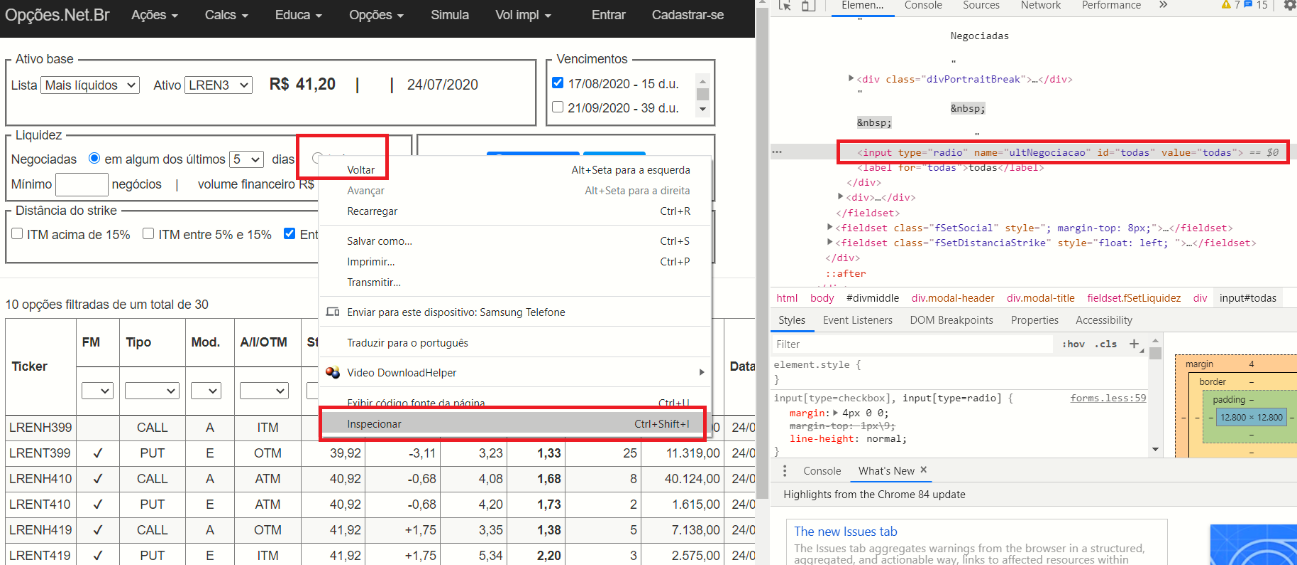
No retêngulo vermelho no canto direito da imagem estão os **Vencimentos.**Cada vencimento possui um pequeno quadrado clicável, que corresponde a um elemento html do tipo **checkbox**. O mesmo ocorre para os filtros de **Distância do strike**com 5 checkboxes. E, por fim, temos um último elemento html chamado **radiobox,**uma pequena bolinha clicável ao lado do texto "todas". Para mostrar todas as opções precisamos deixar todos esses elementos clicados.

O procedimento manual seria acessar a página, clicar em todos os elementos, copiar os códigos das opções e colar para uma planilha excel. Fazer isso para uma opção não é grande problema, mas imagine fazer isso para dezenas de ações, ou tlavez centenas. Não é nada agradável. Por isso automatizar esse procedimento nos faz ganhar tempo, muito tempo.

## Identificando os elementos HTML

Olhamos a página no browser e descobrimos quais os elementos que precisamos clicar. Porém, como dizer a um programa qual quadradinho (checkbox) ele deve clicar?

No browser Chrome, por exemplo, você pode posicionar o cursor do mouse sobre o checkbox, clicar com o botão direito e selecionar a opção "Inspecionar". Isso faz a página HTML do site abrir e assim é possível observar como cada um desses elementos é referenciado dentro da página HTML.



Apenas para exemplificar, na imagem acima clicamos com o botão direito sobre o radiobutton "todas". Na parte à direita da imagem aparece qual o trecho HTML da página corresponde à descrição daquele elemento. O trecho de interesse é:

<input type="radio" name="ultNegociacao" id="todas" value="todas">

Ou seja, é um elemento do tipo radiobutton, e seu identificador (id) é "todas". Esse mesmo passo a passo é realizado com todos os chebkoxes de interesse, coletando o id de cada elemento.

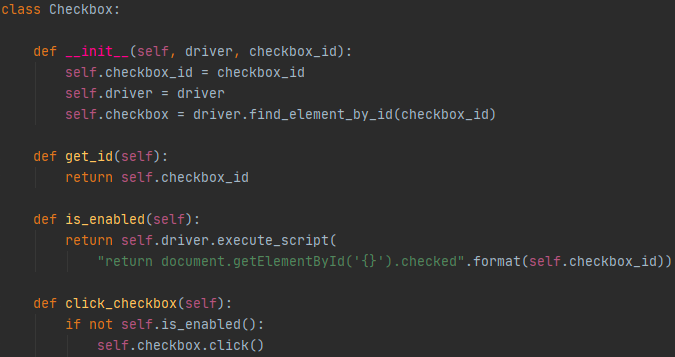
## Como clicar em Checkboxes e Radiobuttons?

