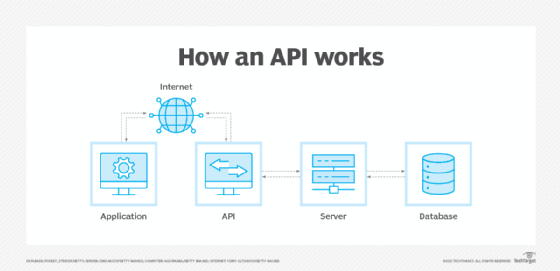
APPLICATION PROGRAMMING INTERFACE (API):

**What is API?**

* Application programming interface is a set of instructions,rules,etc. And it is a tool that allows some software application to communicate with each other.
* This can be used as client-server communication, which can give request from client side and will respond from server side to client’s request.

API also can be used as real-time data. For example, A product was sold and it have been updated in the database through api.



As the above diagram explains that the application user will give request in internet then the internet will give a call to api for user request, after that api will connect with server the server will get the data from database and after server will retrieve data from database, the server will respond for the user request through api, then the api will show the result through internet and the internet will show the output of request given by user.

API can be used with python,ruby,c,c++,etc…

In this application I am going to use application with python language,

This python code can be used with flask framework. The flask framework will be written in python, it doesn’t require particular tools or libraries.

API Requests in Python:

i) Command for python installs requests:

* pip install requests

ii) Command for import request:

* import requests

iii) Write a program using python:

For example,

from flask import Flask, jsonify, request

app = Flask(\_\_name\_\_)

@app.route('/', methods = ['GET', 'POST'])

def home():

if(request.method == 'GET'):

data = "hello world"

return jsonify({'data': data})

@app.route('/home/<hello>', methods = ['GET'])

def disp(hello):

return jsonify({'data': hello})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)

The things we have to learn in api is RESTful api:

* REST (Representational State Transfer) is a popular architectural style for designing networked applications. Learn about RESTful APIs, which are based on REST principles and are widely used for web services.
* Understand concepts such as endpoints, HTTP methods (GET, POST, PUT, DELETE), status codes, and request/response structures.

i) Endpoints:

* Endpoints are specific URLs (Uniform Resource Locators) that are used to access different resources or functionalities provided by an API.
* Each endpoint typically represents a different action or operation that can be performed.
* For example:
  + https://api.example.com/users:

//An endpoint for retrieving a list of users.

* + https://api.example.com/products/:id:

// An endpoint for retrieving information about a specific product identified by its ID.

ii) HTTP Methods:

* HTTP (Hypertext Transfer Protocol) methods define the actions that can be performed on resources identified by URLs.
* The most common HTTP methods used in API interactions are:
  + GET: Used to retrieve data from the server without modifying it.
  + POST: Used to submit data to the server to create a new resource.
  + PUT: Used to update or replace an existing resource on the server.
  + DELETE: Used to remove a resource from the server.

#### **Codes related to “GET” request:**

* 200 OK: The server successfully processed the request, and the requested data is returned.
* 201 Created: A new resource is created on the server as a result of the request.
* 204 No Content: The request is successful, but there is no additional data to return.
* 300 Multiple Choices: The requested resource has multiple representations, each with its own URL.
* 302 Found (Temporary Redirect): The requested resource is temporarily located at a different URL.
* 304 Not Modified: The client’s cached copy of the resource is still valid, and no re-download is necessary.
* 400 Bad Request: The request has malformed syntax or contains invalid data, making it incomprehensible to the server.
* 401 Unauthorized: Authentication is required, and the client’s credentials (e.g., API key) are missing or invalid.
* 500 Internal Server Error: An unexpected server error occurred during request processing.
* 502 Bad Gateway: Acting as a gateway or proxy, the server received an invalid response from an upstream server.

