How to Send a POST Request with Python

Introduction

Most tutorials focus on GET requests, but working with an API frequently requires using POST requests as well. Web servers don't randomly send data; a request must be made to the server to retrieve data before it responds with data.

In this article, we will explore how to use the requests library to make a **POST request** with headers and a body. We will cover the fundamental concepts of headers and request bodies, demonstrate their usage in the **requests.post()** method, and build two real-life projects using **ScraperAPI**.

TL;DR: Sending a Python POST Request

To quickly send a POST request in Python:

- Import the 'requests' library.
- Define the URL
- Prepare Your Data
- Send the POST request using '**requests.post()'**.
- Handle the response.

```
import requests

url = "http://httpbin.org/post"
data = {
    "Id": 001,
    "Customer": "Donald Biden",
}

response = requests.post(url, json=data)
print(response.text)
```

Here is how to send a POST request with the ScraperAPI Async API:

```
import requests

def create_async_request():
    r = requests.post(
        url='https://async.scraperapi.com/jobs',
        json={
            'apiKey': Your_API_Key,
            'url': 'https://crypto.com/price'  # Target URL
        }
    )
    print(r.text)  # Response includes the job ID and status URL
```

Prerequisites

To get the most out of this article, you must have Python installed on your computer along with the **'requests'** library. You can install requests using pip by running this command in your terminal:

```
pip install requests
```

If you manage your project dependencies with Pipenv, use this command:

```
pipenv install requests
```

Once installation is complete, you can import and use requests in your Python scripts like this:

```
import requests
```

Using Requests “post()” Method

The **requests** library provides a simple and intuitive way to interact with web APIs. The **post()** method is specifically designed for sending POST requests.

To send a POST request using the Requests Library, call the **requests.post()** method and pass the target URL as the first argument and then pass the data you want to submit to the server with the “**data**” argument. Let's explore different ways to do this:

Sending a Basic POST Request

The simplest form of a POST request can be sent by just passing the URL to the `**post()**` method of the `**requests**` library. We will be using httpbin.org as our target URL for this tutorial.

```
import requests

url = 'http://httpbin.org/post'
response = requests.post(url)
print(response.text)
```

Setting the POST Data

While our previous example demonstrated a basic POST request, let's now explore how to send data along with it. Sending data in JSON format is a common practice when interacting with APIs.

```
import requests
import json

url = "http://httpbin.org/post"

# Define the JSON data you want to send in the POST request
data = {
    "Id": 100252,
    "Customer": "Donald Biden",
    "Quantity": 1,
    "Price": 2024.00
}
```

```
# Convert the data dictionary to a JSON string
json_data = json.dumps(data)
response = requests.post(url, data=json_data)

print(response.text)
```

Here, we create a dictionary "**data"** containing the key-value pairs that we want to send . Then, we convert this dictionary to JSON format using the **json.dumps()** function. If no **Content-Type** header is passed to the **requests.post()** method, the "**application/x-www-form-urlencoded"** content type will be used instead. Remember to import the **json** module to handle JSON objects in your script.

Sending a POST Request with JSON Data

If you think our JSON example looks a bit complex, you're absolutely right. Fortunately, the **requests** library makes it even easier to handle JSON data. Instead of manually converting your dictionary into a JSON string and setting headers, you can simply use the **json** argument directly with your data. The **requests** library will handle the headers and encoding for you.

Here's a more straightforward way to send JSON data:

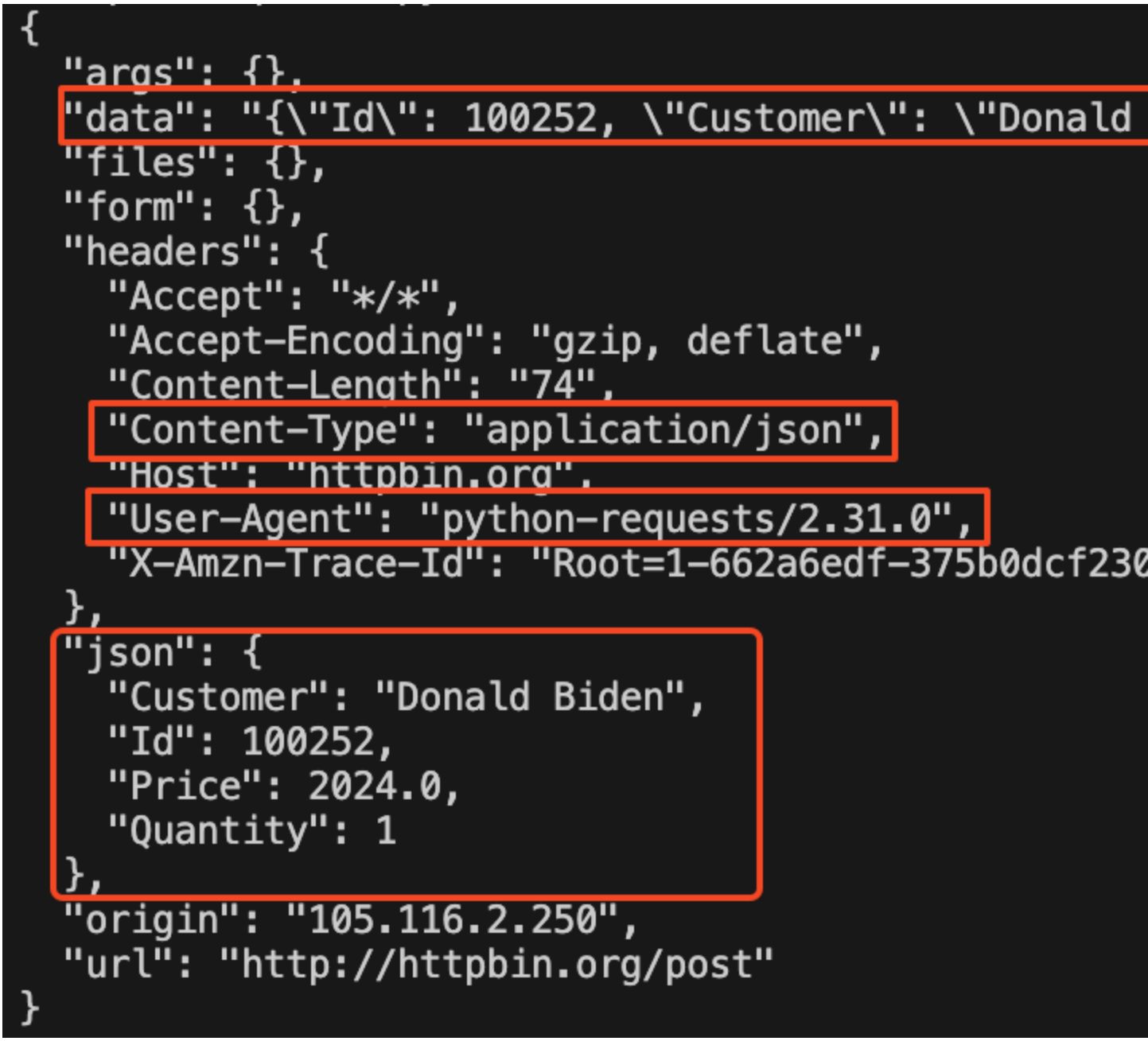
```
import requests

url = "http://httpbin.org/post"
data = {
    "Id": 100252,
    "Customer": "Donald Biden",
    "Quantity": 1,
    "Price": 2024.00
}

response = requests.post(url, json=data)
print(response.text)
```