Para clicar nesses elementos de modo programável utilizamos o pacote Selenium com o Chromedriver, que nos dá acesso a uma versão simplificada do browser Chrome para acessar o website.

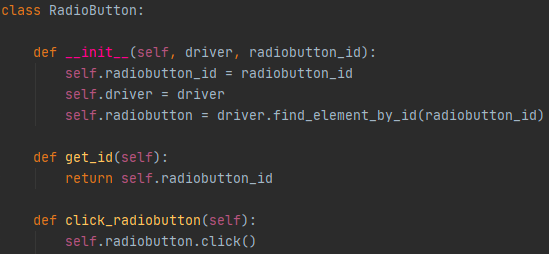
from selenium import webdriver

driver = webdriver.Chrome() # Cria nosso driver

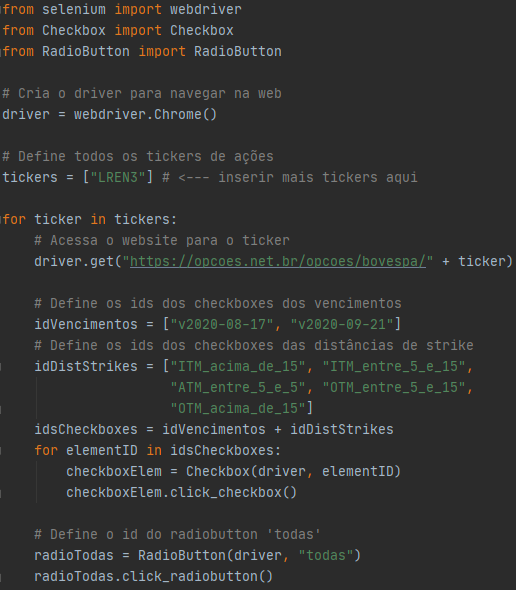
Para checkboxes a classe a seguir nos permite clicar no botão, caso o botão ainda não tenha sido clicado. O teste de botão clicado é importante, pois se um checkbox já clicado for novamente clicado ele fica desabilitado.



Para radiobuttons a classe a seguir permite clicar no elemento. Não é necessário o teste de botão clicado, pois mesmo clicando em um radiobutton já clicado ele vai seguir clicado.



Então, juntando tudo, devemos criar o driver, acessar o website e clicar em todos os elementos escolhidos.



Depois de executar esse trecho de código, o resultado esperado é ter todos os botões escolhidos clicados e mais opções disponíveis para coleta, conforme na imagem a seguir:



## Como coletar os dados da página HTML?

Aqui as coisas já começam a ficar mais interessantes. Depois de disponibilizar para visualização todas as opções dos vencimentos escolhidos podemos iniciar a coleta de dados.

Uma estrutura de dados muito útil é o **DataFrame**que permite o armazenamento de informações em uma estrutura tabular, semelhante a uma tabela de SQL e possui diversos métodos para tratamento e exportação de dados que usaremos mais a frente. Essa estrutura de dados é disponibilizada pelo pacote **pandas**do python.

Analisando novamente a página do site, podemos perceber que todas as opções estão dentro de uma tabela que possui id **tblListaOpc**. Dentro dessa tabela, cada opção está em uma linha diferente, ou seja, podemos acessar cada opção pelo elemento <tr> da tabela.

tableOp = driver.find\_element\_by\_id("tblListaOpc")

rows = tableOp.find\_elements(By.TAG\_NAME, "tr")

Com as opções carregadas na página, utilizamos o comando **find\_element\_by\_id()**que retorna a tabela. Dentro da tabela fazemos uma busca pela tag HTML <tr> e obtemos uma lista de elementos, as linhas da tabela.

Aplicando o método .text para uma das linhas da tabela, temos o seguinte resultado:

print(rows[0].text)

> LRENU336 PUT E OTM 33,61 -18,42 1,46 0,60 1 120,00 24/07/2020

Olhando o site nós sabemos qual a estrutura da tabela e também o que significa cada um dos valores impressos na tela. Para a saída acima temos:

* **Ticker da opção**: LRENU336
* **Tipo:** PUT (Indica de é opção de venda)
* **Estilo ou Modalidade:**E (Indica que é europeia)
* **A/I/OTM:**OTM (Out of the money)
* **Strike**: 33,61

As demais informações não nos interessam. Também conseguimos saber qual a série da opção, representada pelo quinto caractere do ticker da opção, que neste caso é U. Cada mês de vencimento possui uma letra distinta para Calls e Puts, conforme a tabela a seguir:



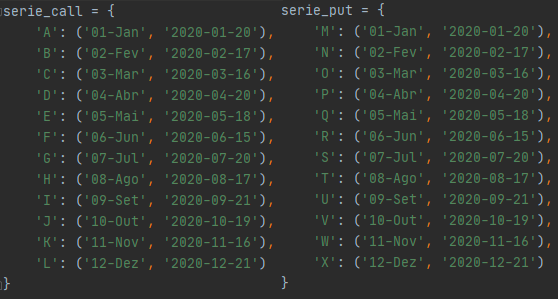
Logo, por ser uma opção Put (opção de venda) de série U, sabemos que a data de vencimento da opção é dia 21/09/2020.