How the code works:

1. First open vscode, then create a new python file.
2. Second, write the python program then save it.
3. After that, open the terminal and write:

* pip install requests

1. After that import:

* import requests

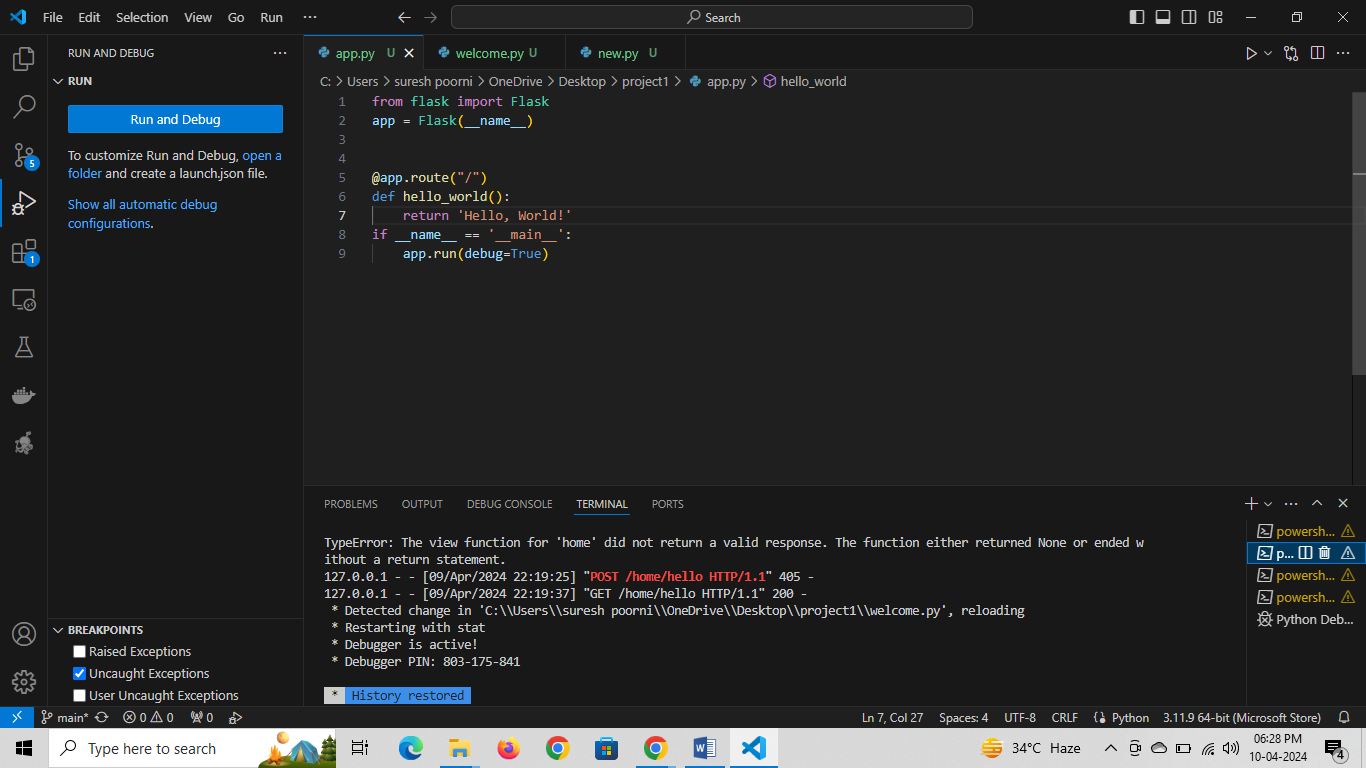
1. Run the python program

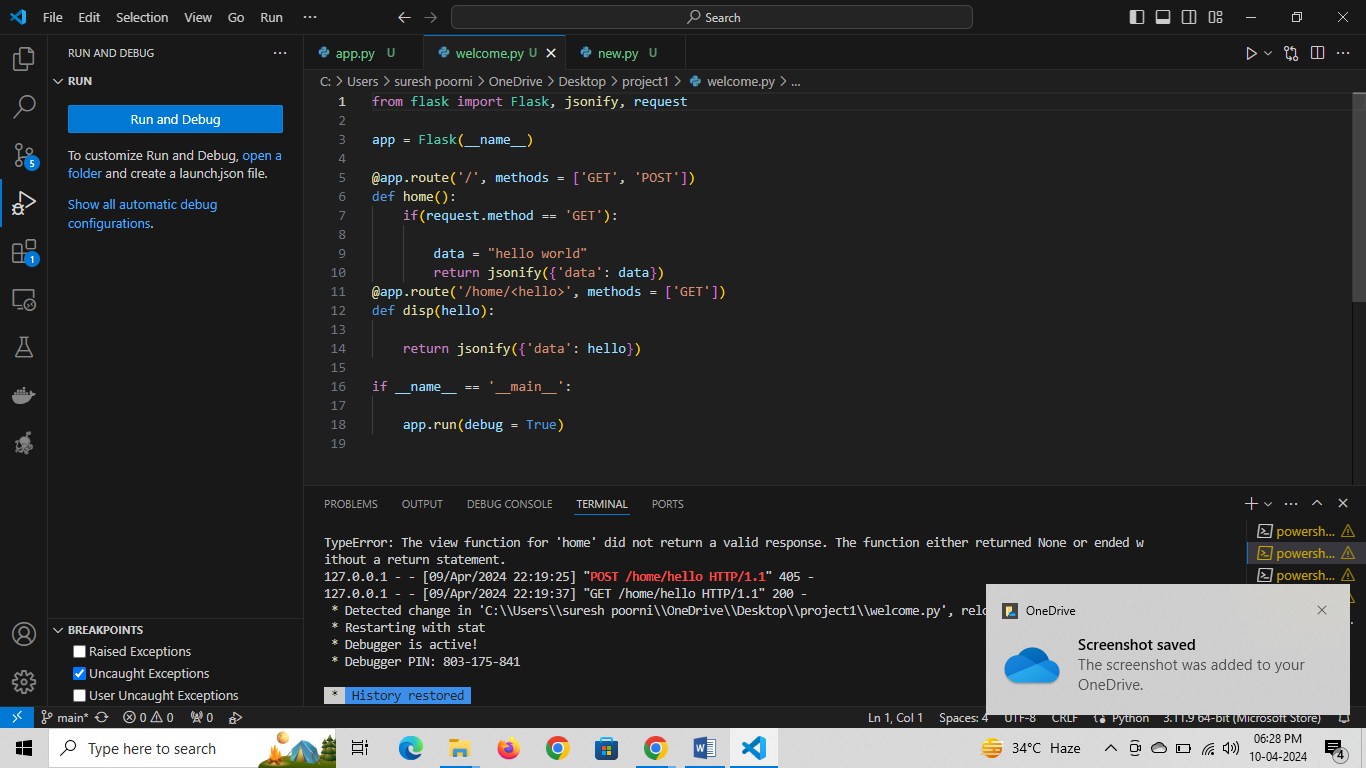
