



[Flask Templates](#) [Jinja2](#) [Flask-REST API](#) [Python SQLAlchemy](#) [Flask Bcrypt](#) [Flask Cookies](#) [Json](#) [Postman](#)

How to return a JSON response from a Flask API ?

Last Updated : 12 Jun, 2023

Flask is one of the most widely used python micro-frameworks to design a REST API. In this article, we are going to learn how to create a simple REST API that returns a simple JSON object, with the help of a flask.

Prerequisites: [Introduction to REST API](#)

What is a REST API?

REST stands for Representational State Transfer and is an architectural style used in modern web development. It defines a set of rules/constraints for a web application to send and receive data. In this article, we will build a REST API in Python using the Flask framework. Flask is a popular micro framework for building web applications.

Approaches: We are going to write a simple flask API that returns a JSON response using two approaches:

1. Using Flask jsonify object.
2. Using the flask_restful library with Flask.

Libraries Required:

- Install the python *Flask* library using the following command:

```
pip install Flask
```

- Install the *flask-restful* library using the following command:

```
pip install Flask-RESTful
```

Approach 1: Using Flask jsonify object – In this approach, we are going to return a JSON response using the flask jsonify method. We are not going to use

the flask-restful library in this method.

- Create a new python file named 'main.py'.
- import Flask, jsonify, and request from the flask framework.
- Register the web app into an app variable using the following syntax.

```
app = Flask(__name__)
```

- Create a new function named 'ReturnJSON'. This function is going to return the sample JSON response.
- Route the 'ReturnJSON' function to your desired URL using the following syntax.

```
@app.route('/path_of_the_response', methods = ['GET'])
def ReturnJSON():
    pass
```

- Inside the 'ReturnJSON' function if the request method is 'GET' then create a python dictionary with the two elements message.
- Jsonify the python dictionary and return it.
- Build the flask application using the following command.

```
if __name__=='__main__':
    app.run(debug=True)
```

- Run the 'main.py' file in the terminal or the IDE.

Code:

Python3

```
from flask import Flask,jsonify,request

app = Flask(__name__)

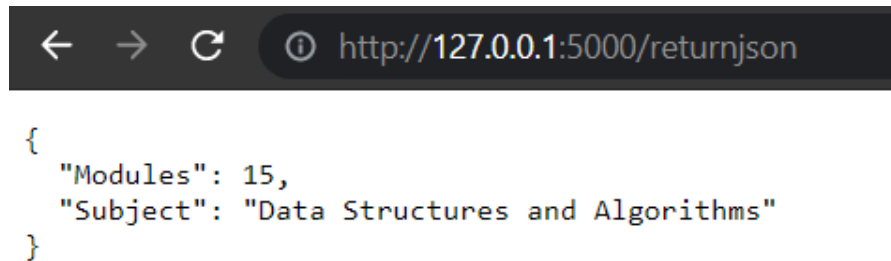
@app.route('/returnjson', methods = ['GET'])
def ReturnJSON():
    if(request.method == 'GET'):
        data = {
            "Modules" : 15,
```

```
        "Subject" : "Data Structures and Algorithms",
    }

    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True)
```

Output:

A screenshot of a web browser window. The address bar shows the URL 'http://127.0.0.1:5000/returnjson'. The page content displays a JSON object: {'Modules': 15, 'Subject': 'Data Structures and Algorithms'}.

```
{
  "Modules": 15,
  "Subject": "Data Structures and Algorithms"
}
```

Approach 2: Using the flask_restful library with Flask – In this approach, we are going to create a simple JSON response with the help of the flask-restful library. The steps are discussed below:

- Create a new python file named 'main.py'.
- Import Flask from the flask framework.
- Import API and Resource from the 'flask_restful' library.
- Register the web app into an app variable using the following syntax.

```
app = Flask(__name__)
```

- Register the app variable as an API object using the API method of the 'flask_restful' library.

```
api = Api(app)
```

- Create a resource class named 'ReturnJSON'.
- Inside the resource, the class creates a 'get' method.
- Return a dictionary with the simple JSON response from the 'get' method.

- Add the resource class to the API using the add_resource method.
- Build the flask application using the following command.

```
if __name__=='__main__':  
    app.run(debug=True)
```

- Run the 'main.py' file in the terminal or the IDE.

Code:

Python3

```
from flask import Flask  
from flask_restful import Api, Resource  
  
app = Flask(__name__)  
  
api = Api(app)  
  
class returnjson(Resource):  
    def get(self):  
        data={  
            "Modules": 15,  
            "Subject": "Data Structures and Algorithms"  
        }  
        return data  
  
api.add_resource(returnjson, '/returnjson')  
  
if __name__=='__main__':  
    app.run(debug=True)
```

Output:

```
← → ↻ ⓘ http://127.0.0.1:5000/returnjson

{
  "Modules": 15,
  "Subject": "Data Structures and Algorithms"
}
```



3

Previous Article

[Flask Cookies](#)

Next Article

[Flask and SQLAlchemy Tutorial for Database](#)

Similar Reads

How to intercept response.send() / response.json() in Express JS

In the context of Express , "intercept" usually refers to the process of capturing or modifying a request or response in a middleware function before it reaches its...

3 min read

How to Write Postman Test to Compare the Response JSON against anothe...

In the era of API-driven development, testing the API responses to ensure their correctness is of paramount importance. One such test includes comparing the...

4 min read

Flask API Authentication with JSON Web Tokens

Authentication is the process of verifying the identity of the user. It checks whether the user is real or not. It is used to provide access to resources only to...

7 min read

Why does Vite create multiple TypeScript config files: tsconfig.json,...

In Vite TypeScript projects you may have noticed three typescript configuration files, tsconfig.json, tsconfig.app.json, and tsconfig.node.json. Each file is used for ...

6 min read

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application. The Session is the time between the client logs in to...

4 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

How to return an error if response not received in 1 minute in Node.js ?

In Node.js, sometimes due to some errors, a response takes more time than usual and in this case, instead of waiting for an uncertain period, we can return an erro...

2 min read

Article Tags : [Python](#) [Web Technologies](#) [Python Flask](#)

Practice Tags : [python](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

[About Us](#)
[Legal](#)
[In Media](#)
[Contact Us](#)
[Advertise with us](#)
[GFG Corporate Solution](#)
[Placement Training Program](#)
[GeeksforGeeks Community](#)

DSA

[Data Structures](#)
[Algorithms](#)
[DSA for Beginners](#)
[Basic DSA Problems](#)
[DSA Roadmap](#)
[Top 100 DSA Interview Problems](#)
[DSA Roadmap by Sandeep Jain](#)
[All Cheat Sheets](#)

Languages

[Python](#)
[Java](#)
[C++](#)
[PHP](#)
[GoLang](#)
[SQL](#)
[R Language](#)
[Android Tutorial](#)
[Tutorials Archive](#)

Data Science & ML

[Data Science With Python](#)
[Data Science For Beginner](#)
[Machine Learning](#)
[ML Maths](#)
[Data Visualisation](#)
[Pandas](#)
[NumPy](#)
[NLP](#)
[Deep Learning](#)

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Interview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved