# You

I have following python code for web scrapping:
import requests
import json
import numpy as np

url = 'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country=india&symbol=TTPW&from\_date=01/01/2023&to\_date=12/31/2023'
response = requests.get(url)
json\_output = response.json()
json\_dict = json.loads(json\_output)
data\_list = json\_dict['data']
data = []
for item in json\_output['data']:
data.append(list(item.values()))
np\_array = np.array(data)
numpy\_array = np.array(data)

At the step: json\_output = response.json() I receive following traceback:

>>> json\_output = response.json()

Traceback (most recent call last):

File "D:\Programs\Python\Lib\site-packages\requests\models.py", line 971, in json

return complexjson.loads(self.text, \*\*kwargs)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\\_\_init\_\_.py", line 346, in loads

return \_default\_decoder.decode(s)

^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\decoder.py", line 337, in decode

obj, end = self.raw\_decode(s, idx=\_w(s, 0).end())

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\decoder.py", line 355, in raw\_decode

raise JSONDecodeError("Expecting value", s, err.value) from None

json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "D:\Programs\Python\Lib\site-packages\requests\models.py", line 975, in json
raise RequestsJSONDecodeError(e.msg, e.doc, e.pos)
requests.exceptions.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

# Copilot

The error you're encountering is a JSONDecodeError. This error is raised when there's an issue decoding a JSON document. In your case, it's being raised when you're trying to parse the response from your HTTP request into JSON format using response.json().

The error message Expecting value: line 1 column 1 (char 0) typically means that the JSON document is empty. This could be because the URL you're trying to scrape is not returning any data or is not returning data in JSON format.

Here are a few things you can try to debug this issue:

1. **Check the Response**: Print the response text before trying to convert it to JSON to see what you're receiving. You can do this by adding a print statement like so:

print(response.text)

1. **Check the Status Code**: Ensure that your request is successful by checking the status code. If the status code is 200, then your request was successful. If not, there was an issue with your request. You can check the status code like so:

print(response.status\_code)

1. **Check the URL**: Ensure that the URL you're trying to scrape is correct and is supposed to return data in JSON format.

Remember to replace your@email.com with your actual email in the URL. If these steps don't help resolve the issue, the problem might be with the API endpoint itself, and you might need to contact the API provider for assistance. Hope this helps! Let me know if you have any other questions. 😊

# You

reponse.text is: prints null output but response.status shows 200. So data is there. When I say:

>>>json\_ouput:

{'data': [{'direction\_color': 'greenFont', 'rowDate': 'Dec 29, 2023', 'rowDateRaw': 1703808000, 'rowDateTimestamp': '2023-12-29T00:00:00Z', 'last\_close': '332.15', 'last\_open': '330.10', 'last\_max': '336.00', 'last\_min': '326.05', 'volume': '19.91M', 'volumeRaw': 19909152, 'change\_precent': '0.71', 'last\_closeRaw': '332.14999389648438', 'last\_openRaw': '330.10000610351562', 'last\_maxRaw': '336.00000000000000', 'last\_minRaw': '326.04998779296875', 'change\_precentRaw': 0.7125549395080137}, {'direction\_color': 'greenFont', 'rowDate': 'Dec 28, 2023', 'rowDateRaw': 1703721600, 'rowDateTimestamp': '2023-12-28T00:00:00Z', 'last\_close': '329.80', 'last\_open': '326.60', 'last\_max': '331.00', 'last\_min': '324.00', 'volume': '16.18M', 'volumeRaw': 16182026, 'change\_precent': '1.52', 'last\_closeRaw': '329.79998779296875', 'last\_openRaw': '326.60000610351562', 'last\_maxRaw': '331.00000000000000', 'last\_minRaw': '324.00000000000000', 'change\_precentRaw': 1.523774541003327},

So data has been received.

# Copilot

I see, you're getting the data correctly. The issue seems to be with the line where you're trying to load the JSON output again using json.loads(json\_output).

The response.json() method already converts the response content to a Python dictionary. So, you don't need to load it again.

Here's how you can modify your code:

import requests

import numpy as np

url = 'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country=india&symbol=TTPW&from\_date=01/01/2023&to\_date=12/31/2023'

response = requests.get(url)

json\_output = response.json() # This is already a Python dictionary

data\_list = json\_output['data']

data = []

for item in data\_list:

data.append(list(item.values()))

np\_array = np.array(data)

In this corrected code, I've removed the unnecessary json.loads(json\_output) line and changed the for loop to iterate over data\_list instead of json\_output['data']. This should resolve the issue you were facing. Let me know if this helps! 😊

# You

Lets go further.
I want to take this data to excel using xlwings. Define Python UDF which will be called using xlwings. I will give you input and output contents.

Name of UDF: get\_url\_data

Input arguments: country (string), symbol (string), from\_date (date in format dd/mm/yyyy), to\_date (date in format dd/mm/yyyy)

Ouput shall be convereted into list of lists because excel to xlwings interface requires that.

You need to plug in above input parameters into the url mentioned above inplace of existing parameters. You can use variables inplace of hard coded parameters:

url = 'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country=india&symbol=TTPW&from\_date=01/01/2023&to\_date=12/31/2023'

# Copilot

Sure, I can help you define a Python User Defined Function (UDF) that can be called using xlwings. Here's how you can define the get\_url\_data function:

import requests

import numpy as np

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json()

data\_list = json\_output['data']

data = []

for item in data\_list:

data.append(list(item.values()))

return data

In this function, I've used Python's f-string formatting to insert the function parameters into the URL. The @xw.func decorator is used to expose this function to Excel via xlwings.

Please replace your@email.com with your actual email in the URL.

You can call this function from Excel using the formula =get\_url\_data(country, symbol, from\_date, to\_date). Make sure to replace country, symbol, from\_date, and to\_date with the actual values or cell references.

Remember to run your Python script with the UDF using xlwings command line client or add-in. Also, ensure that the Excel workbook from which you're calling the UDF is the active workbook when you run your Python script.

Let me know if this helps or if you have any other questions! 😊

# You

But whether there will be issue I supply date like this: "01/01/2023". I think you need to to modify the code so that the date string will be converted into python understandable date format dd/mm/yyyy.

# Copilot

You're correct. The date format you're using ("dd/mm/yyyy") is common in many regions, but it's not the standard format that Python's datetime module expects, which is "yyyy-mm-dd". However, in this case, you don't need to convert the date format because the API you're using seems to accept dates in the "dd/mm/yyyy" format.

If you ever need to convert date strings to a different format in Python, you can use the datetime module's strptime and strftime functions. Here's an example:

from datetime import datetime

# Convert a date string in "dd/mm/yyyy" format to a datetime object

date\_object = datetime.strptime("01/01/2023", "%d/%m/%Y")

# Convert the datetime object back to a string in "yyyy-mm-dd" format

date\_string = datetime.strftime(date\_object, "%Y-%m-%d")

In this example, strptime is used to parse a date string into a datetime object, and strftime is used to format a datetime object as a string. The arguments to these functions are the date string and a format string that specifies the expected format of the date.

But for your current requirement, you don't need to do this conversion. You can directly use the date in "dd/mm/yyyy" format in the URL. Let me know if you have any other questions! 😊

# You

Whether you need xw decorator for input arguments or above code will work without that?

# Copilot

The @xw.func decorator is sufficient to expose your Python function as a User Defined Function (UDF) in Excel via xlwings. You don't need additional decorators for the input arguments.

Here's the function again for reference:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json()

data\_list = json\_output['data']

data = []

for item in data\_list:

data.append(list(item.values()))

return data

This function will be available in Excel as =get\_url\_data(country, symbol, from\_date, to\_date). The arguments country, symbol, from\_date, and to\_date can be cell references or direct values in the Excel formula.

Remember to replace your@email.com with your actual email in the URL.

Please note that you need to run your Python script with the UDF using the xlwings command line client or add-in. Also, ensure that the Excel workbook from which you're calling the UDF is the active workbook when you run your Python script.

Let me know if you have any other questions! 😊

# You

But you did not import json.

# Copilot

You're correct, and I apologize for the oversight. However, in this specific case, you actually don't need to import the json module. The requests module has a built-in JSON decoder, which is what you're using when you call response.json(). This method attempts to decode the response content as JSON, which is why you don't need to use json.loads().

Here's your function again for clarity:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json() # This is already a Python dictionary

data\_list = json\_output['data']

data = []

for item in data\_list:

data.append(list(item.values()))

return data

Please replace your@email.com with your actual email in the URL.

I hope this clears up any confusion. Let me know if you have any other questions! 😊

# You

I called function like this in excel:
=get\_url\_data("india","TTPW","01/01/2023","31/12/2023")

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\web\_scarpping.py", line 11, in get\_url\_data
for item in data\_list:
TypeError: 'NoneType' object is not iterable

# Copilot

The error message TypeError: 'NoneType' object is not iterable suggests that data\_list is None. This means that json\_output['data'] is returning None, which implies that the key 'data' does not exist in the json\_output dictionary.

This could happen if the API request does not return any data or if the structure of the returned data is different than expected.

To handle this, you can add a check to see if the key 'data' exists in the json\_output dictionary before trying to access it. Here's how you can modify your function:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json()

data = []

if 'data' in json\_output:

data\_list = json\_output['data']

for item in data\_list:

data.append(list(item.values()))

return data

In this modified function, I've added an if statement to check if the key 'data' exists in the json\_output dictionary. If it does, the function proceeds as before. If it doesn't, the function returns an empty list.

Please replace your@email.com with your actual email in the URL.

I hope this helps! Let me know if you have any other questions. 😊

# You

Even though my response shows <Response [200]>

# Copilot

The <Response [200]> indicates that your HTTP request was successful and the server returned a status code of 200, which means "OK". However, a status code of 200 doesn't necessarily mean that the response contains the data you're expecting. It's possible for a server to return a status code of 200 with an empty response body or with a body that doesn't contain the expected data.

If you're still getting a NoneType error when trying to access json\_output['data'], it suggests that the 'data' key is not present in the json\_output dictionary. This could happen if the server returned a JSON object that doesn't contain a 'data' key, or if the server returned an empty response body.

You might want to print out the entire json\_output to see what data you're actually receiving. You can do this by adding a print statement like so:

print(json\_output)

This will print the entire JSON response to the console, which can help you understand the structure of the response and why the 'data' key might be missing.