Understanding the Response

As before, we print the server's response to examine the results.



When you send a POST request to httpbin.org/post, the server responds by echoing back information about the request. This typically includes:

- **data**: The JSON data you sent in the request, which you can verify against the data dictionary you created.
- **User-Agent**: Details about the tool used to send the request, which in this case is the Python requests library.

- **Content-Type**: This specifies the data type in the body of the POST message.

By examining the response content, you can verify if your POST request was successful and if the data was received correctly by the server.

POST Method's Syntax and Parameters

The **requests.post()** method in Python's requests library is versatile and allows for a wide range of configurations to customize your POST requests. Let's break down its syntax and explore the various parameters it accepts.

Syntax

The **requests.post()** method has the following syntax:

```
requests.post(url, data={key: value}, json={key: value}, **kwargs)
```

Basic Parameters:

- **url (Required)**: This is the URL to which the POST request is sent.
- **data**: This parameter is used to send data in the body of the POST request. It can accept various data formats, such as dictionaries, lists of tuples, bytes, or file-like objects. The specific format depends on the requirements of the API or server you're interacting with.
- **json**: A Python object that will be serialized to JSON format and sent in the request body. This is a convenient alternative to using the **data** parameter with JSON data.
- ****kwargs**: These are additional keyword arguments to provide further customization:

Example

```
url = "http://httpbin.org/post"
json_object = {"key": "value"}
response = requests.post(url, data=json_object, timeout=2.00)
```

Arguments

headers

The **headers** argument allows you to specify HTTP headers to include in the request. This is used to provide additional information to the server, such as the content type or authentication details or modify the default headers.

```
headers = {"Content-Type": "application/json"}
response = requests.post(url, data=json_data, headers=headers)
```

If no "**Content-Type"** header is provided, **requests** defaults to "**application/x-www-form-urlencoded**".

cookies

Cookies can be sent with a POST request to manage sessions or store "user-specific" data on the client-side. This can be a dictionary or a **CookieJar** object.

```
cookies = {"favcolor": "Blue"}
response = requests.post(url, data=json_data, cookies=cookies)
```

allow_redirects

The allow_redirects argument controls whether redirects should be followed. By default, redirects are followed. If you want to disable this behavior, set **allow_redirects** to **False**.

```
response = requests.post(url, data=json_data, allow_redirects=False)
```

proxies

If you need to route your request through a proxy, you can use the **proxies** argument.

```
proxies = {"http": "http://Your_proxy_IP_Address:Your_proxy_port", "https":
"https://Your_proxy_IP_Address_2:Your_proxy_port_2"}

response = requests.post(url, data=json_data, proxies=proxies)
```

stream

Typically, **requests** downloads the entire response content before making it available to your script. Setting **stream** to **True** allows you to process the response data as it arrives in **chunks**.

```
response = requests.post(url, data=json_data, stream=True)
```

timeout

The **timeout** parameter specifies a time limit for the request in seconds, after which requests will raise a timeout exception if no response is received.

```
response = requests.post(url, data=json_data, timeout=2.00)
```

files

The **files** argument allows you to upload files with your POST request. It expects a dictionary where keys are the field names for the files, and values are file-like objects opened in binary mode. This is commonly used for sending images, documents, or any other type of file data to a server.

```
import requests

url = "http://httpbin.org/post"
myfiles = {'file': open('cute_pup.png' ,'rb')}  # Open the file in binary
mode

response = requests.post(url, files = myfiles)

print(response.text)
```

These are just a few of the commonly used arguments. The requests library offers extensive customization options, making it suitable for diverse scenarios and API interactions. For more detailed information on the '**requests'** library, refer to the official documentation.