* python filename.py

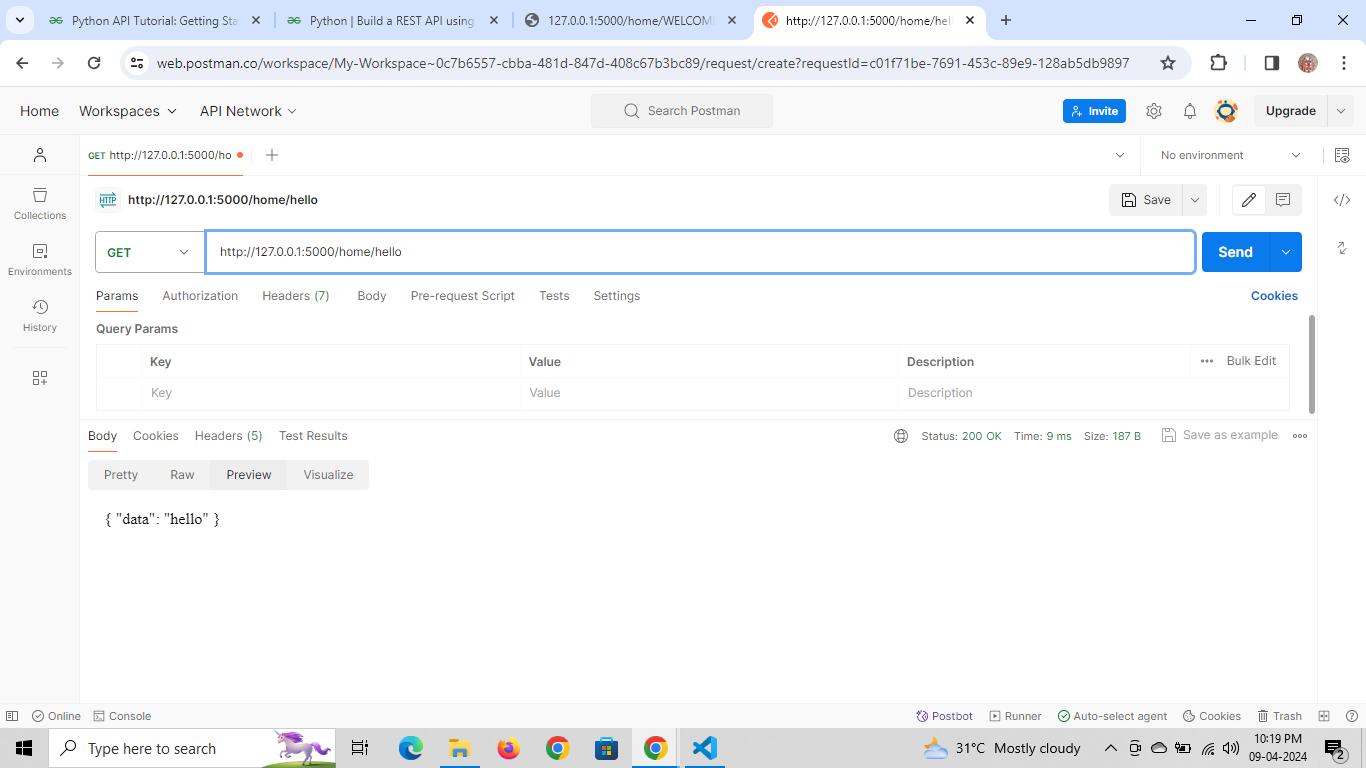
1. Then, the server will get started and click the link 127.0.0.1:5000
2. The web will show the content in browser.
3. The link to change with five methods are get,put,post,patch,delete
4. These link can be changed at the endpoint
5. For eg: 127.0.0:5000/home

