# You

I am trying to use selenium-stealth driver. Following is the example code: from selenium import webdriver from selenium\_stealth import stealth import time

options = webdriver.ChromeOptions() options.add\_argument("start-maximized")

# options.add\_argument("--headless")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"]) options.add\_experimental\_option('useAutomationExtension', False) driver = webdriver.Chrome(options=options, executable\_path=r"D:\chromedriver.exe")

stealth(driver, languages=["en-US", "en"], vendor="Google Inc.", platform="Win32", webgl\_vendor="Intel Inc.", renderer="Intel Iris OpenGL Engine", fix\_hairline=True, )

url = "https://bot.sannysoft.com/" driver.get(url) time.sleep(5) driver.quit()

But when I execute:

driver = webdriver.Chrome(options=options, executable\_path=r"D:\chromedriver.exe")

I see the error:

> driver = webdriver.Chrome(options=options, executable\_path=r"D:\chromedriver.exe")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument 'executable\_path'

# Copilot

The error message is indicating that the webdriver.Chrome method does not recognize executable\_path as a valid argument.

In Selenium WebDriver, the executable\_path is not a keyword argument for the webdriver.Chrome class. Instead, it's a positional argument.

You should use it like this:

driver = webdriver.Chrome(r"D:\chromedriver.exe", options=options)

Here, the path to the ChromeDriver is the first argument and the options are the second argument. This should resolve the error you're seeing. Let me know if you have any other questions! 😊

# You

>>> driver = webdriver.Chrome(r"D:\chromedriver.exe", options=options)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: WebDriver.\_\_init\_\_() got multiple values for argument 'options'

# Copilot

I apologize for the confusion. The error message indicates that the options argument is being provided more than once, which is not allowed.

The webdriver.Chrome method automatically checks certain locations for the ChromeDriver executable, so you typically don't need to provide the executable\_path argument if the ChromeDriver is installed in one of these locations.

If you have installed ChromeDriver in a location that is not automatically checked, you can add that location to your system's PATH environment variable.

However, if you still want to use the executable\_path argument, you can try the following:

from selenium.webdriver.chrome.service import Service

# ...

driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options)

In this code, the Service class is used to specify the path to the ChromeDriver executable. This should resolve the error you're seeing. Let me know if you have any other questions! 😊

# You

In the following code:

from selenium import webdriver from selenium\_stealth import stealth import time

options = webdriver.ChromeOptions() options.add\_argument("start-maximized")

# options.add\_argument("--headless")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"]) options.add\_experimental\_option('useAutomationExtension', False) driver = webdriver.Chrome(options=options) stealth(driver, languages=["en-US", "en"], vendor="Google Inc.", platform="Win32", webgl\_vendor="Intel Inc.", renderer="Intel Iris OpenGL Engine", fix\_hairline=True, ) url = "https://in.investing.com/equities/cadila-healthcare-historical-data?end\_date=1710959400&st\_date=1483209000" driver.get(url)

After last command extend it further and change suitably to use the following portion of the code:

soup = BeautifulSoup(resp.content, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

# Get the data using list comprehension

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text.

for tr in tbody.find\_all('tr')

]

# Combine headers and data

result = [headers] + data

return result

# Copilot

Sure, I can help you integrate the BeautifulSoup code with your Selenium code. Here's how you can do it:

from selenium import webdriver

from selenium\_stealth import stealth

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

options = webdriver.ChromeOptions()

options.add\_argument("start-maximized")

# options.add\_argument("--headless")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

options.add\_experimental\_option('useAutomationExtension', False)

driver = webdriver.Chrome(options=options)

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

url = "https://in.investing.com/equities/cadila-healthcare-historical-data?end\_date=1710959400&st\_date=1483209000"

driver.get(url)

time.sleep(5) # Allow the page to load

# Now use BeautifulSoup to parse the page source

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

# Get the data using list comprehension

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]

for tr in tbody.find\_all('tr')

]

# Combine headers and data

result = [headers] + data

driver.quit()

print(result)

In this code, we first navigate to the webpage using Selenium. Then, we use BeautifulSoup to parse the page source that Selenium retrieves. This allows us to use BeautifulSoup's parsing capabilities on webpages that require JavaScript to display their content.

Please note that you might need to adjust the time.sleep(5) line. This line is necessary to ensure that the webpage has fully loaded before we try to parse it. The exact amount of time to wait might vary depending on your internet speed and the specific webpage.

Let me know if you have any other questions! 😊

Source(s)

1. [github.com](https://github.com/rocketpy/selenium_tutorial/tree/02d201223c43a04100d75ff061a833de1854b258/Selenium_stealth_mode.py)

2. [github.com](https://github.com/rehmatworks/dnb/tree/582714af008f6b1283d2381b230860b9fc46f92a/companies%2Fmanagement%2Fcommands%2Ffetchcompanies.py)

3. [github.com](https://github.com/pgshow/7tenderScraper/tree/445a2d529e40ea3a0298e33e8188da0503bc89fc/selenium_stealth%2Ftests%2Fchrome_headless_with_stealth_test.py)

4. [github.com](https://github.com/donwayo/ebayKleinanzeigen/tree/ea7904eb00ea2f616d92de4e8d9d077129ca1dde/kleinanzeigen.py)

5. [github.com](https://github.com/International-Persuasion-Machines/cogsec_2021_presentation/tree/fb139093633bf07ed27878786b775a3ac0f017a9/selenium_stealth_version.py)

# You

I have a selenium code as under which works:

from selenium import webdriver from selenium\_stealth import stealth from bs4 import BeautifulSoup import time from datetime import datetime as dt import re import xlwings as xw import re # Regular expression library

@xw.func def get\_historical\_data(script\_name, end\_date, start\_date): end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp()) start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp()) options = webdriver.ChromeOptions() options.add\_argument("start-maximized") options.add\_experimental\_option("excludeSwitches", ["enable-automation"]) options.add\_experimental\_option('useAutomationExtension', False) driver = webdriver.Chrome(options=options) url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}" stealth(driver, languages=["en-US", "en"], vendor="Google Inc.", platform="Win32", webgl\_vendor="Intel Inc.", renderer="Intel Iris OpenGL Engine", fix\_hairline=True, ) driver.get(url) time.sleep(2) soup = BeautifulSoup(driver.page\_source, 'html.parser') div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'}) table = div.find('table', {'class': 'common-table medium js-table'}) colgroup = table.find('colgroup') headers = [col.get('class')[0] for col in colgroup.find\_all('col')] tbody = table.find('tbody') data = [ [ dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip() for i, td in enumerate(tr.find\_all('td')) ]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text. for tr in tbody.find\_all('tr') ] result = [headers] + data driver.quit() return result

I want to define chromedriver.exe located at d:\ and chrome binary located at: C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application

Can that be done by changing options?