If you're still having trouble, please let me know! I'm here to help. 😊

# You

json\_ouput shows all data but when I issue:

json\_output = response.json()

>>> json\_output = response.json()

Traceback (most recent call last):

File "D:\Programs\Python\Lib\site-packages\requests\models.py", line 971, in json

return complexjson.loads(self.text, \*\*kwargs)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\\_\_init\_\_.py", line 346, in loads

return \_default\_decoder.decode(s)

^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\decoder.py", line 337, in decode

obj, end = self.raw\_decode(s, idx=\_w(s, 0).end())

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\Python\Lib\json\decoder.py", line 355, in raw\_decode

raise JSONDecodeError("Expecting value", s, err.value) from None

json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "D:\Programs\Python\Lib\site-packages\requests\models.py", line 975, in json
raise RequestsJSONDecodeError(e.msg, e.doc, e.pos)
requests.exceptions.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

# Copilot

I apologize for the confusion. The error you're seeing typically occurs when the response.json() method is called on a response object that doesn't contain valid JSON data.

If you're seeing all the data when you print json\_output but getting an error when you call response.json(), it's possible that the data is not in JSON format or not properly formatted as JSON.

Here are a few things you can try:

1. **Check the Response Headers**: The response headers can give you information about the format of the response data. You can check the Content-Type header like so:

print(response.headers['Content-Type'])

This will print the media type of the response data. If the data is in JSON format, you should see something like application/json.

1. **Print the Response Text**: If the Content-Type is not application/json, the data might be in a different format. You can print the raw response text to see what it looks like:

print(response.text)

This will print the response data as a string. If the data is not in JSON format, you might need to parse it differently.

1. **Try Parsing the Data Manually**: If the data is a string that looks like JSON but isn't being parsed correctly by response.json(), you can try parsing it manually with json.loads():

import json

data = json.loads(response.text)

This will attempt to parse the response text as JSON and assign the resulting object to data.

Please try these steps and let me know what you find. I'm here to help! 😊

# You

>>> print(response.headers['Content-Type'])

text/html; charset=UTF-8

So even though output looks like json but the response is as above. What are the options?

# Copilot

The Content-Type of text/html; charset=UTF-8 indicates that the server is returning the data as HTML, not JSON. This could be why you're having trouble parsing it with response.json().

However, you mentioned that the output looks like JSON. It's possible that the server is returning JSON data as a string within the HTML response. In this case, you might be able to extract the JSON data from the HTML and parse it separately.

Here's a general approach you can take:

1. **Convert the Response to Text**: Use response.text to get the response data as a string.

response\_text = response.text

1. **Find the JSON Data**: If the JSON data is embedded in the HTML, you'll need to find it. This could be tricky depending on how the data is structured. If the JSON data is enclosed in a specific HTML tag, you can use an HTML parser like BeautifulSoup to find it.

from bs4 import BeautifulSoup

soup = BeautifulSoup(response\_text, 'html.parser')

json\_data = soup.find('tagname').text # Replace 'tagname' with the actual tag name

1. **Parse the JSON Data**: Once you have the JSON data as a string, you can parse it with json.loads().

import json

data = json.loads(json\_data)

Please note that this is a general approach and might need to be adjusted based on the actual structure of the HTML response. If you're still having trouble, please let me know! I'm here to help. 😊

# You

If I manually define like this:

>>> country='india'

>>> symbol='TTPW'

>>> from\_date='01/01/2023'

>>> to\_date='31/12/2023'

>>> url

'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

As you can see url in python still does not take input parameters. Why?

# Copilot

The issue here is that you're using single quotes ' ' for your string which doesn't support string interpolation. In Python, you need to use f-strings or the format() method for string interpolation.

Here's how you can do it with f-strings:

country = 'india'

symbol = 'TTPW'

from\_date = '01/01/2023'

to\_date = '31/12/2023'

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

And here's how you can do it with the format() method:

country = 'india'

symbol = 'TTPW'

from\_date = '01/01/2023'

to\_date = '31/12/2023'

url = 'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={}&symbol={}&from\_date={}&to\_date={}'.format(country, symbol, from\_date, to\_date)

In both cases, the {} placeholders in the string will be replaced with the values of the variables. Please replace your@email.com with your actual email in the URL.

I hope this helps! Let me know if you have any other questions. 😊

# You

I think xlwings function has problem here:

data = []

for item in data\_list:

data.append(list(item.values()))

When I check these commands in python:

>>> data = []

>>> for item in data\_list:

... data.append(list(item.values()))

...

Traceback (most recent call last):

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

TypeError: 'NoneType' object is not iterable

# Copilot

The error message TypeError: 'NoneType' object is not iterable suggests that data\_list contains a None value. When the loop tries to call item.values() on None, it raises this error because None is not iterable.

To handle this, you can add a check in your loop to skip None values:

data = []

for item in data\_list:

if item is not None:

data.append(list(item.values()))

In this modified loop, item.values() will only be called if item is not None. This should prevent the TypeError from being raised.

Please try this modification and let me know if it helps! 😊

# You

I still get same error. Check again:

>>> get\_url\_data('india','TTPW','01/01/2023','31/12/2023')

Traceback (most recent call last):

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

File "<stdin>", line 7, in get\_url\_data

TypeError: 'NoneType' object is not iterable

# Copilot

I apologize for the confusion. The error message TypeError: 'NoneType' object is not iterable suggests that data\_list is None. This means that json\_output['data'] is returning None, which implies that the key 'data' does not exist in the json\_output dictionary.