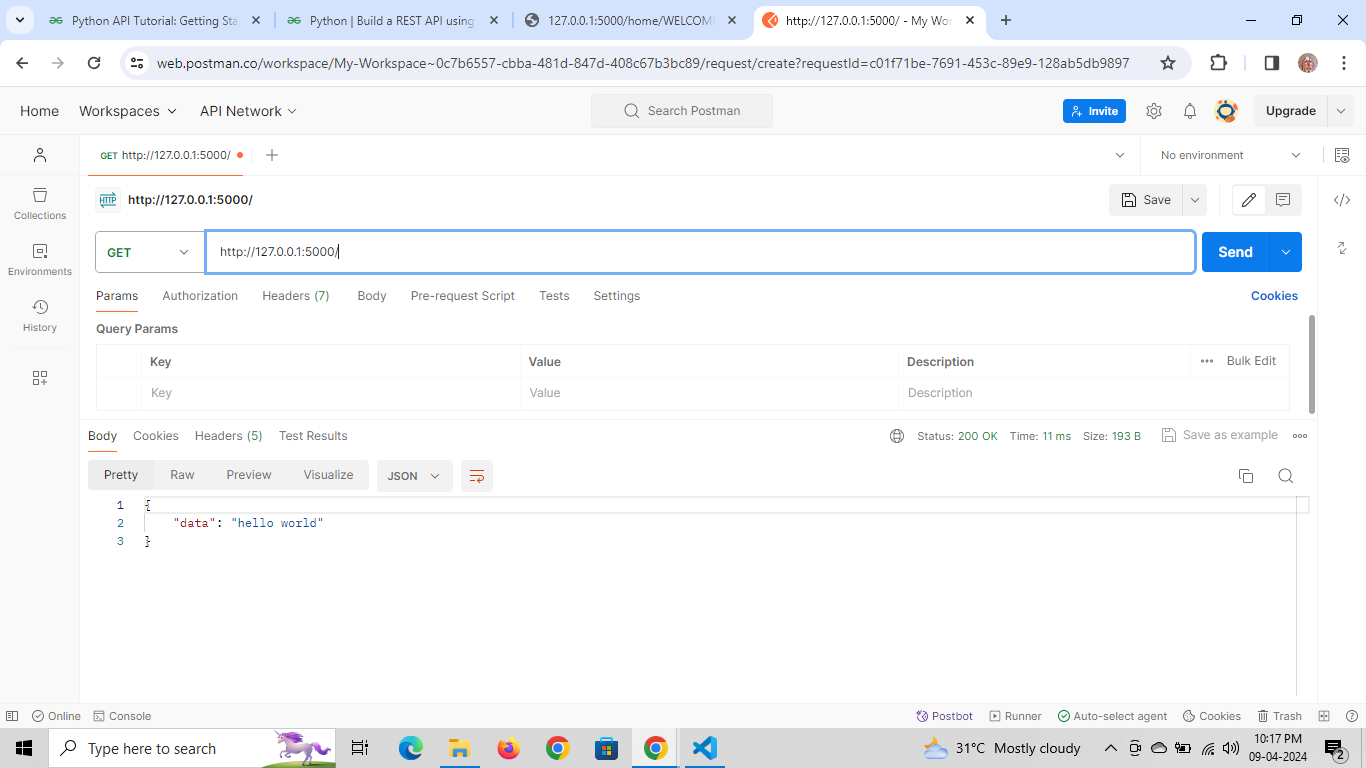
//This will show the content of home page.