# Copilot

Yes, you can specify the path to the ChromeDriver executable and the Chrome binary in your Selenium options. Here's how you can do it:

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium\_stealth import stealth

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = webdriver.ChromeOptions()

options.add\_argument("start-maximized")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

options.add\_experimental\_option('useAutomationExtension', False)

options.binary\_location = r"C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe"

driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

driver.get(url)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text.

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

In this code, options.binary\_location is used to specify the path to the Chrome binary, and webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options) is used to specify the path to the ChromeDriver executable.

Please replace "C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe" and "D:\chromedriver.exe" with the actual paths to your Chrome binary and ChromeDriver executable, respectively.

Let me know if you have any other questions! 😊

Source(s)

1. [github.com](https://github.com/byom26/Web-Scrapping/tree/d26a0638c5ec5bbb76cea96701d914534f7a34a2/fiverrProjects%2Fproject-9%2FspainWedPlanner.py)

# You

Is there a way to avoid advertisments and other junk from loading while webpage is being loaded?

# Copilot

Yes, there are a few ways to avoid loading advertisements and other unwanted elements when using Selenium:

1. **Use an ad-blocking extension**: You can add an ad-blocking extension to your Selenium browser instance. This will block most advertisements from loading. However, this method might not be effective against all types of ads, especially those that are loaded dynamically or embedded directly into the webpage's content.
2. **Modify the hosts file**: The hosts file is a computer file used by an operating system to map hostnames to IP addresses. By adding the addresses of known ad servers to your hosts file and mapping them to the local address (127.0.0.1), you can prevent your browser from connecting to these servers and thus block ads. However, this method requires administrative privileges and affects all browsers on your computer, not just the Selenium browser instance.
3. **Use a WebDriver with built-in ad-blocking**: Some WebDrivers, like uBlock Origin, have built-in ad-blocking capabilities. These can be more effective than using an ad-blocking extension, but they might not be available for all browsers.
4. **Disable images and JavaScript**: You can disable images and JavaScript in your Selenium browser instance to prevent ads from loading. However, this will also affect the functionality of the webpage and might not be suitable for all use cases.

Here's an example of how to use an ad-blocking extension with Selenium:

from selenium import webdriver

options = webdriver.ChromeOptions()

options.add\_extension('path\_to\_adblock\_extension.crx')

driver = webdriver.Chrome(options=options)

In this code, replace 'path\_to\_adblock\_extension.crx' with the path to the .crx file of your ad-blocking extension.

Please note that using ad-blocking methods might violate the terms of service of some websites. Always make sure to respect the website's rules and policies when web scraping. 😊

# You

If my userdata in chrome is at:

C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data

Can you tell me where that extension is found?

# Copilot

Chrome extensions are typically stored in the Default/Extensions folder under the Chrome User Data directory. So, in your case, the path to the extensions would be:

C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data\Default\Extensions

Each extension will have its own folder within the Extensions directory, named with the extension's ID. Inside each extension's folder, there will be one or more version folders, containing the actual files for the extension.

Please note that the extension's ID is a long string of letters, and it's used in the URL when you navigate to the extension's details page on the Chrome Web Store.

However, the .crx file of an extension (which is what you would need to load the extension into a Selenium WebDriver) is not stored in this location. The .crx file is essentially a packaged and compressed version of the extension's files, and it's what you download from the Chrome Web Store when you install an extension. Unfortunately, Chrome does not store the .crx file once the extension is installed.

If you need the .crx file of an extension for use with Selenium, you would typically need to download it from the Chrome Web Store. There are various online tools and tutorials available that can guide you through the process of downloading a .crx file from the Chrome Web Store. Please make sure to follow all relevant terms of service and laws when using such tools. 😊

# You

When I define extension like this: options.add\_extension('path\_to\_adblock\_extension.crx')

Everytime when browser opens, it installs itself again and again. Is there any workaround?

# Copilot

Yes, there is a workaround for this. Instead of adding the extension every time you launch the browser, you can create a new Chrome profile and install the extension there. Then, you can instruct Selenium to use this profile every time it launches Chrome. Here's how you can do it:

1. **Create a new Chrome profile**: Open Chrome, click on the profile icon in the top right corner, and then click on "Add". Give the new profile a name, and then click on "Add". A new Chrome window will open with this profile.
2. **Install the extension in the new profile**: In the new Chrome window, go to the Chrome Web Store, find your extension, and install it.
3. **Find the profile's directory**: In the new Chrome window, type chrome://version in the address bar and press Enter. Look for the "Profile Path" field and note down the directory path.
4. **Use the profile in Selenium**: When launching Chrome with Selenium, use the user-data-dir and profile-directory Chrome options to specify the profile's directory. Here's an example:

from selenium import webdriver

options = webdriver.ChromeOptions()

options.add\_argument("--user-data-dir=C:/path/to/user/data/dir") # replace with the path to your User Data directory

options.add\_argument("--profile-directory=Profile 1") # replace with your profile directory

driver = webdriver.Chrome(options=options)

In this code, replace "C:/path/to/user/data/dir" with the path to your User Data directory (the parent directory of the profile's directory), and replace "Profile 1" with your profile directory (the last part of the profile's directory path).

Now, every time Selenium launches Chrome, it will use the specified profile, and the extension will already be installed.

Please note that the paths to the User Data directory and the profile directory might be different on your system. Make sure to replace them with the correct paths. 😊

# You

But that can even be done in my existing profile. Why create new one?

# Copilot

You're correct. You can certainly use your existing profile. The reason to create a new profile is to keep your testing environment separate from your personal browsing environment.

Here are a few reasons why you might want to use a separate profile for Selenium automation:

1. **Isolation**: Using a separate profile for automation can help prevent your personal browsing data (like your browsing history, bookmarks, and saved passwords) from affecting your tests. It also prevents your tests from modifying your personal data.
2. **Performance**: If you have a lot of extensions installed in your personal profile, they could slow down your Selenium tests. Using a separate profile allows you to only install the extensions that are needed for your tests.
3. **Consistency**: Tests can sometimes fail due to changes in the browser's state that are made during manual browsing. Using a separate profile for automation ensures that the browser is in a consistent state each time the tests are run.
4. **Convenience**: If the testing profile gets cluttered with data or if an extension causes issues, you can simply delete the profile and create a new one, without affecting your personal browsing profile.

However, if these considerations are not relevant for your use case, you can certainly use your existing profile. 😊

# You

What to do when I am using default profile for this command: options.add\_argument("--profile-directory=Profile 1")

# Copilot

If you're using the default profile, you don't need to specify the --profile-directory argument. The default profile is used automatically when you don't specify a profile directory.

However, if you still want to specify the default profile explicitly, you can do so by setting --profile-directory to "Default", like this:

options.add\_argument("--profile-directory=Default")

This tells Chrome to use the default profile in the User Data directory. Please note that the profile directory name is case-sensitive, so make sure to capitalize the "D" in "Default". 😊

# You

My final code after above change (full code):

from selenium import webdriver from selenium.webdriver.chrome.service import Service from selenium\_stealth import stealth from bs4 import BeautifulSoup import time from datetime import datetime as dt import re import xlwings as xw import re # Regular expression library

@xw.func def get\_historical\_data(script\_name, end\_date, start\_date): end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp()) start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp()) options = webdriver.ChromeOptions() options.add\_argument("--user-data-dir=C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data") # replace with the path to your User Data director options.add\_argument("--profile-directory=Profile 2") # replace with your profile directory

# options.add\_extension(r'D:\dev\AdBlock-—-best-ad-blocker.crx')

options.add\_argument("start-maximized")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

options.add\_experimental\_option('useAutomationExtension', False)

options.binary\_location = r"C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe"

driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

driver.get(url)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text.

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

I get traceback when function is called from excel:

pythoncom error: Python error invoking COM method.

Traceback (most recent call last): File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke* return self.*invoke*(dispid, lcid, wFlags, args) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke* return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex* return func(\*args) ^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 502, in call\_udf ret = func(\*args) ^^^^^^^^^^^ File "d:\dev\selenium\_stealth\_google\_canary.py", line 23, in get\_historical\_data driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

# Copilot

The error message indicates that there's an issue when the get\_historical\_data function is being called from Excel via xlwings. This could be due to a number of reasons, but it's often related to the fact that Selenium requires a graphical user interface (GUI) to operate, and Excel macros typically run in a non-GUI environment.

Here are a few things you can try:

1. **Run Excel as an Administrator**: Sometimes, running Excel with elevated privileges can resolve issues with running Selenium scripts from Excel.
2. **Use a Virtual Display**: If you're running this on a server or other headless environment, you might need to use a virtual display to provide a GUI for Selenium. Python packages like PyVirtualDisplay can help with this.
3. **Debug the Function Independently**: Try running the get\_historical\_data function independently in a Python environment to see if it works as expected. If it does, the issue is likely with how Excel is interacting with Python, not with the function itself.
4. **Check Your Python and xlwings Setup**: Make sure that you've set up Python and xlwings correctly to work with Excel. The xlwings documentation has a good guide on how to do this.

If none of these suggestions help, could you provide more details about your setup? For example, what version of Excel are you using? Are you running this on a local machine or a server? Do other xlwings functions work correctly? The more information you provide, the better I can help you troubleshoot this issue. 😊

# You

Soem problem in these alternate profile related. When I comment them, the UDF works:

# options.add\_argument("--user-data-dir=C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data") # replace with the path to your User Data director

# options.add\_argument("--profile-directory=Profile 2") # replace with your profile directory

# Copilot

I see. It seems like the issue is related to the Chrome profile. Here are a few things you could check:

1. **Profile Path**: Ensure that the profile path you're providing is correct. The --user-data-dir should point to the directory containing the 'User Data', and --profile-directory should point to the specific profile folder (like 'Profile 2') within the 'User Data' directory.
2. **Profile Existence**: Make sure that the profile you're trying to use actually exists. You can check this by navigating to the path in your file explorer. If the profile doesn't exist, Chrome might fail to launch.
3. **Permissions**: Ensure that the user running the script has the necessary permissions to access and modify the specified profile directory.
4. **Profile In Use**: If the profile is being used in another Chrome instance, it might cause issues. Ensure that all other Chrome windows using that profile are closed.

If you continue to face issues, you might want to consider using the default profile or creating a new profile specifically for your Selenium tasks. 😊

# You

What does this command do: options.add\_experimental\_option('useAutomationExtension', True) - I changed the argument from False to True

# Copilot

The options.add\_experimental\_option('useAutomationExtension', True) command enables the automation extension in Chrome when Selenium is controlling the browser.

The automation extension is a built-in component of Chrome that provides additional capabilities for automating the browser, such as interacting with various browser APIs. By default, when Selenium launches Chrome, this extension is enabled.

However, some websites use the presence of this extension to detect automated browsing and may behave differently or even block access when they detect it. To avoid this, some users disable the automation extension by setting 'useAutomationExtension' to False.

By changing 'useAutomationExtension' to True, you're re-enabling this extension. This means that the additional capabilities provided by the extension will be available, but you may be more likely to be detected as an automated browser by some websites. 😊

# You

As you know after adding new profile and setting up, I got following traceback:

Traceback (most recent call last): File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke* return self.*invoke*(dispid, lcid, wFlags, args) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke* return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex* return func(\*args) ^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 502, in call\_udf ret = func(\*args) ^^^^^^^^^^^ File "d:\dev\selenium\_stealth\_google\_canary.py", line 24, in get\_historical\_data driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\chrome\webdriver.py", line 45, in **init** super().**init**( File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\chromium\webdriver.py", line 61, in **init** super().**init**(command\_executor=executor, options=options) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 208, in **init** self.start\_session(capabilities) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 292, in start\_session response = self.execute(Command.NEW\_SESSION, caps)["value"] ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 347, in execute self.error\_handler.check\_response(response) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 229, in check\_response raise exception\_class(message, screen, stacktrace) selenium.common.exceptions.SessionNotCreatedException: Message: session not created: Chrome failed to start: exited normally. (chrome not reachable) (The process started from chrome location C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe is no longer running, so ChromeDriver is assuming that Chrome has crashed.)

But there is this work around: Cookies To avoid the login process you have two options, the first one involves saving cookies in a file and loading it when necessary.

url = 'https://www.microsoft.com' driver.get(url)

# manually enter username and password

import pickle pickle.dump(driver.get\_cookies(), open("cookies.pkl","wb")) driver.quit() then restart the webdriver and run

driver.get(url) cookies = pickle.load(open("cookies.pkl", "rb")) for cookie in cookies: driver.add\_cookie(cookie) driver.refresh()

suggested here:

https://stackoverflow.com/questions/75549263/avoid-login-each-time-i-run-script-in-selenium-python/75589648#75589648

Can you check and see whether by changing our above code we can get this worksround to work?

# Copilot

Yes, you can certainly incorporate the cookie-saving workaround into your existing code. Here's how you can modify your function to save and load cookies:

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium\_stealth import stealth

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

import pickle

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = webdriver.ChromeOptions()

options.add\_argument("--user-data-dir=C:\\Users\\baksh\\AppData\\Local\\Google\\Chrome SxS\\User Data") # replace with the path to your User Data director

options.add\_argument("--profile-directory=Profile 2") # replace with your profile directory

options.add\_argument("start-maximized")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

options.add\_experimental\_option('useAutomationExtension', False)

options.binary\_location = r"C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe"

driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

# Load cookies if they exist

try:

cookies = pickle.load(open("cookies.pkl", "rb"))

for cookie in cookies:

driver.add\_cookie(cookie)

driver.refresh()

except (FileNotFoundError, EOFError):

pass

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text.

for tr in tbody.find\_all('tr')

]

result = [headers] + data

# Save cookies

pickle.dump(driver.get\_cookies(), open("cookies.pkl","wb"))

driver.quit()

return result

In this code, we're loading the cookies from the cookies.pkl file (if it exists) after navigating to the URL, and then refreshing the page to apply the cookies. At the end of the function, we're saving the cookies from the current session to the cookies.pkl file.

This will allow you to maintain your login session across multiple runs of your function, as long as the cookies are still valid. Please note that cookies have an expiration date, so you may still need to log in manually from time to time.

Let me know if you have any other questions! 😊

# You

With these options enabled Not commented by #):

# options.add\_argument("user-data-dir=C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data") # replace with the path to your User Data director

# options.add\_argument("profile-directory=Profile 3") # replace with your profile directory

Chrome crashes. Can you search whether this a bug with selenium or google chrome? IS there any solution found by anyone to this traceback (as above)

# Copilot

I found some relevant information on StackOverflow that might help you resolve the issue:

1. **Check the Profile Path**: Ensure that the profile path you're providing is correct. The --user-data-dir should point to the directory containing the 'User Data', and --profile-directory should point to the specific profile folder (like 'Profile 3') within the 'User Data' directory[**1**](https://stackoverflow.com/questions/53599079/selenium-chromedriver-unable-to-set-user-data-dir).
2. **Use Absolute Paths**: Some users have reported that using absolute paths for the --user-data-dir option resolved their issues[**2**](https://stackoverflow.com/questions/67126246/user-data-dir-not-working-anymore-on-selenium-with-chromedriver-90).
3. **Set Remote Debugging Port**: Another user suggested setting the remote debugging port as a workaround for a similar issue[**2**](https://stackoverflow.com/questions/67126246/user-data-dir-not-working-anymore-on-selenium-with-chromedriver-90). You can do this by adding the following line to your code:

options.add\_argument("--remote-debugging-port=9222")

Please try these suggestions and see if they resolve your issue. If not, please let me know so I can assist you further. 😊

Source(s)

1. [Selenium ChromeDriver: unable to set user-data-dir](https://stackoverflow.com/questions/53599079/selenium-chromedriver-unable-to-set-user-data-dir)

2. [user-data-dir not working anymore on selenium with chromedriver 90](https://stackoverflow.com/questions/67126246/user-data-dir-not-working-anymore-on-selenium-with-chromedriver-90)

3. [Selenium Web Driver with user-data-dir set to crash in Chrome ... - Posts](https://linuxtut.com/selenium-web-driver-with-user-data-dir-set-to-crash-in-chrome-headless-mode-c692d/)

4. [How to use Chrome Profile in Selenium Webdriver Python 3](https://stackoverflow.com/questions/52394408/how-to-use-chrome-profile-in-selenium-webdriver-python-3)

5. <https://mail.google.com/mail/u/0/>

# You

See this part of the traceback (FYI - I could not get it to work with: options.add\_argument("--remote-debugging-port=9222")

File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 229, in check\_response raise exception\_class(message, screen, stacktrace) selenium.common.exceptions.SessionNotCreatedException: Message: session not created: Chrome failed to start: exited normally. (chrome not reachable) (The process started from chrome location C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe is no longer running, so ChromeDriver is assuming that Chrome has crashed.)

Full traceback is:

pythoncom error: Python error invoking COM method.

Traceback (most recent call last): File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke* return self.*invoke*(dispid, lcid, wFlags, args) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke* return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex* return func(\*args) ^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 502, in call\_udf ret = func(\*args) ^^^^^^^^^^^ File "d:\dev\selenium\_stealth\_google\_canary.py", line 25, in get\_historical\_data driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\chrome\webdriver.py", line 45, in **init** super().**init**( File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\chromium\webdriver.py", line 61, in **init** super().**init**(command\_executor=executor, options=options) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 208, in **init** self.start\_session(capabilities) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 292, in start\_session response = self.execute(Command.NEW\_SESSION, caps)["value"] ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\webdriver.py", line 347, in execute self.error\_handler.check\_response(response) File "D:\Programs\Python\Lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 229, in check\_response raise exception\_class(message, screen, stacktrace) selenium.common.exceptions.SessionNotCreatedException: Message: session not created: Chrome failed to start: exited normally. (chrome not reachable) (The process started from chrome location C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe is no longer running, so ChromeDriver is assuming that Chrome has crashed.)

Do you get any clue?

# Copilot

The error message selenium.common.exceptions.SessionNotCreatedException: Message: session not created: Chrome failed to start: exited normally typically indicates that Selenium is unable to start a new session because it cannot launch the Chrome browser. This could be due to several reasons:

1. **ChromeDriver and Chrome Browser Version Mismatch**: Ensure that your ChromeDriver version is compatible with your installed version of the Chrome browser[**1**](https://stackoverflow.com/questions/57092707/selenium-common-exceptions-sessionnotcreatedexception-message-session-not-crea)[**5**](https://blog.csdn.net/weixin_63067252/article/details/134408400).
2. **Incorrect Path to Chrome Binary or ChromeDriver**: Make sure that the paths to the Chrome binary and the ChromeDriver executable are correct[**1**](https://stackoverflow.com/questions/57092707/selenium-common-exceptions-sessionnotcreatedexception-message-session-not-crea).
3. **Chrome Profile Issues**: If you're using a specific Chrome profile, make sure that the profile exists and is accessible[**1**](https://stackoverflow.com/questions/57092707/selenium-common-exceptions-sessionnotcreatedexception-message-session-not-crea).
4. **Chrome Processes Still Running**: If there are any Chrome processes still running from previous sessions, they might interfere with starting a new session[**4**](https://stackoverflow.com/questions/53073411/selenium-webdriverexceptionchrome-failed-to-start-crashed-as-google-chrome-is). You can try closing all running Chrome processes before starting a new session.
5. **Chrome Sandbox Issues**: If you're running Selenium in a sandboxed or containerized environment, you might need to add the --no-sandbox option[**2**](https://groups.google.com/g/robotframework-users/c/4iyQGFQHEx4).
6. **Remote Debugging Port Issues**: If you're using a remote debugging port, make sure that the port is not being used by another process[**2**](https://groups.google.com/g/robotframework-users/c/4iyQGFQHEx4).

If none of these suggestions help, could you provide more details about your setup? For example, what versions of Chrome and ChromeDriver are you using? Are you running this on a local machine or a server? The more information you provide, the better I can help you troubleshoot this issue. 😊

Source(s)

1. [selenium.common.exceptions.SessionNotCreatedException: Message: session ...](https://stackoverflow.com/questions/57092707/selenium-common-exceptions-sessionnotcreatedexception-message-session-not-crea)

2. [Getting error with Chrome browser in selenium library - Google Groups](https://groups.google.com/g/robotframework-users/c/4iyQGFQHEx4)

3. [SessionNotCreatedException: Message: session not created from ...](https://stackoverflow.com/questions/53808653/sessionnotcreatedexception-message-session-not-created-from-disconnected-unab)

4. [python - Selenium: WebDriverException:Chrome failed to start: crashed ...](https://stackoverflow.com/questions/53073411/selenium-webdriverexceptionchrome-failed-to-start-crashed-as-google-chrome-is)

5. [Linux调用Selenium报session not created: Chrome failed to start: exited ...](https://blog.csdn.net/weixin_63067252/article/details/134408400)

6. <http://google.com/>

7. <https://www.google.nl/>

8. <https://sites.google.com/chromium.org/driver/>

# You

But all the points you have stated above are not relevant because if I comment out like this:

# options.add\_argument("user-data-dir=C:\Users\baksh\AppData\Local\Google\Chrome SxS\User Data") # replace with the path to your User Data director

# options.add\_argument("profile-directory=Profile 3") # replace with your profile directory

Everything works well.

# Copilot

I see, if commenting out the lines related to the user data directory and profile directory resolves the issue, it suggests that the problem is indeed related to these settings.

It's possible that the profile directory "Profile 3" might not exist or there could be issues with access permissions. Alternatively, there could be a conflict with another instance of Chrome running with the same profile.

If you want to use a specific profile with Selenium, make sure that the profile exists and is not being used by another Chrome instance. If you continue to face issues, using the default profile or running without specifying a profile might be the best approach.

Remember, the main purpose of using a specific profile is to maintain certain settings or session data across multiple browsing sessions. If you don't require this functionality, running without a specific profile should not impact your Selenium tasks. 😊

# You

I found out. I just closed another instance of google chrome and it worked. With these two paramaters in tact (not commented out).

I think the issue as you said seems to be: "Alternatively, there could be a conflict with another instance of Chrome running with the same profile". But another instace is not of that profile. But since both the instances access same user data folder, that seems to be an issue.

Is there any workaround that the currently running instance of chrome remain in tact and the other instance with user data directory and another profile run in parallel?

# Copilot

Yes, you're correct. Even if the profiles are different, Chrome instances using the same User Data directory can conflict with each other. This is because the User Data directory contains information that is shared across profiles, such as browser settings and state.

One possible workaround is to create a copy of your User Data directory and use that for your Selenium tasks. Here's how you can do it:

1. **Copy the User Data directory**: You can create a copy of your User Data directory and place it in a different location. This will create a separate environment for your Selenium tasks that won't interfere with your normal browsing.
2. **Use the copied User Data directory in Selenium**: When launching Chrome with Selenium, use the --user-data-dir option to specify the path to the copied User Data directory.