Using POST Requests With Sessions

When working with web services or APIs, especially those that require authentication or maintain state across requests, using sessions can significantly simplify your code and improve performance. **requests** provides a **Session** object that allows you to persist certain parameters across requests. This is useful for sending multiple POST requests to the same server, as it can automatically handle cookies and other state information.

Here's an example of how to use a session to send POST requests with JSON data:

```
import requests

url = "http://httpbin.org/post"

# Define the JSON data you want to send in the POST request
data = {
    "Id": 100252,
    "Customer": "Donald Biden",
    "Quantity": 1,
    "Price": 2024.00
}

data_two = {
    "Id": 100253,
    "Customer": "Thomas John",
    "Quantity": 3,
    "Price": 1994.99
}

# Create a session object
session = requests.Session()

# Set the Content-Type header to application/json for all requests in the
session
session.headers.update({'Content-Type': 'application/json'})

# Send the POST request with JSON data using the session object
response_one = session.post(url, json=data)

# You can send another POST request using the same session
```

```
response_two = session.post(url, json=data_two)

# Close the session
session.close()
```

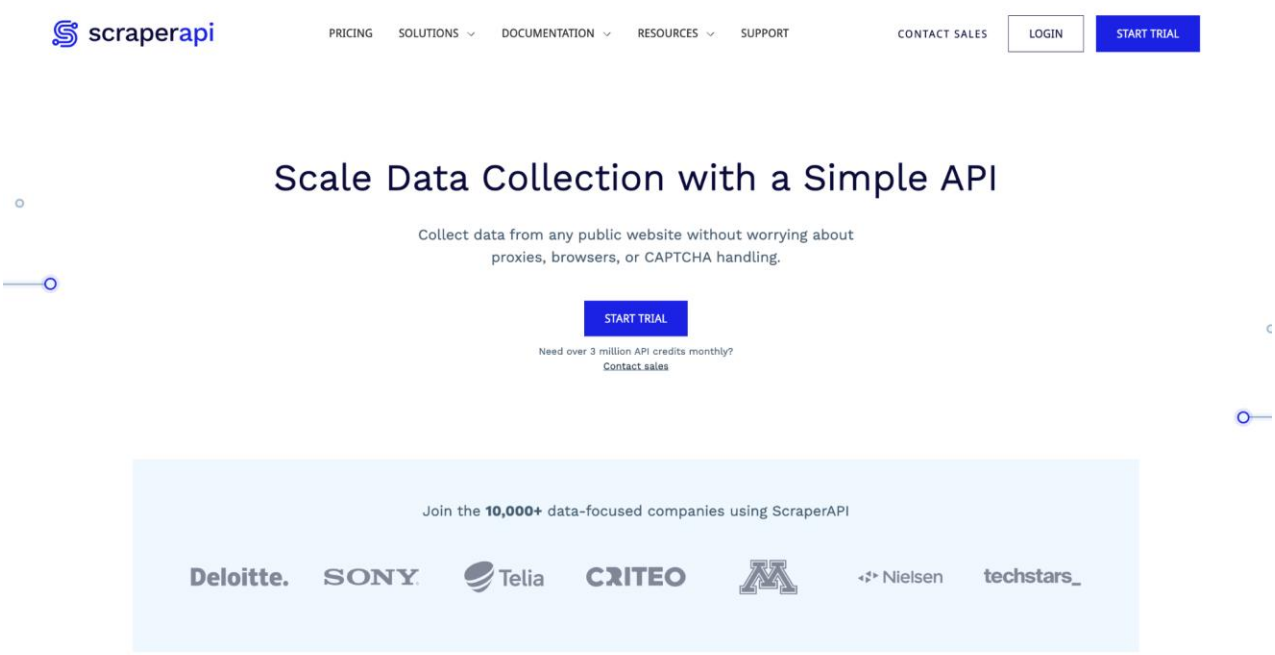
Here, we create a Session object using requests.Session(). This session object allows us to persist certain parameters, such as headers, across multiple requests. Then, we update the session's headers to include **'Content-Type': 'application/json**', ensuring that all requests sent through this session will have this header.

Next, we use the **session.post()** method to send our POST request with the JSON data. This method works similarly to **requests.post(),** but it uses the session's parameters. We can send multiple requests using the same session, and the session will handle cookies and other stateful information automatically.

Finally, we close the session using **session.close()** to free up resources.

Using ScraperAPI's async scraper with POST requests

For larger scraping projects, dealing with complexities like dynamic content, IP blocking, and browser management can become overwhelming. ScraperAPI is a tool designed to handle these challenges for you. It provides an API that allows you to scrape websites without worrying about infrastructure or anti-scraping measures, saving you time and effort.



Using ScraperAPI is straightforward. Just send the URL you would like to scrape to the API along with your API key and our API will return the HTML response from the URL you want to scrape.

To get started, create an account on ScraperAPI and obtain your API key. You can start with a free trial that includes **5,000 free API credits.**

Sending a POST Request to Crypto.com Using ScraperAPI's Async Requests Method

One of ScraperAPI's latest features is its async scraping functionality, which is ideal for handling websites that might have slower response times or when you need to scrape many URLs concurrently.

Let's explore how to use ScraperAPI's async requests method with POST requests to retrieve cryptocurrency prices from crypto.com.

Submitting an Asynchronous Scraping Job

To submit an async job, you send a POST request to the ScraperAPI endpoint with your API key and the target URL. This request triggers a job, and the response includes a unique **statusUrl** that you can use to check the job's progress.

```
import requests

def create_async_request():
    r = requests.post(
        url='https://async.scraperapi.com/jobs',
        json={
            'apiKey': Your_API_Key,
            'url': 'https://crypto.com/price'  # Target URL
        }
    )
    print(r.text)  # Response includes the job ID and status URL
```

*Note: Remember to replace **Your_API_Key** with your **ScraperAPI API key**.*

Upon execution, you should receive a response with an **id** (the job identifier), a **statusUrl** (where you can check the job status), and other details. Your response should be similar to this:


```
{"id":"373442c0-2ddd-4549-ad67-
e8df20d9eab2","attempts":0,"status":"running","statusUrl":"https://async.sc
raperapi.com/jobs/373442c0-2ddd-4549-ad67-
e8df20d9eab2","url":"https://crypto.com/price","supposedToRunAt":"2024-04-
25T18:43:02.228Z"}
```


This response indicates that the job has been created and is currently running.

Check the Status of an Async Job

To check the status of the async job, use the **statusUrl** provided in the response:

```
import requests

def check_async_response(statusUrl):
    response = requests.get(url=statusUrl)
    print(response.text)
```

By running this code, you can track the job's progress. Initially, the status might be "**running**", but once completed, it changes to "**finished**", with the response containing the scraped data.

Retrieving Results

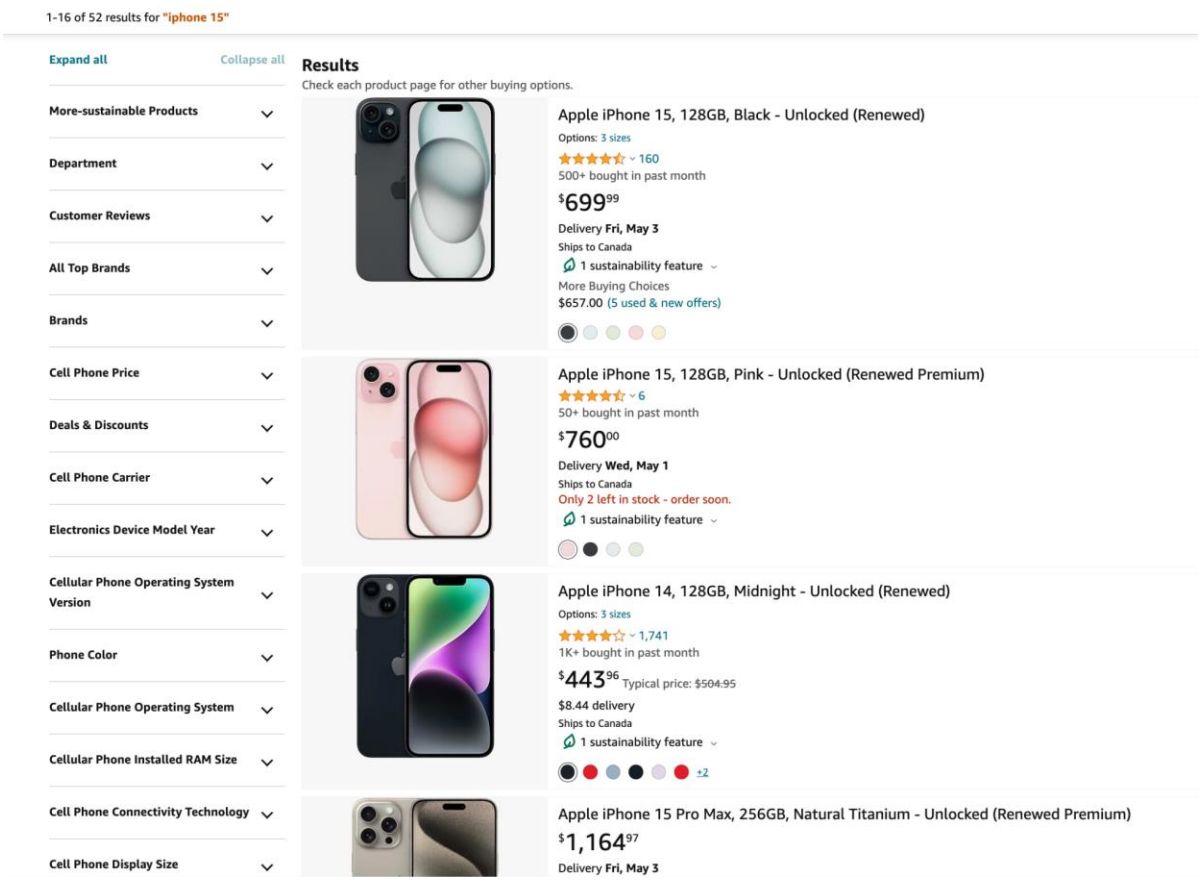
When the job status is "**finished**", the response key in the output will contain the scraped content. You need to retrieve the results within 4 hours, as the response data is stored for a limited time before being deleted.


```
{
    "id": "6cf7e34f-9575-4578-b736-5fc4c10642c1",
    "attempts": 0,
    "status": "finished",
    "statusUrl": "https://async.scraperapi.com/jobs/6cf7e34f-9575-4578-b736-
5fc4c10642c1",
    "url": "https://crypto.com/price",
    "response": {
        "headers": {
            "date": "Thu, 25 Apr 2024 18:30:41 GMT",
```

```
            "content-type": "text/html; charset=utf-8",
            "content-length": "580450",
            "connection": "keep-alive",
            "x-powered-by": "Express",
            "access-control-allow-origin": "undefined",
            "access-control-allow-headers": "Origin, X-Requested-With, Content-
Type, Accept",
            "access-control-allow-methods": "HEAD,GET,POST,DELETE,OPTIONS,PUT",
            "access-control-allow-credentials": "true",
            "x-robots-tag": "none",
            "set-cookie": [
                "__cf_bm=de1cLy7ovqKrsve2oAPm3ON8hd_40aBlpp4xfJGZGng-1714069840-
1.0.1.1-
e0Gy5Dj2AcQbjycbep5_U3AFaYJkA_vnEDFwFseHorJzYXcewZOA4zBoJrjeCzMnMETJL3c.rNR1YgLxkUJ
cgA; path=/; expires=Thu, 25-Apr-24 19:00:40 GMT; domain=.crypto.com; HttpOnly;
Secure; SameSite=None",
                "_cfuvid=MEhpvpk8UxL0qYf6f2wA2LX233JqXen2tXcEt3IRVic-1714069840990-
0.0.1.1-604800000; path=/; domain=.crypto.com; HttpOnly; Secure; SameSite=None"
            ],
            "sa-final-url": "https://crypto.com/price",
            "sa-statuscode": "200",
            "sa-credit-cost": "1",
            "sa-proxy-hash": "undefined",
            "etag": "W/\"8db62-YXxX1TgxjdwlH9trc7gaXNiPuT4\"",
            "vary": "Accept-Encoding",
            "strict-transport-security": "max-age=15724800; includeSubDomains"
        },
        "body": "Response HTML here",
        "statusCode": 200,
        "credits": 1
    }
}
```