This could happen if the API request does not return any data or if the structure of the returned data is different than expected.

To handle this, you can add a check to see if the key 'data' exists in the json\_output dictionary before trying to access it. Here's how you can modify your function:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json()

data = []

if 'data' in json\_output and json\_output['data'] is not None:

data\_list = json\_output['data']

for item in data\_list:

data.append(list(item.values()))

return data

In this modified function, I've added an if statement to check if the key 'data' exists in the json\_output dictionary and it is not None. If it does, the function proceeds as before. If it doesn't, the function returns an empty list.

Please replace your@email.com with your actual email in the URL.

I hope this helps! Let me know if you have any other questions. 😊

# You

How to resolve this:

url = f'http://api.scraperlink.com/investpy/?email={emailurl}&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

When I define like this {emailurl} is input variable here: I get traceback:

>>> url = f'http://api.scraperlink.com/investpy/?email={emailurl}&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

Traceback (most recent call last):

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

NameError: name 'emailurl' is not defined

# Copilot

The error message NameError: name 'emailurl' is not defined means that Python can't find a variable named emailurl. Before you can use emailurl in your f-string, you need to define it.

Here's an example of how you can define emailurl:

emailurl = 'your@email.com'

Please replace 'your@email.com' with your actual email. After defining emailurl, you should be able to use it in your f-string without getting a NameError.

Here's the complete code:

emailurl = 'your@email.com'

country = 'india'

symbol = 'TTPW'

from\_date = '01/01/2023'

to\_date = '31/12/2023'

url = f'http://api.scraperlink.com/investpy/?email={emailurl}&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

I hope this helps! Let me know if you have any other questions. 😊

# You

When I call the following function:

>>> def get\_url\_data(country, symbol, from\_date, to\_date):

... url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

... response = requests.get(url)

... json\_output = response.json() # This is already a Python dictionary

... data\_list = json\_output['data']

... data = []

... for item in json\_output:

... if item is not None:

... data.append(list(item.values()))

... return data

Error is:

>>> data\_list = json\_output['data']

>>> data = []

>>> for item in json\_output:

... if item is not None:

... data.append(list(item.values()))

...

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

AttributeError: 'str' object has no attribute 'values'

# Copilot

I apologize for the confusion. The error message AttributeError: 'str' object has no attribute 'values' is being raised because you're trying to call values() on a string object, which is not allowed.

The issue lies in the line for item in json\_output:. Here, json\_output is a dictionary, and when you iterate over a dictionary in Python, you're iterating over its keys, not its values. In this case, the keys are strings, which is why you're seeing this error.

You should be iterating over data\_list instead of json\_output. Here's the corrected function:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json() # This is already a Python dictionary

data\_list = json\_output['data']

data = []

for item in data\_list:

if item is not None:

data.append(list(item.values()))

return data

In this corrected function, I've changed the for loop to iterate over data\_list instead of json\_output. This should resolve the issue you were facing. Let me know if this helps! 😊

# You

The final xlwings code that works is as under:

import requests
import json
import xlwings as xw

@xw.func
def get\_url\_data(country, symbol, from\_date, to\_date):
url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'
response = requests.get(url)
json\_output = response.json() # This is already a Python dictionary
data\_list = json\_output['data']
data = []
for item in data\_list:
if item is not None:
data.append(list(item.values()))
return data

The only issue being there are no headers in each column in excel. Can you insert headers by identifying them from json file and that should be the first column in the data fetched as above.

# Copilot

Sure, you can add headers to your data by extracting the keys from the items in data\_list. These keys can be added as the first row in your data list. Here's how you can modify your function:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json() # This is already a Python dictionary

data\_list = json\_output['data']

data = []

if data\_list:

# Add headers as the first row in data

headers = list(data\_list[0].keys())

data.append(headers)

for item in data\_list:

if item is not None:

data.append(list(item.values()))

return data

In this modified function, I've added a check to see if data\_list is not empty. If it's not, the function extracts the keys from the first item in data\_list and adds them as the first row in data. This row will serve as the headers in your Excel sheet.

I hope this helps! Let me know if you have any other questions. 😊

# You

I see output like this: one of the columns:

rowDateRaw
1703808000
1703721600
1703635200
1703548800
1703203200
1703116800
1703030400

I think this is date only but not recognizable format in excel. Can you convert this number stamp to date using python script Define it as udf with unput as this number and output as excel date.

# Copilot

The numbers you're seeing are likely Unix timestamps, which represent the number of seconds that have passed since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.

You can convert Unix timestamps to a more readable date format using Python's datetime module. Here's how you can define a User Defined Function (UDF) to do this:

import datetime

import xlwings as xw

@xw.func

def convert\_timestamp(timestamp):

# Convert the timestamp to an integer

timestamp = int(timestamp)

# Convert the Unix timestamp to a datetime object

dt\_object = datetime.datetime.fromtimestamp(timestamp)

# Format the datetime object as a string in the format 'dd/mm/yyyy'

date\_string = dt\_object.strftime('%d/%m/%Y')

return date\_string

In this function, datetime.datetime.fromtimestamp(timestamp) converts the Unix timestamp to a datetime object, and dt\_object.strftime('%d/%m/%Y') formats this datetime object as a string in the format 'dd/mm/yyyy'.