Here's an example:

options = webdriver.ChromeOptions()

options.add\_argument("--user-data-dir=C:\\path\\to\\copied\\User Data") # replace with the path to the copied User Data directory

options.add\_argument("--profile-directory=Profile 3") # replace with your profile directory

In this code, replace "C:\\path\\to\\copied\\User Data" with the path to the copied User Data directory.

Please note that this will create a completely separate browsing environment, so any changes you make in one environment (like installing extensions or changing settings) won't affect the other. 😊

# You

Instead of following code using google chrome, can you give me similar code using firefox? from selenium import webdriver from selenium.webdriver.chrome.service import Service from selenium\_stealth import stealth from bs4 import BeautifulSoup import time from datetime import datetime as dt import re import xlwings as xw import re # Regular expression library import pickle

@xw.func def get\_historical\_data(script\_name, end\_date, start\_date): end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp()) start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp()) options = webdriver.ChromeOptions()

# options.add\_argument("--remote-debugging-port=9292")

# options.add\_argument("user-data-dir=D:\User Data") # replace with the path to your User Data director

# options.add\_argument("profile-directory=Default") # replace with your profile directory

options.add\_extension(r'D:\dev\AdBlock-—-best-ad-blocker.crx')

options.add\_argument("start-maximized")

options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

options.add\_experimental\_option('useAutomationExtension', False)

options.binary\_location = r"C:\Users\baksh\AppData\Local\Google\Chrome SxS\Application\chrome.exe"

driver = webdriver.Chrome(service=Service(r"D:\chromedriver.exe"), options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

# try:

# cookies = pickle.load(open("cookies.pkl", "rb"))

# for cookie in cookies:

# driver.add\_cookie(cookie)

# driver.refresh()

# except (FileNotFoundError, EOFError):

# pass

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]# .strftime("%d/%m/%Y") in dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") at the end deleted to avoid date being converted to text.

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

# Copilot

Sure, I can help you modify your code to use Firefox (GeckoDriver) instead of Chrome (ChromeDriver). Here's how you can do it:

from selenium import webdriver

from selenium\_stealth import stealth

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

import pickle

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = webdriver.FirefoxOptions()

options.add\_argument("-private") # Enable private browsing

driver = webdriver.Firefox(executable\_path=r"D:\geckodriver.exe", options=options) # Replace with the path to your GeckoDriver

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

stealth(driver,

languages=["en-US", "en"],

vendor="Google Inc.",

platform="Win32",

webgl\_vendor="Intel Inc.",

renderer="Intel Iris OpenGL Engine",

fix\_hairline=True,

)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

In this code, I've replaced webdriver.ChromeOptions() with webdriver.FirefoxOptions(), and webdriver.Chrome() with webdriver.Firefox(). I've also added the -private argument to enable private browsing in Firefox, which is similar to using a new profile in Chrome.

Please note that Selenium Stealth currently only supports Chrome, so the stealth() function may not work as expected with Firefox. Also, replace "D:\geckodriver.exe" with the path to your GeckoDriver executable.

Let me know if you have any other questions! 😊

# You

But there are no options like:

# options.add\_argument("user-data-dir=D:\User Data") # replace with the path to your User Data director

# options.add\_argument("profile-directory=Default") # replace with your profile directory

When I use firefox driver?

# Copilot

You're correct. Firefox handles profiles differently than Chrome. Instead of using command-line arguments, Firefox uses a ProfileManager to manage different profiles.

Here's an example of how you can use a specific profile with Firefox:

from selenium import webdriver

from selenium.webdriver.firefox.firefox\_profile import FirefoxProfile

profile = FirefoxProfile(r'C:\path\to\your\profile') # replace with the path to your Firefox profile

driver = webdriver.Firefox(firefox\_profile=profile, executable\_path=r"D:\geckodriver.exe") # replace with the path to your GeckoDriver

In this code, replace "C:\path\to\your\profile" with the path to your Firefox profile directory, and replace "D:\geckodriver.exe" with the path to your GeckoDriver executable.

Please note that you can find your Firefox profiles in the Firefox/Profiles/ directory within your Firefox user data directory. Each profile is stored in a separate folder within the Profiles directory.

Also, please be aware that running automated tasks on your default Firefox profile might interfere with your normal browsing. It's generally recommended to create a separate profile for automation tasks. 😊

# You

Imported functions from the following modules: selenium\_stealth\_google\_canary

DevTools listening on ws://127.0.0.1:61110/devtools/browser/2e989a52-851e-4d53-96a8-7288ea10c5c9 Created TensorFlow Lite XNNPACK delegate for CPU. pythoncom error: Python error invoking COM method.

Traceback (most recent call last): File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke* return self.*invoke*(dispid, lcid, wFlags, args) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke* return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex* return func(\*args) ^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 502, in call\_udf ret = func(\*args) ^^^^^^^^^^^ File "d:\dev\selenium\_stealth\_google\_canary.py", line 17, in get\_historical\_data driver = webdriver.Firefox(firefox\_profile=profile, executable\_path=r"D:\geckodriver.exe") ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ TypeError: WebDriver.**init**() got an unexpected keyword argument 'firefox\_profile'

# Copilot

The error message TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument 'firefox\_profile' suggests that the firefox\_profile argument is not recognized. This could be due to changes in the Selenium API.

In recent versions of Selenium, the firefox\_profile argument has been replaced with options. You can set the Firefox profile in the options like this:

from selenium import webdriver

from selenium.webdriver.firefox.options import Options

profile\_directory = r'C:\path\to\your\profile' # replace with the path to your Firefox profile

options = Options()

options.profile = profile\_directory

driver = webdriver.Firefox(options=options, executable\_path=r"D:\geckodriver.exe") # replace with the path to your GeckoDriver

In this code, replace "C:\path\to\your\profile" with the path to your Firefox profile directory, and replace "D:\geckodriver.exe" with the path to your GeckoDriver executable[**2**](https://stackoverflow.com/questions/76831949/webdriver-init-got-an-unexpected-keyword-argument-firefox-profile).