Scraping Amazon Data with ScraperAPI's Async Scraper

ScraperAPI offers specialized tools for scraping data from popular platforms like Amazon. Our "**Amazon Search API (Async)**" provides a structured way to retrieve product information based on search terms, and it's useful for larger-scale data extraction tasks. Let's explore how to utilize this API to scrape Amazon product search data.



To begin, we import the **requests** library. Then we define a **main()** function that will contain our code. Since we'll be sending data in JSON format, we set the **Content-Type** header to **application/jso**n. This ensures that the server understands the format of the data we're sending.

```
import requests

def main():
    url = "https://async.scraperapi.com/structured/amazon/search"
    headers = {
        "Content-Type": "application/json"
    }
    data = {
        "apiKey": "Your_API_Key",
        "query": "iPhone 15",
        "ref": "olp_f_usedAcceptable",
        "s": "price-desc-rank",
        "tld": "com",
        "callback": {
```

```
            "type": "webhook",
            "url": "https://www.amazon.com/"  # Replace with your callback URL
        }
    }
    response = requests.post(url, json=data, headers=headers)
    print(response.text)

main()
```

*Note: Remember to replace **Your_API_Key** with your **ScraperAPI API key**.*

Here, we provide the parameters that define our Amazon search:

- **apiKey**: This is where you replace "**Your_API_Key**" with your actual ScraperAPI API key, which authenticates your access to the Async Scraper.
- **query**: This is the search term you want to use on Amazon, for example, "**iPhone 15**" to retrieve results related to the iPhone 15.
- **ref**: A reference string used by Amazon, such as "**olp_f_usedAcceptable**".
- **s**: This parameter controls the sorting of the search results. For instance, "**price-desc-rank**" sorts the results by price in descending order.
- **tld:** This specifies the top-level domain of the Amazon website you want to search, such as "**com**" for amazon.com or "**co.uk**" for amazon.co.uk.

With our parameters set, we send a POST request to the API endpoint using the **requests.post()** method, including the specified headers and JSON data. ScraperAPI will then process our request and respond with information about the submitted job, including a unique job ID and a **statusUrl.**


```
{"id":"bfa2f6ca-e76a-4fcc-a7ae-
da8beaa4040d","attempts":0,"status":"running","statusUrl":"https://async.sc
raperapi.com/jobs/bfa2f6ca-e76a-4fcc-a7ae-da8beaa4040d","query":"iPhone
15","ref":"olp_f_usedAcceptable","s":"price-desc-
rank","tld":"com","supposedToRunAt":"2024-04-25T19:05:07.026Z"}
```


To check the status of our scraping job and retrieve the results, we define a function called **check_status()**. This function takes the **statusUrl** obtained from the initial response as its argument.

Using **requests.get()**, we send a GET request to the **statusUrl**, allowing us to determine whether the job is still running or has completed.

```
def check_status(statusUrl):
    r = requests.get(url = statusUrl)
    print(r.text)
check_status("https://async.scraperapi.com/jobs/bfa2f6ca-e76a-4fcc-a7ae-da8beaa4040d")
```