You can call this function from Excel using the formula =convert\_timestamp(timestamp). Replace timestamp with the actual Unix timestamp or a cell reference containing the timestamp.

Please note that Unix timestamps are in UTC, so the resulting date might be different than expected if you're in a different time zone. Let me know if this helps or if you have any other questions! 😊

# You

Can you change the function because my input is excel range (list of list) and output is also list of list

# Copilot

Sure, if you're working with a range of cells in Excel (which xlwings treats as a list of lists), you can modify the function to iterate over each timestamp in the input range and return a list of lists with the converted dates. Here's how you can do it:

import datetime

import xlwings as xw

@xw.func

def convert\_timestamps(timestamps):

# Initialize an empty list to store the converted dates

converted\_dates = []

# Iterate over each list in the input range

for row in timestamps:

# Initialize an empty list to store the converted dates for this row

converted\_row = []

# Iterate over each timestamp in the row

for timestamp in row:

# Convert the timestamp to an integer

timestamp = int(timestamp)

# Convert the Unix timestamp to a datetime object

dt\_object = datetime.datetime.fromtimestamp(timestamp)

# Format the datetime object as a string in the format 'dd/mm/yyyy'

date\_string = dt\_object.strftime('%d/%m/%Y')

# Add the converted date to the row list

converted\_row.append(date\_string)

# Add the row list to the converted dates list

converted\_dates.append(converted\_row)

# Return the list of lists with the converted dates

return converted\_dates

In this function, timestamps is expected to be a list of lists, where each inner list represents a row of timestamps from your Excel range. The function returns a list of lists with the converted dates, which can be written back to an Excel range.

You can call this function from Excel using the formula =convert\_timestamps(range). Replace range with the actual range of cells containing the timestamps.

I hope this helps! Let me know if you have any other questions. 😊

# You

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\web\_scarpping.py", line 31, in convert\_timestamps
for timestamp in row:
TypeError: 'float' object is not iterable

# Copilot

I apologize for the confusion. The error TypeError: 'float' object is not iterable is raised because the function is trying to iterate over a float value. This happens when the function expects a list of lists (a list of rows, where each row is a list of timestamps), but it receives a list of floats instead.

If you're passing a single row or column of timestamps to the function, you can modify the function to handle both a list of lists and a list of floats. Here's how you can do it:

import datetime

import xlwings as xw

@xw.func

def convert\_timestamps(timestamps):

# Initialize an empty list to store the converted dates

converted\_dates = []

# Check if timestamps is a list of lists

if all(isinstance(i, list) for i in timestamps):

# Iterate over each list in the input range

for row in timestamps:

# Initialize an empty list to store the converted dates for this row

converted\_row = []

# Iterate over each timestamp in the row

for timestamp in row:

# Convert the timestamp to an integer

timestamp = int(timestamp)

# Convert the Unix timestamp to a datetime object

dt\_object = datetime.datetime.fromtimestamp(timestamp)

# Format the datetime object as a string in the format 'dd/mm/yyyy'

date\_string = dt\_object.strftime('%d/%m/%Y')

# Add the converted date to the row list

converted\_row.append(date\_string)

# Add the row list to the converted dates list

converted\_dates.append(converted\_row)

else:

# If timestamps is a list of floats, convert each timestamp

for timestamp in timestamps:

timestamp = int(timestamp)

dt\_object = datetime.datetime.fromtimestamp(timestamp)

date\_string = dt\_object.strftime('%d/%m/%Y')

converted\_dates.append(date\_string)

# Return the list of lists (or list) with the converted dates

return converted\_dates

In this modified function, the function checks if timestamps is a list of lists. If it is, the function proceeds as before. If it's not, the function assumes that timestamps is a list of floats and converts each timestamp individually.

I hope this helps! Let me know if you have any other questions. 😊

# You

Can you convert the final output into vertical. Presently it is horizontal in shape (occupying several columns instead of rows)

# Copilot

Sure, you can transpose the output so that it's vertical (occupying several rows instead of columns). Here's how you can modify the function to do this:

import requests

import xlwings as xw

@xw.func

def get\_url\_data(country, symbol, from\_date, to\_date):

url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'

response = requests.get(url)

json\_output = response.json() # This is already a Python dictionary

data\_list = json\_output['data']

data = []

if data\_list:

# Add headers as the first row in data

headers = list(data\_list[0].keys())

data.append(headers)

for item in data\_list:

if item is not None:

data.append(list(item.values()))