1. End the task by exiting the terminal.









The above pictures explains the running of api programming.

Date: 10/04/2024

Today I was learning about how api works in real data.

For example, the weather can be predicated and updated in website by api programming

Source code:

from flask import Flask, jsonify, request

app = Flask(\_\_name\_\_)

weather\_data = {

    "city": "New York",

    "temperature": 22,

    "humidity": 60

}

@app.route('/weather', methods=['GET'])

def get\_weather():

    return jsonify(weather\_data)

@app.route('/weather', methods=['POST'])

def update\_weather():

    data = request.get\_json()

    if not data:

        return jsonify({"error": "No data provided"}), 400

    weather\_data.update(data)

    return jsonify({"message": "Weather data updated successfully"}), 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

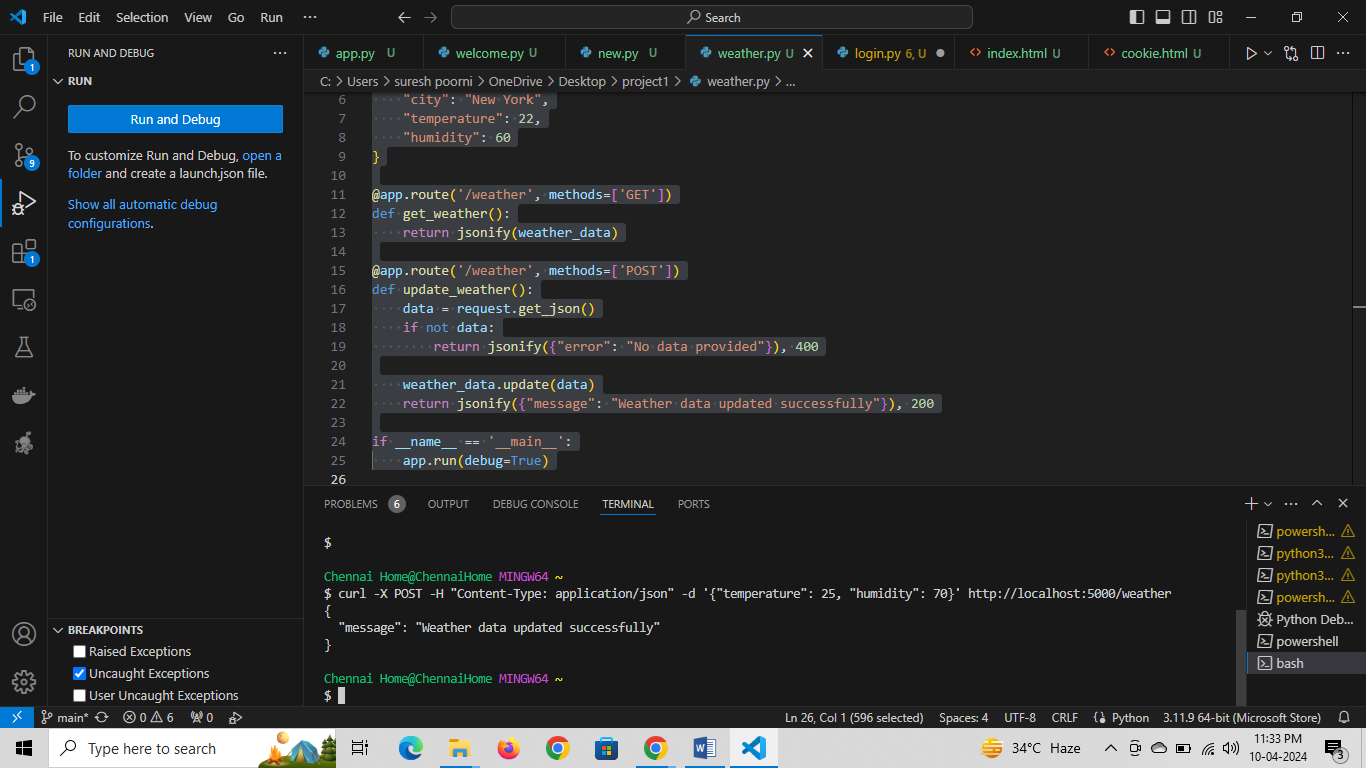
The program can be run in vscode or any other application.