Once the job is finished, the response from the **statusUrl** will contain the extracted product data under the response key in the JSON object. The final response will include the structured data extracted from Amazon, formatted in JSON, making it easy to parse and utilize for further analysis or integration into your applications.


```
{
    "id": "bfa2f6ca-e76a-4fcc-a7ae-da8beaa4040d",
    "attempts": 0,
    "status": "finished",
    "statusUrl": "https://async.scraperapi.com/jobs/bfa2f6ca-e76a-4fcc-a7ae-da8beaa4040d",
    "query": "iPhone 15",
    "ref": "olp_f_usedAcceptable",
    "s": "price-desc-rank",
    "tld": "com",
    "response": {
        "headers": {
            "date": "Thu, 25 Apr 2024 19:05:11 GMT",
            "content-type": "application/json; charset=utf-8",
            "content-length": "26593",
            "connection": "keep-alive",
            "x-powered-by": "Express",
            "access-control-allow-origin": "undefined",
            "access-control-allow-headers": "Origin, X-Requested-With, Content-Type,
Accept",
            "access-control-allow-methods": "HEAD,GET,POST,DELETE,OPTIONS,PUT",
            "access-control-allow-credentials": "true",
            "x-robots-tag": "none",
            "set-cookie": [
                "sp-cdn=delete; Domain=.amazon.com; Expires=Thu, 01-Jan-1970 00:00:10 GMT;
Path=/; Secure; HttpOnly"
            ],
            "sa-final-url":
"https://www.amazon.com/s?k=iPhone%2015&ref=olp_f_usedAcceptable&s=price-desc-rank",
            "sa-statuscode": "200",
            "sa-credit-cost": "5",
            "sa-proxy-hash": "undefined",
```

```
            "etag": "W/\"67e1-qw7jH5r29IrlzNOWtgrJyQt30i8\"",
            "vary": "Accept-Encoding",
            "strict-transport-security": "max-age=15724800; includeSubDomains"
        },
        "body": {
            "ads": [
                {
                    "name": "Boost Infinite iPhone 15 Pro Max (256 GB) \u2014 Natural
Titanium [Locked]. Requires unlimited plan starting at $60/mo.",
                    "asin": "B0CHBQTL9Z",
                    "brand": "iPhone on Boost Infinte",
                    "image": "https://m.media-amazon.com/images/I/81YQTED1r5L.jpg",
                    "has_prime": false,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.4,
                    "total_reviews": 0,
                    "url": "https://aax-us-
iad.amazon.com/x/c/RC98KFmWsAEdxduP9dXJQTYAAAGPFqW3UAEAAAH2AQBvbm9fdHhuX2JpZDcgICBvbm9fdHhuX
2ltcDEgICDgnjxr/https://www.amazon.com/gp/aw/d/B0CHBQTL9Z/?_encoding=UTF8&pd_rd_plhdr=t&aaxi
tk=d3d8785e02cd16d1c54fa1ccc21b7282&hsa_cr_id=0&qid=1714071910&sr=1-1-9e67e56a-6f64-441f-
a281-df67fc737124&ref_=sbx_be_s_sparkle_lsi4d_asin_0_bkgd&pd_rd_w=2pU4D&content-
id=amzn1.sym.417820b0-80f2-4084-adb3-fb612550f30b%3Aamzn1.sym.417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_p=417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&pd_rd_wg=72Qhi&pd_rd_r=a19f9e66-2e07-4342-8293-
6c1f2087fa47",
                    "type": "top_stripe_ads"
                },
                {
                    "name": "Apple iPhone 15 Pro (128 GB) - Natural Titanium | [Locked] |
Boost Infinite plan required starting at $60/mo. | Unlimited Wireless | No trade-in needed
to start | Get the latest iPhone every year",
                    "asin": "B0CHBMRD1G",
                    "brand": "iPhone on Boost Infinte",
                    "image": "https://m.media-amazon.com/images/I/81lOtLouS4L.jpg",
                    "has_prime": false,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.5,
                    "total_reviews": 0,
                    "url": "https://aax-us-
iad.amazon.com/x/c/RC98KFmWsAEdxduP9dXJQTYAAAGPFqW3UAEAAAH2AQBvbm9fdHhuX2JpZDcgICBvbm9fdHhuX
2ltcDEgICDgnjxr/https://www.amazon.com/gp/aw/d/B0CHBMRD1G/?_encoding=UTF8&pd_rd_plhdr=t&aaxi
tk=d3d8785e02cd16d1c54fa1ccc21b7282&hsa_cr_id=0&qid=1714071910&sr=1-2-9e67e56a-6f64-441f-
a281-df67fc737124&ref_=sbx_be_s_sparkle_lsi4d_asin_1_bkgd&pd_rd_w=2pU4D&content-
id=amzn1.sym.417820b0-80f2-4084-adb3-fb612550f30b%3Aamzn1.sym.417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_p=417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&pd_rd_wg=72Qhi&pd_rd_r=a19f9e66-2e07-4342-8293-
6c1f2087fa47",
                    "type": "top_stripe_ads"
```

```
                },
                {
                    "name": "Apple iPhone 15 (512 GB) - Pink | [Locked] | Boost Infinite
plan required starting at $60/mo. | Unlimited Wireless | No trade-in needed to start | Get
the latest iPhone every year",
                    "asin": "B0CHBPJYF2",
                    "brand": "iPhone on Boost Infinte",
                    "image": "https://m.media-amazon.com/images/I/71SK-KD0OwL.jpg",
                    "has_prime": false,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.1,
                    "total_reviews": 0,
                    "url": "https://aax-us-
iad.amazon.com/x/c/RC98KFmWsAEdxduP9dXJQTYAAAGPFqW3UAEAAAH2AQBvbm9fdHhuX2JpZDcgICBvbm9fdHhuX