# Transpose the data

transposed\_data = list(map(list, zip(\*data)))

return transposed\_data

In this modified function, I've added the line `trans

# You

Can you convert output of below code into vertical (rows) instead of horizontal:

import datetime
import xlwings as xw

@xw.func
def convert\_timestamps(timestamps):
# Initialize an empty list to store the converted dates
converted\_dates = []
# Check if timestamps is a list of lists
if all(isinstance(i, list) for i in timestamps):
# Iterate over each list in the input range
for row in timestamps:
# Initialize an empty list to store the converted dates for this row
converted\_row = []
# Iterate over each timestamp in the row
for timestamp in row:
# Convert the timestamp to an integer
timestamp = int(timestamp)
# Convert the Unix timestamp to a datetime object
dt\_object = datetime.datetime.fromtimestamp(timestamp)
# Format the datetime object as a string in the format 'dd/mm/yyyy'
date\_string = dt\_object.strftime('%d/%m/%Y')
# Add the converted date to the row list
converted\_row.append(date\_string)
# Add the row list to the converted dates list
converted\_dates.append(converted\_row)
else:
# If timestamps is a list of floats, convert each timestamp
for timestamp in timestamps:
timestamp = int(timestamp)
dt\_object = datetime.datetime.fromtimestamp(timestamp)
date\_string = dt\_object.strftime('%d/%m/%Y')
converted\_dates.append(date\_string)
# Return the list of lists (or list) with the converted dates
return converted\_dates

# Copilot

Sure, you can modify the function to return a list of lists where each inner list contains only one element. This will make the output vertical when written to an Excel range. Here's how you can do it:

import datetime

import xlwings as xw

@xw.func

def convert\_timestamps(timestamps):

# Initialize an empty list to store the converted dates

converted\_dates = []

# Check if timestamps is a list of lists

if all(isinstance(i, list) for i in timestamps):

# Iterate over each list in the input range

for row in timestamps:

# Iterate over each timestamp in the row

for timestamp in row:

# Convert the timestamp to an integer

timestamp = int(timestamp)

# Convert the Unix timestamp to a datetime object

dt\_object = datetime.datetime.fromtimestamp(timestamp)

# Format the datetime object as a string in the format 'dd/mm/yyyy'

date\_string = dt\_object.strftime('%d/%m/%Y')

# Add the converted date to the converted\_dates list as a single-item list

converted\_dates.append([date\_string])

else:

# If timestamps is a list of floats, convert each timestamp

for timestamp in timestamps:

timestamp = int(timestamp)

dt\_object = datetime.datetime.fromtimestamp(timestamp)

date\_string = dt\_object.strftime('%d/%m/%Y')

# Add the converted date to the converted\_dates list as a single-item list

converted\_dates.append([date\_string])

# Return the list of lists with the converted dates

return converted\_dates

In this modified function, each converted date is added to the converted\_dates list as a single-item list ([date\_string] instead of date\_string). This will make the output vertical when written to an Excel range.

I hope this helps! Let me know if you have any other questions. 😊

# You

In the code below, I have scrapped data using custom api. Please change the code to get data in a similar way from yahoo finance using yfinance python library:

import requests
import json
import datetime
import xlwings as xw
@xw.func
def get\_url\_data(country, symbol, from\_date, to\_date):
url = f'http://api.scraperlink.com/investpy/?email=asharindani51@gmail.com&type=historical\_data&product=stocks&country={country}&symbol={symbol}&from\_date={from\_date}&to\_date={to\_date}'
response = requests.get(url)
json\_output = response.json() # This is already a Python dictionary
data\_list = json\_output['data']
data = []
if data\_list:
# Add headers as the first row in data
headers = list(data\_list[0].keys())
data.append(headers)
for item in data\_list:
if item is not None:
data.append(list(item.values()))
return data

# Copilot

Sure, you can use the yfinance library to download historical market data from Yahoo Finance. Here's how you can modify your function to do this:

import yfinance as yf

import xlwings as xw

@xw.func

def get\_stock\_data(symbol, start\_date, end\_date):

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

In this function, yf.Ticker(symbol).history(start=start\_date, end=end\_date) downloads the historical market data for the specified symbol between the start date and end date. The history method returns a pandas DataFrame.

The line data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist() prepares the data for output. It creates a list of lists where the first list is the column headers and the remaining lists are the rows of data.

You can call this function from Excel using the formula =get\_stock\_data(symbol, start\_date, end\_date). Replace symbol, start\_date, and end\_date with the actual values or cell references.

Please note that the dates should be in the format 'yyyy-mm-dd'. Also, you need to install the yfinance library if you haven't already. You can install it using pip:

pip install yfinance

I hope this helps! Let me know if you have any other questions. 😊

# You

Can you find out why using the above in xlwings like: =get\_stock\_data\_y("ZYDUSLIFE","01/01/2018","12/31/2023")

got me error in excel cellas #NAME?

# Copilot

The #NAME? error in Excel typically means that Excel doesn't recognize text in the formula. In the context of xlwings, this error usually occurs when the Python function you're trying to call from Excel is not available.

Here are a few things you can check:

1. **Function Name**: Make sure that the function name in your Excel formula matches the name of the function in your Python script. In your case, the function should be named get\_stock\_data\_y.
2. **Python Script Running**: Ensure that your Python script with the xlwings function is running while you're trying to call the function from Excel.
3. **xlwings Add-in**: Make sure that the xlwings add-in is installed and active in Excel.
4. **Workbook Name**: If you're using the xlwings add-in, make sure that the name of your Excel workbook matches the name of your Python script (without the .py extension), or that you've set the correct path to your Python script in the xlwings add-in settings.

If you've checked all of these and you're still seeing the #NAME? error, there might be an issue with your xlwings setup. You might want to try reinstalling the xlwings add-in or looking for error messages in the Python console.