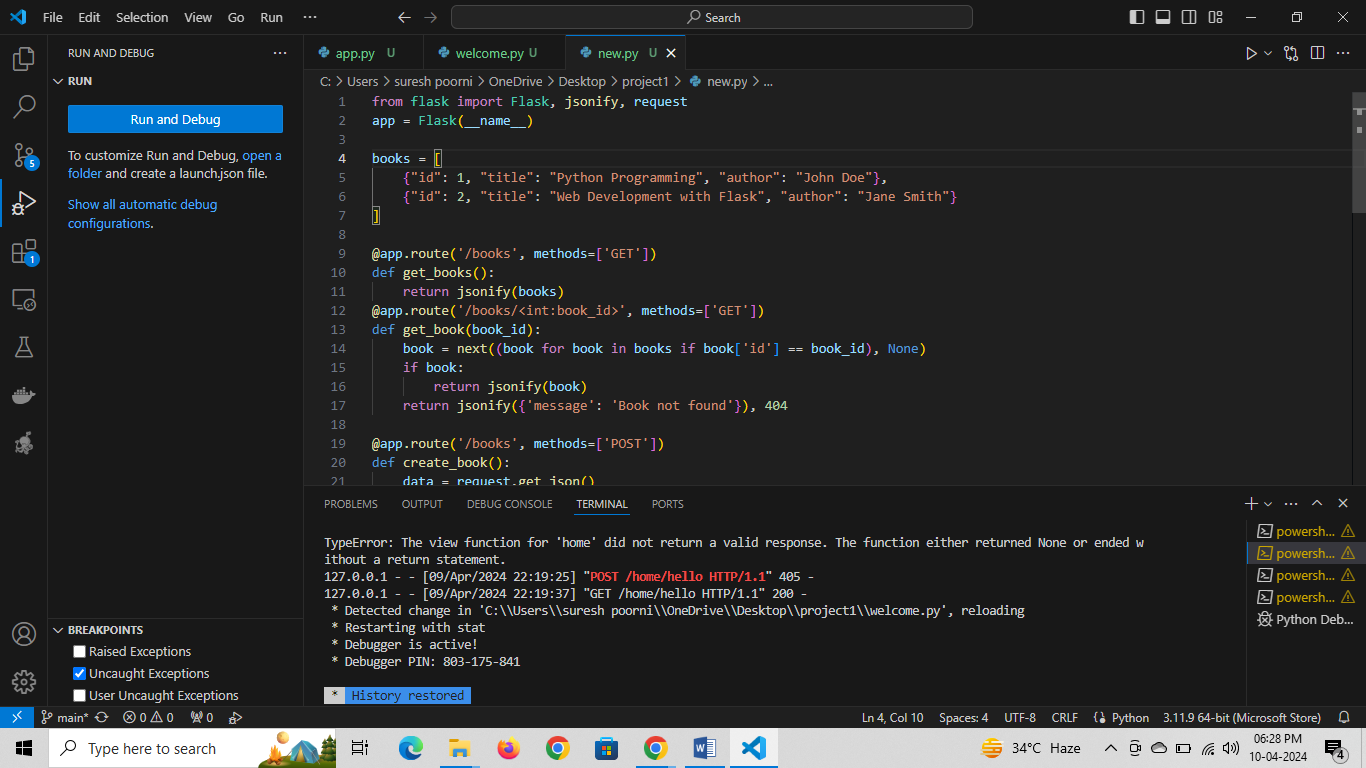
This example shows the steps:

* We create a Flask application instance.
* We define a **/weather** endpoint for both GET and POST requests.
* The GET request returns the weather data as JSON.
* The POST request updates the weather data based on the JSON payload provided in the request body.

To run this Flask application:

1. Save the code into a Python file, e.g., **app.py**.
2. Open a terminal and navigate to the directory containing **app.py**.
3. Run the Flask application:





This program will works as the command input,

The code run as the logic and the output will show in browser and can update the data in the program.