2ltcDEgICDgnjxr/https://www.amazon.com/gp/aw/d/B0CHBPJYF2/?_encoding=UTF8&pd_rd_plhdr=t&aaxi
tk=d3d8785e02cd16d1c54fa1ccc21b7282&hsa_cr_id=0&qid=1714071910&sr=1-3-9e67e56a-6f64-441f-
a281-df67fc737124&ref_=sbx_be_s_sparkle_lsi4d_asin_2_bkgd&pd_rd_w=2pU4D&content-
id=amzn1.sym.417820b0-80f2-4084-adb3-fb612550f30b%3Aamzn1.sym.417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_p=417820b0-80f2-4084-adb3-
fb612550f30b&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&pd_rd_wg=72Qhi&pd_rd_r=a19f9e66-2e07-4342-8293-
6c1f2087fa47",
                    "type": "top_stripe_ads"
                }
            ],
            "results": [
                {
                    "type": "search_product",
                    "position": 1,
                    "asin": "B0CHBQTL9Z",
                    "name": "iPhone 15 Pro Max (256 GB) \u2014 Natural Titanium [Locked].
Requires unlimited plan starting at $60/mo.",
                    "image": "https://m.media-amazon.com/images/I/81YQTED1r5L.jpg",
                    "section_name": null,
                    "has_prime": true,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.4,
                    "total_reviews": 108,
                    "url": "https://www.amazon.com/Apple-iPhone-Pro-Max-
trade/dp/B0CHBQTL9Z/ref=sxin_9_hcs-iphone-pl-us-1?content-id=amzn1.sym.924d5a95-2224-4bd3-
93c0-a60c9770a288%3Aamzn1.sym.924d5a95-2224-4bd3-93c0-
a60c9770a288&cv_ct_cx=iPhone+15&dib=eyJ2IjoiMSJ9.mkVwt1leAIWwm7eXPZKnwBipfbqLQGvzK8viyhIvugu
OLaFIB9S-77bJkyElM615.TRC7OQ48MLGcWS4f-aZW2XEz5uJWCqmM-
rvRAykInhM&dib_tag=se&keywords=iPhone+15&pd_rd_i=B0CHBQTL9Z&pd_rd_r=6083195e-1f49-43d0-af9a-
3c827a6514c4&pd_rd_w=BQaTJ&pd_rd_wg=Fuoa9&pf_rd_p=924d5a95-2224-4bd3-93c0-
a60c9770a288&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&qid=1714071910&sbo=RZvfv%2F%2FHxDF%2BO5021pAnSA%3D
%3D&sr=1-1-acced174-0150-4fc5-bcd1-b989ebc3c57d",
                    "spec": {},
                    "price_string": "$0.01",
```

```
                    "price_symbol": "$",
                    "price": 0.01,
                    "original_price": {
                        "price_string": "$1,199.99",
                        "price_symbol": "$",
                        "price": 1199.99
                    }
                },
                {
                    "type": "search_product",
                    "position": 2,
                    "asin": "B0CHBMRD1G",
                    "name": "Apple iPhone 15 Pro (128 GB) - Natural Titanium | [Locked] |
Boost Infinite plan required starting at $60/mo. | Unlimited Wireless | No trade-in needed
to start | Get the latest iPhone every year",
                    "image": "https://m.media-amazon.com/images/I/81lOtLouS4L.jpg",
                    "section_name": null,
                    "has_prime": true,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.5,
                    "total_reviews": 35,
                    "url": "https://www.amazon.com/Apple-iPhone-Pro-128-
trade/dp/B0CHBMRD1G/ref=sxin_9_hcs-iphone-pl-us-1?content-id=amzn1.sym.924d5a95-2224-4bd3-
93c0-a60c9770a288%3Aamzn1.sym.924d5a95-2224-4bd3-93c0-
a60c9770a288&cv_ct_cx=iPhone+15&dib=eyJ2IjoiMSJ9.mkVwt1leAIWwm7eXPZKnwBipfbqLQGvzK8viyhIvugu
OLaFIB9S-77bJkyElM615.TRC7OQ48MLGcWS4f-aZW2XEz5uJWCqmM-
rvRAykInhM&dib_tag=se&keywords=iPhone+15&pd_rd_i=B0CHBMRD1G&pd_rd_r=6083195e-1f49-43d0-af9a-
3c827a6514c4&pd_rd_w=BQaTJ&pd_rd_wg=Fuoa9&pf_rd_p=924d5a95-2224-4bd3-93c0-
a60c9770a288&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&qid=1714071910&sbo=RZvfv%2F%2FHxDF%2BO5021pAnSA%3D
%3D&sr=1-2-acced174-0150-4fc5-bcd1-b989ebc3c57d",
                    "spec": {},
                    "price_string": "$0.01",
                    "price_symbol": "$",
                    "price": 0.01,
                    "original_price": {
                        "price_string": "$999.99",
                        "price_symbol": "$",
                        "price": 999.99
                    }
                },
                {
                    "type": "search_product",
                    "position": 3,
                    "asin": "B0CHBPTX1P",
                    "name": "Apple iPhone 15 Plus (128 GB) - Pink | [Locked] | Boost
Infinite plan required starting at $60/mo. | Unlimited Wireless | No trade-in needed to
start | Get the latest iPhone every year",
                    "image": "https://m.media-amazon.com/images/I/71iNOcZuxEL.jpg",
                    "section_name": null,
                    "has_prime": true,
```

```
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4,
                    "total_reviews": 14,
                    "url": "https://www.amazon.com/Apple-iPhone-Plus-128-
trade/dp/B0CHBPTX1P/ref=sxin_9_hcs-iphone-pl-us-1?content-id=amzn1.sym.924d5a95-2224-4bd3-
93c0-a60c9770a288%3Aamzn1.sym.924d5a95-2224-4bd3-93c0-
a60c9770a288&cv_ct_cx=iPhone+15&dib=eyJ2IjoiMSJ9.mkVwt1leAIWwm7eXPZKnwBipfbqLQGvzK8viyhIvugu
OLaFIB9S-77bJkyElM615.TRC7OQ48MLGcWS4f-aZW2XEz5uJWCqmM-