I hope this helps! Let me know if you have any other questions. 😊

# You

See I change the name of the function to get\_stock\_data\_yf and then tried like this:

=get\_stock\_data\_yf("ZYDUSLIFE","01/01/2018","31/12/2023")

I get no data only top colume headers:

Open High Low Close Adj Close Volume

# Copilot

The function get\_stock\_data\_yf is correctly returning the column headers, which means it's able to connect to Yahoo Finance and retrieve the data structure. However, if it's not returning any data rows, it could mean that there's no available data for the specified symbol "ZYDUSLIFE" in the date range from "01/01/2018" to "31/12/2023".

Here are a few things you can check:

1. **Symbol**: Make sure that "ZYDUSLIFE" is the correct symbol for the stock you're interested in. Stock symbols can vary between different financial platforms, so the symbol used on one platform might not be recognized on Yahoo Finance.
2. **Date Range**: Check if there's available data for "ZYDUSLIFE" in the specified date range. It's possible that the stock didn't exist or wasn't publicly traded during that time.
3. **Data Availability**: Not all stocks have complete historical data available. It's possible that Yahoo Finance doesn't have data for "ZYDUSLIFE" for the specified date range.

You can check these things manually by going to the Yahoo Finance website, searching for "ZYDUSLIFE", and looking at its historical data. If you're still having trouble, please let me know! I'm here to help. 😊

# You

I traced the error - date format in which yfinance accepts the data: %Y-%m-%d:

Can you give me the above code changed which converts to this format when I input like this: dd/mm/yyyy

# Copilot

Sure, you can use Python's datetime module to convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd". Here's how you can modify your function to do this:

import datetime

import xlwings as xw

import yfinance as yf

@xw.func

def get\_stock\_data\_yf(symbol, start\_date, end\_date):

# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"

start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

In this function, datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d") converts the start date from "dd/mm/yyyy" to "yyyy-mm-dd". The same is done for the end date.

You can call this function from Excel using the formula =get\_stock\_data\_yf(symbol, start\_date, end\_date). Replace symbol, start\_date, and end\_date with the actual values or cell references.

Please note that the dates should be in the format 'dd/mm/yyyy'. Also, you need to install the yfinance library if you haven't already. You can install it using pip:

pip install yfinance

I hope this helps! Let me know if you have any other questions. 😊

Source(s)

1. [github.com](https://github.com/JoeHoulton/LeaveManagementAppQA/tree/8bc9f02c2b4fd07edf6f2ec1a5895ab69b346700/LeaveManagementAppQA%2Fdashboard%2Fviews.py)

# You

Error traceback is:

"Exception: All elements of a 2d list or tuple must be of the same length
raise Exception(
File ""D:\Programs\Python\Lib\site-packages\xlwings\conversion\_*init*\_.py"", line 94, in write
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
xl\_result = conversion.write(ret, None, ret\_info[""options""])
File ""D:\Programs\Python\Lib\site-packages\xlwings\udfs.py"", line 504, in call\_udf
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
res = call\_udf(script, fname, args, this\_workbook, FromVariant(caller))
File ""D:\Programs\Python\Lib\site-packages\xlwings\com\_server.py"", line 235, in CallUDF
^^^^^^^^^^^
return func(\*args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 639, in *invokeex*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return S\_OK, -1, self.*invokeex*(dispid, lcid, wFlags, args, None, None)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 310, in *invoke*
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
return self.*invoke*(dispid, lcid, wFlags, args)
File ""D:\Programs\Python\Lib\site-packages\win32com\server\policy.py"", line 305, in *Invoke*"

# Copilot

The error message Exception: All elements of a 2d list or tuple must be of the same length is raised when the function returns a list of lists (a 2D list or matrix) where not all inner lists have the same length. This can happen when you're trying to write this data to an Excel range, as Excel requires that all rows in the range have the same number of columns.

In the context of your function, this error could be caused by missing data for some dates. When the yfinance library downloads historical market data, it might not return data for every single day in the specified date range. For example, stock markets are closed on weekends and holidays, so there won't be any data for these days. If the data for a particular day is missing, yfinance will simply skip that day, which could result in rows with missing values.

To fix this issue, you can modify your function to fill in missing dates with None or some other placeholder value. However, this might complicate your function and make the data harder to interpret.

A simpler solution would be to drop any rows with missing values before returning the data. Here's how you can modify your function to do this:

import pandas as pd

import xlwings as xw

import yfinance as yf

@xw.func

def get\_stock\_data\_yf(symbol, start\_date, end\_date):

# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"

start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Drop rows with missing values

hist.dropna(inplace=True)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

In this modified function, hist.dropna(inplace=True) removes any rows in the DataFrame that contain NaN values. This should ensure that all rows in the output data have the same length.

I hope this helps! Let me know if you have any other questions. 😊

Source(s)

1. [github.com](https://github.com/JoeHoulton/LeaveManagementAppQA/tree/8bc9f02c2b4fd07edf6f2ec1a5895ab69b346700/LeaveManagementAppQA%2Fdashboard%2Fviews.py)

# You

Why pandas? Use numpy.

# Copilot

Sure, you can use numpy instead of pandas to handle the data. However, please note that yfinance returns data in a pandas DataFrame, which is a more high-level data structure and provides more functionalities for data manipulation compared to numpy arrays.