A coleta das informações de cada linha da tabela pode ser feita com o script abaixo:

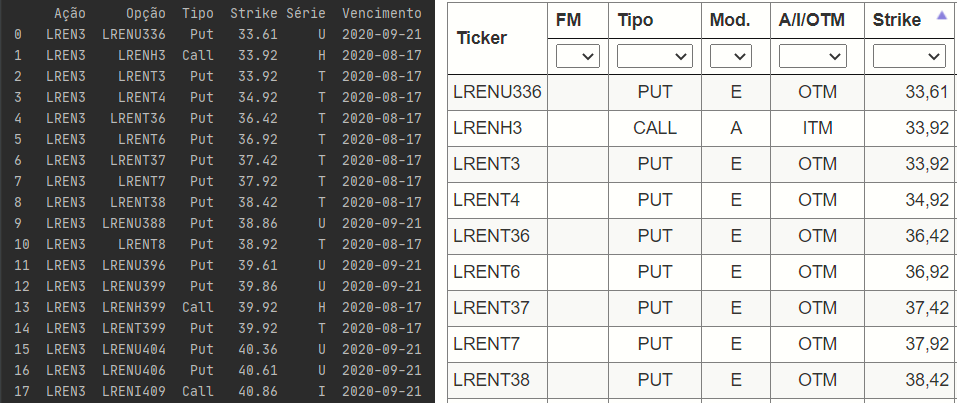


A função **coletaDados()** recebe o driver e o ticker da ação e identifica a tabela, coleta as linhas e ignora o cabeçalho, mantendo apenas as linhas com as opções na variável **rows**. Depois, para cada linha (**row**), são buscadas as tags <td>, que correspondem às colunas da tabela.

Os dados armazenados no DataFrame são o ticker da ação, o ticker da opção, o strike, o tipo da opção, a série e o vencimento. A data de vencimento é armazenada no DataFrame a partir de um dicionário previamente configurado no programa:

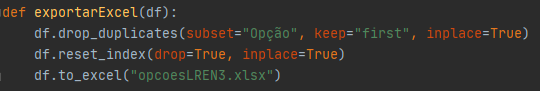


Logo, com a série é possível ter acesso à data de vencimento e armazenar essa informação no DataFrame. O resultado dessa busca de opções de LREN3 é o DataFrame abaixo que foi impresso no console (imagem à esquerda) e ao lado está um corte do webiste para comparação.

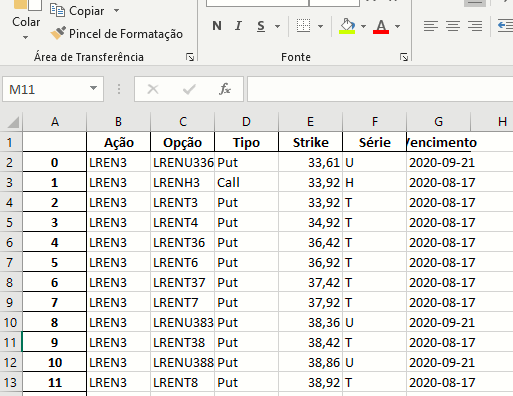


## Exportando os dados para um arquivo .xlsx

DataFrames possuem um método que nos permite exportar os dados para um arquivo de Excel chamado **.to\_excel().**A função **exportarExcel()**confere se há algum dado repretido no DataFrame e depois exporta os dados para um arquivo de extensão .xlsx.



A planilha criada fica com a seguinte cara:



## Considerações Finais

Este projeto mostrou como coletar dados da Internet utilizando python, selenium, Chromedriver e pandas. Depois de realizada a coleta de dados um arquivo .xlsx foi criado, o qual pode ser utilizado para análises do investidor em uma planilha que possua link de dados com a bolsa de valores, fornecendo dados em tempo real sobre os tickers das opções.

O leitor não precisa se limitar à busca de opções de LREN3. Com pequenas modificações no código é possível coletar as opções de todas as ações que o site disponibiliza.

Infelizmente não consegui colocar neste artigo os trechos de código mais longos com cores e identado que pudessem ser copiados pelo leitor. Para ficar "bonito" a única alternativa que encontrei foi colar as imagens dos códigos. Mas para facilitar a vida do leitor disponibilizei uma versão totalmente funcional de todo o código apresentado aqui no link <https://github.com/jlggross/StockOptionsWebScrapper/blob/master/article.py>.

Além disso o racional de funcionamento desse script está inserido em um projeto que desenvolvi para a coleta de dados sobre opções de diferentes websites, que depois são agregados e limpos, mantendo apenas um registro para cada opção.

O projeto completo pode ser encontrado na minha página do github <https://github.com/jlggross/StockOptionsWebScrapper>. Aproveite para ler a descrição do projeto no diretório raiz, pois está bem detalhado e serve como aprendizado.