Please note that the Firefox profile directory is typically located in the Firefox/Profiles/ directory within your Firefox user data directory[**2**](https://stackoverflow.com/questions/76831949/webdriver-init-got-an-unexpected-keyword-argument-firefox-profile). Each profile is stored in a separate folder within the Profiles directory[**2**](https://stackoverflow.com/questions/76831949/webdriver-init-got-an-unexpected-keyword-argument-firefox-profile).

Let me know if you have any other questions! 😊

Source(s)

1. [How to load firefox profile with Python Selenium?](https://stackoverflow.com/questions/50321278/how-to-load-firefox-profile-with-python-selenium)

2. [WebDriver.\_\_init\_\_() got an unexpected keyword argument 'firefox\_profile'](https://stackoverflow.com/questions/76831949/webdriver-init-got-an-unexpected-keyword-argument-firefox-profile)

3. [TypeError: WebDriver.\_\_init\_\_ () got an unexpected keyword argument ...](https://stackoverflow.com/questions/74468866/typeerror-webdriver-init-got-an-unexpected-keyword-argument-service)

4. [TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument ...](https://stackoverflow.com/questions/70326002/typeerror-webdriver-init-got-an-unexpected-keyword-argument-firefox-opti)

5. [How to connect selenium grid with python | by Ethan Han | Medium](https://medium.com/@ethan.han.qa/how-to-connect-selenium-grid-with-python-35bb460803f4)

# You

This part does not seem to be correct:

profile\_directory = r'C:\path\to\your\profile' # replace with the path to your Firefox profile options = Options() options.profile = profile\_directory

driver = webdriver.Firefox(options=options, executable\_path=r"D:\geckodriver.exe")

Traceback:

pythoncom error: Python error invoking COM method.

Traceback (most recent call last): File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 305, in *Invoke* return self.*invoke*(dispid, lcid, wFlags, args) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 310, in *invoke* return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\win32com\server\policy.py", line 639, in *invokeex* return func(\*args) ^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py", line 235, in CallUDF res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller)) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\Programs\Python\Lib\site-packages\xlwings\udfs.py", line 502, in call\_udf ret = func(\*args) ^^^^^^^^^^^ File "d:\dev\selenium\_stealth\_google\_canary.py", line 17, in get\_historical\_data options = Options() ^^^^^^^ NameError: name 'Options' is not defined

# Copilot

The error NameError: name 'Options' is not defined suggests that the Options class is not being recognized. This is likely because the appropriate module has not been imported.

For Firefox, the Options class is part of the selenium.webdriver.firefox.options module. You can import it like this:

from selenium.webdriver.firefox.options import Options

Then you can use Options to create a new options object:

options = Options()

Here's how you can modify your code:

from selenium import webdriver

from selenium.webdriver.firefox.options import Options

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

import pickle

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = Options()

profile\_directory = r'C:\path\to\your\profile' # replace with the path to your Firefox profile

options.profile = profile\_directory

driver = webdriver.Firefox(options=options, executable\_path=r"D:\geckodriver.exe") # replace with the path to your GeckoDriver

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

In this code, replace "C:\path\to\your\profile" with the path to your Firefox profile directory, and replace "D:\geckodriver.exe" with the path to your GeckoDriver executable[**1**](https://stackoverflow.com/questions/65960797/how-to-address-options-is-not-defined-error-in-python-selenium)[**2**](https://stackoverflow.com/questions/46052056/the-constructor-firefoxdriverfirefoxoptions-is-undefined)[**3**](https://stackoverflow.com/questions/65923383/nameerror-name-webdriver-is-not-defined-error-using-selenium-and-python).

Let me know if you have any other questions! 😊

Source(s)

1. [How to address "Options is not defined error" in Python Selenium ...](https://stackoverflow.com/questions/65960797/how-to-address-options-is-not-defined-error-in-python-selenium)

2. [selenium webdriver - The constructor FirefoxDriver(FirefoxOptions) is ...](https://stackoverflow.com/questions/46052056/the-constructor-firefoxdriverfirefoxoptions-is-undefined)

3. [NameError: name 'webdriver' is not defined error using Selenium and ...](https://stackoverflow.com/questions/65923383/nameerror-name-webdriver-is-not-defined-error-using-selenium-and-python)

4. <https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/firefox/FirefoxDriver.html>

5. <http://www.google.com>

# You

driver = webdriver.Firefox(options=options, executable\_path=r"D:\geckodriver.exe")

raises traceback: TypeError: WebDriver.**init**() got an unexpected keyword argument 'executable\_path'

# Copilot

The error message TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument 'executable\_path' suggests that the executable\_path argument is not recognized. This is due to changes in Selenium 4.x[**1**](https://stackoverflow.com/questions/76802588/python-selenium-unexpected-keyword-argument-executable-path)[**2**](https://stackoverflow.com/questions/76550506/typeerror-webdriver-init-got-an-unexpected-keyword-argument-executable-p).

In Selenium 4.x, the executable\_path argument has been replaced with the service argument. You can specify the path to the GeckoDriver executable using the Service class from selenium.webdriver.firefox.service[**1**](https://stackoverflow.com/questions/76802588/python-selenium-unexpected-keyword-argument-executable-path)[**2**](https://stackoverflow.com/questions/76550506/typeerror-webdriver-init-got-an-unexpected-keyword-argument-executable-p).

Here's how you can modify your code:

from selenium import webdriver

from selenium.webdriver.firefox.options import Options

from selenium.webdriver.firefox.service import Service

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

import pickle

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = Options()

profile\_directory = r'C:\path\to\your\profile' # replace with the path to your Firefox profile

options.profile = profile\_directory

service = Service(r"D:\geckodriver.exe") # replace with the path to your GeckoDriver

driver = webdriver.Firefox(service=service, options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

In this code, replace "C:\path\to\your\profile" with the path to your Firefox profile directory, and replace "D:\geckodriver.exe" with the path to your GeckoDriver executable[**1**](https://stackoverflow.com/questions/76802588/python-selenium-unexpected-keyword-argument-executable-path)[**2**](https://stackoverflow.com/questions/76550506/typeerror-webdriver-init-got-an-unexpected-keyword-argument-executable-p).