If you still prefer to use numpy, you can convert the DataFrame to a numpy array. Here's how you can modify your function:

import datetime

import xlwings as xw

import yfinance as yf

import numpy as np

@xw.func

def get\_stock\_data\_yf(symbol, start\_date, end\_date):

# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"

start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Convert DataFrame to numpy array

data\_array = hist.to\_numpy()

# Prepare data for output

data = [hist.columns.values.tolist()] + data\_array.tolist()

return data

In this modified function, hist.to\_numpy() converts the DataFrame to a numpy array. Please note that this will only include the data values, not the index or column labels of the DataFrame. The column labels are added separately to the output data.

I hope this helps! Let me know if you have any other questions. 😊

Source(s)

1. [github.com](https://github.com/JoeHoulton/LeaveManagementAppQA/tree/8bc9f02c2b4fd07edf6f2ec1a5895ab69b346700/LeaveManagementAppQA%2Fdashboard%2Fviews.py)

# You

Suppose I want to test yfinance directly in python, can you give me how to directly use yfinance without python UDF style.

# Copilot

Sure, you can use yfinance directly in a Python script to download historical market data. Here's an example:

import yfinance as yf

# Define the ticker symbol

symbol = "AAPL"

# Get data on this ticker

tickerData = yf.Ticker(symbol)

# Get the historical prices for this ticker

tickerDf = tickerData.history(period='1d', start='2020-1-1', end='2020-12-31')

# See your data

print(tickerDf)

In this script, yf.Ticker(symbol) creates a Ticker object for the specified symbol. tickerData.history(period='1d', start='2020-1-1', end='2020-12-31') downloads the historical market data for this ticker. The period parameter defines the frequency at which to gather the historical data; common options would include 1d (daily), 1mo (monthly), 1y (yearly).

The start and end parameters define the date range for the historical data.

Finally, print(tickerDf) prints the downloaded data to the console.

Please replace "AAPL" with your actual ticker symbol and adjust the date range as needed.

I hope this helps! Let me know if you have any other questions. 😊

# You

I found out the bug in the code below:

import pandas as pd
import xlwings as xw
import yfinance as yf

@xw.func
def get\_stock\_data\_yf(symbol, start\_date, end\_date):
# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"
start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")
end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Drop rows with missing values

hist.dropna(inplace=True)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date) - requires 3 input arguments as you can see in your testcode above.

# Copilot

I apologize for the confusion. The yf.Ticker(symbol).history() function requires a period argument if the start and end arguments are not provided. However, if you provide both start and end arguments, the period argument is not necessary.

In your function get\_stock\_data\_yf, you're correctly providing both start and end arguments to the history() function, so you don't need to provide a period argument.

Here's the correct usage of the history() function in your case:

import datetime

import xlwings as xw

import yfinance as yf

@xw.func

def get\_stock\_data\_yf(symbol, start\_date, end\_date):

# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"

start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(start=start\_date, end=end\_date)

# Drop rows with missing values

hist.dropna(inplace=True)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

In this function, yf.Ticker(symbol).history(start=start\_date, end=end\_date) downloads the historical market data for the specified symbol between the start\_date and end\_date.

I hope this helps! Let me know if you have any other questions. 😊

Source(s)

1. [github.com](https://github.com/JoeHoulton/LeaveManagementAppQA/tree/8bc9f02c2b4fd07edf6f2ec1a5895ab69b346700/LeaveManagementAppQA%2Fdashboard%2Fviews.py)

# You

But you said above in example code:

In this script, yf.Ticker(symbol) creates a Ticker object for the specified symbol. tickerData.history(period='1d', start='2020-1-1', end='2020-12-31') downloads the historical market data for this ticker. The period parameter defines the frequency at which to gather the historical data; common options would include 1d (daily), 1mo (monthly), 1y (yearly).

The start and end parameters define the date range for the historical data.

# Copilot

I apologize for any confusion. The period parameter in the history() function is optional and is used to define the frequency of historical market data. If you provide a start and end date, the history() function will return daily data for that date range, so you don't need to specify period='1d'.

However, if you want to specify the period parameter for clarity, you can certainly do so. Here's how you can modify your function:

import datetime

import xlwings as xw

import yfinance as yf

@xw.func

def get\_stock\_data\_yf(symbol, start\_date, end\_date):

# Convert the date format from "dd/mm/yyyy" to "yyyy-mm-dd"

start\_date = datetime.datetime.strptime(start\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

end\_date = datetime.datetime.strptime(end\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

# Download historical market data

hist = yf.Ticker(symbol).history(period='1d', start=start\_date, end=end\_date)

# Drop rows with missing values

hist.dropna(inplace=True)

# Prepare data for output

data = [hist.columns.values.tolist()] + hist.reset\_index().values.tolist()

return data

In this modified function, yf.Ticker(symbol).history(period='1d', start=start\_date, end=end\_date) downloads the daily historical market data for the specified symbol between the start\_date and end\_date.

I hope this clears up any confusion. Let me know if you have any other questions! 😊

Source(s)

1. [github.com](https://github.com/JoeHoulton/LeaveManagementAppQA/tree/8bc9f02c2b4fd07edf6f2ec1a5895ab69b346700/LeaveManagementAppQA%2Fdashboard%2Fviews.py)