rvRAykInhM&dib_tag=se&keywords=iPhone+15&pd_rd_i=B0CHBPTX1P&pd_rd_r=6083195e-1f49-43d0-af9a-
3c827a6514c4&pd_rd_w=BQaTJ&pd_rd_wg=Fuoa9&pf_rd_p=924d5a95-2224-4bd3-93c0-
a60c9770a288&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&qid=1714071910&sbo=RZvfv%2F%2FHxDF%2BO5021pAnSA%3D
%3D&sr=1-3-acced174-0150-4fc5-bcd1-b989ebc3c57d",
                    "spec": {},
                    "price_string": "$0.01",
                    "price_symbol": "$",
                    "price": 0.01,
                    "original_price": {
                        "price_string": "$929.99",
                        "price_symbol": "$",
                        "price": 929.99
                    }
                },
                {
                    "type": "search_product",
                    "position": 4,
                    "asin": "B0CHBNXW73",
                    "name": "Apple iPhone 15 (128 GB) - Pink | [Locked] | Boost Infinite
plan required starting at $60/mo. | Unlimited Wireless | No trade-in needed to start | Get
the latest iPhone every year",
                    "image": "https://m.media-amazon.com/images/I/71SK-KD0OwL.jpg",
                    "section_name": null,
                    "has_prime": true,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "stars": 4.1,
                    "total_reviews": 24,
                    "url": "https://www.amazon.com/Apple-iPhone-128-Unlimited-
trade/dp/B0CHBNXW73/ref=sxin_9_hcs-iphone-pl-us-1?content-id=amzn1.sym.924d5a95-2224-4bd3-
93c0-a60c9770a288%3Aamzn1.sym.924d5a95-2224-4bd3-93c0-
a60c9770a288&cv_ct_cx=iPhone+15&dib=eyJ2IjoiMSJ9.mkVwt1leAIWwm7eXPZKnwBipfbqLQGvzK8viyhIvugu
OLaFIB9S-77bJkyElM615.TRC7OQ48MLGcWS4f-aZW2XEz5uJWCqmM-
rvRAykInhM&dib_tag=se&keywords=iPhone+15&pd_rd_i=B0CHBNXW73&pd_rd_r=6083195e-1f49-43d0-af9a-
3c827a6514c4&pd_rd_w=BQaTJ&pd_rd_wg=Fuoa9&pf_rd_p=924d5a95-2224-4bd3-93c0-
a60c9770a288&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&qid=1714071910&sbo=RZvfv%2F%2FHxDF%2BO5021pAnSA%3D
%3D&sr=1-4-acced174-0150-4fc5-bcd1-b989ebc3c57d",
                    "spec": {},
                    "price_string": "$0.01",
                    "price_symbol": "$",
                    "price": 0.01,
```

```
                    "original_price": {
                        "price_string": "$829.99",
                        "price_symbol": "$",
                        "price": 829.99
                    }
                },
                {
                    "type": "search_product",
                    "position": 5,
                    "asin": "B0CHBQTL9Z",
                    "name": "iPhone 15 Pro Max (256 GB) \u2014 Natural Titanium [Locked]. Requires unlimited plan starting at $60/mo.",
                    "image": "https://m.media-amazon.com/images/I/81YQTED1r5L.jpg",
                    "has_prime": true,
                    "is_best_seller": false,
                    "is_amazon_choice": false,
                    "is_limited_deal": false,
                    "purchase_history_message": "300+ bought in past month",
                    "stars": 4.4,
                    "total_reviews": 108,
                    "url": "https://www.amazon.com/Apple-iPhone-Pro-Max-trade/dp/B0CHBQTL9Z/ref=sxin_9_hcs-iphone-pl-us-1?content-id=amzn1.sym.924d5a95-2224-4bd3-93c0-a60c9770a288%3Aamzn1.sym.924d5a95-2224-4bd3-93c0-a60c9770a288&cv_ct_cx=iPhone+15&dib=eyJ2IjoiMSJ9.mkVwt1leAIWwm7eXPZKnwBipfbqLQGvzK8viyhIvuguOLaFIB9S-77bJkyElM615.TRC7OQ48MLGcWS4f-aZW2XEz5uJWCqmM-rvRAykInhM&dib_tag=se&keywords=iPhone+15&pd_rd_i=B0CHBQTL9Z&pd_rd_r=6083195e-1f49-43d0-af9a-3c827a6514c4&pd_rd_w=BQaTJ&pd_rd_wg=Fuoa9&pf_rd_p=924d5a95-2224-4bd3-93c0-a60c9770a288&pf_rd_r=0CSW0M6TC43N4C5C0AJZ&qid=1714071910&sbo=RZvfv%2F%2FHxDF%2BO5021pAnSA%3D%3D&sr=1-1-acced174-0150-4fc5-bcd1-b989ebc3c57d",
                    "spec": {},
                    "price_string": "$0.01",
                    "price_symbol": "$",
                    "price": 0.01,
                    "original_price": {
                        "price_string": "$1,199.99",
                        "price_symbol": "$",
                        "price": 1199.99
                    }
                }, Truncated to the first 5 results...
            }
            ],
            "explore_more_items": [],
            "next_pages": []
        },
        "statusCode": 200,
        "credits": 5
    }
}
```

Keep Learning

You've seen how easy it is to make POST requests to websites or APIs using Requests, and how we can reduce much of the common code in our applications by using it. By following this guide, you should now have a solid understanding of how to send POST requests using Python.

We learned how to send JSON data using parameters like **'data'**, **'json'**, or **'files'**, and even built real-world projects using POST requests with ScraperAPI. To further enhance your knowledge, we encourage you to explore the following resources:

- Requests Documentation
- ScraperAPI Documentation
- ScraperAPI page