Let me know if you have any other questions! 😊

Source(s)

1. [Python Selenium: 'unexpected keyword argument 'executable\_path'](https://stackoverflow.com/questions/76802588/python-selenium-unexpected-keyword-argument-executable-path)

2. [TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument ...](https://stackoverflow.com/questions/76550506/typeerror-webdriver-init-got-an-unexpected-keyword-argument-executable-p)

3. [Selenium TypeError: \_\_init\_\_ () got an unexpected keyword argument ...](https://stackoverflow.com/questions/70993717/selenium-typeerror-init-got-an-unexpected-keyword-argument-service)

4. [TypeError: WebDriver.\_\_init\_\_() got an unexpected keyword argument ...](https://stackoverflow.com/questions/74468866/typeerror-webdriver-init-got-an-unexpected-keyword-argument-service)

5. <https://example>

6. <https://stackoverflow.com/a/76550727/7058266>

7. <https://github.com/SeleniumHQ/selenium/commit/9f5801c82fb3be3d5850707c46c3f8176e3ccd8e>

8. <https://www.google.com/>

9. <https://www.facebook.com/>

# You

I think binary location also needs to be defined:

selenium.common.exceptions.SessionNotCreatedException: Message: Expected browser binary location, but unable to find binary in default location, no 'moz:firefoxOptions.binary' capability provided, and no binary flag set on the command line

# Copilot

You're correct. If Firefox is not installed in the default location, you need to specify the path to the Firefox binary[**1**](https://stackoverflow.com/questions/25713824/setting-path-to-firefox-binary-on-windows-with-selenium-webdriver)[**2**](https://stackoverflow.com/questions/65318382/expected-browser-binary-location-but-unable-to-find-binary-in-default-location). You can do this using the options.binary\_location attribute[**1**](https://stackoverflow.com/questions/25713824/setting-path-to-firefox-binary-on-windows-with-selenium-webdriver)[**2**](https://stackoverflow.com/questions/65318382/expected-browser-binary-location-but-unable-to-find-binary-in-default-location). Here's how you can modify your code:

from selenium import webdriver

from selenium.webdriver.firefox.options import Options

from selenium.webdriver.firefox.service import Service

from bs4 import BeautifulSoup

import time

from datetime import datetime as dt

import re

import xlwings as xw

import re # Regular expression library

import pickle

@xw.func

def get\_historical\_data(script\_name, end\_date, start\_date):

end\_date\_timestamp = int(dt.strptime(end\_date, "%d/%m/%Y").timestamp())

start\_date\_timestamp = int(dt.strptime(start\_date, "%d/%m/%Y").timestamp())

options = Options()

profile\_directory = r'C:\path\to\your\profile' # replace with the path to your Firefox profile

options.profile = profile\_directory

options.binary\_location = r"C:\path\to\firefox\binary" # replace with the path to your Firefox binary

service = Service(r"D:\geckodriver.exe") # replace with the path to your GeckoDriver

driver = webdriver.Firefox(service=service, options=options)

url = f"https://in.investing.com/equities/{script\_name}-historical-data?end\_date={end\_date\_timestamp}&st\_date={start\_date\_timestamp}"

driver.get(url)

time.sleep(2)

soup = BeautifulSoup(driver.page\_source, 'html.parser')

div = soup.find('div', {'class': 'common-table-scroller js-table-scroller'})

table = div.find('table', {'class': 'common-table medium js-table'})

colgroup = table.find('colgroup')

headers = [col.get('class')[0] for col in colgroup.find\_all('col')]

tbody = table.find('tbody')

data = [

[

dt.strptime(" ".join([td.text.strip().rsplit(' ', 1)[0], re.sub(r'\D', '', td.text.strip().rsplit(' ', 1)[1])]), "%b %d, %Y") if i == 0 else float(td.text.replace(',', '')) if 1 <= i <= 4 else td.text.strip()

for i, td in enumerate(tr.find\_all('td'))

]

for tr in tbody.find\_all('tr')

]

result = [headers] + data

driver.quit()

return result

In this code, replace "C:\path\to\your\profile" with the path to your Firefox profile directory, replace "D:\geckodriver.exe" with the path to your GeckoDriver executable, and replace "C:\path\to\firefox\binary" with the path to your Firefox binary[**1**](https://stackoverflow.com/questions/25713824/setting-path-to-firefox-binary-on-windows-with-selenium-webdriver)[**2**](https://stackoverflow.com/questions/65318382/expected-browser-binary-location-but-unable-to-find-binary-in-default-location).

Let me know if you have any other questions! 😊

Source(s)

1. [Setting path to firefox binary on windows with selenium webdriver](https://stackoverflow.com/questions/25713824/setting-path-to-firefox-binary-on-windows-with-selenium-webdriver)

2. [Expected browser binary location, but unable to find binary in default ...](https://stackoverflow.com/questions/65318382/expected-browser-binary-location-but-unable-to-find-binary-in-default-location)

3. [How to handle Firefox binary default location? - Stack Overflow](https://stackoverflow.com/questions/41506268/how-to-handle-firefox-binary-default-location)

4. [How to set Firefox Binary path of firefox in selenium in Linux?](https://stackoverflow.com/questions/42094021/how-to-set-firefox-binary-path-of-firefox-in-selenium-in-linux)

5. <https://www.mozilla.org/en-US/firefox/new/>

6. <http://www.seleniumhq.org>

7. <http://google.com/>

8. <http://filehippo.com/download_firefox/67599/>

# You

Firefox not supported - I found out.

# Copilot

I see, it seems like the functionality you're trying to use is not supported in Firefox. If you have any other questions or need further assistance with Selenium or any other topic, feel free to ask! 😊