API Documents

Flask API Tool

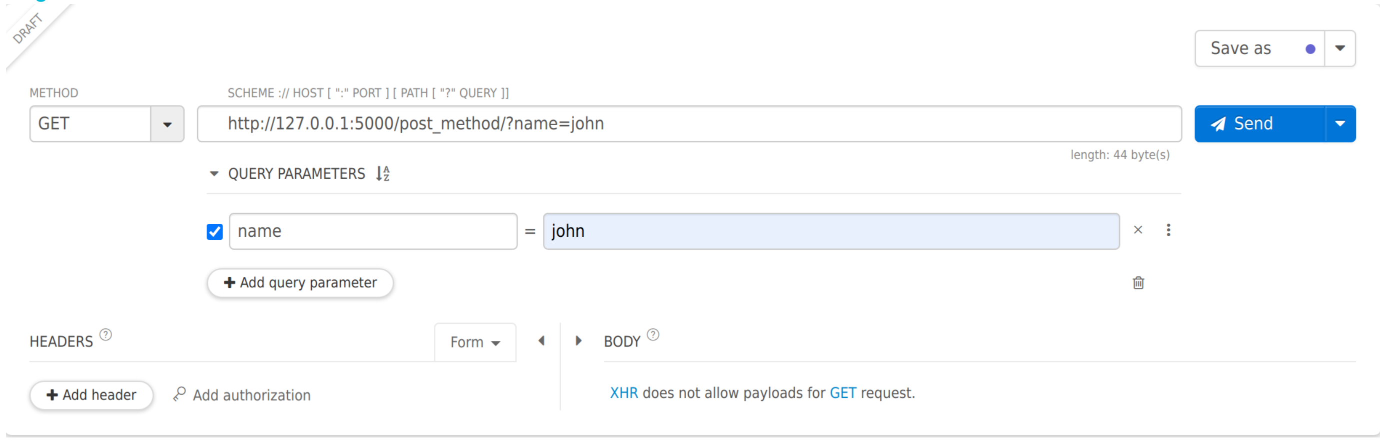
Flask query parameter Request or api router parameter

ex:

my route http://127.0.0.1:5000/post_method/

how to give parameter in route http://127.0.0.1:5000/post_method/?name=john

single value



http://127.0.0.1:5000/post_method/?name=john

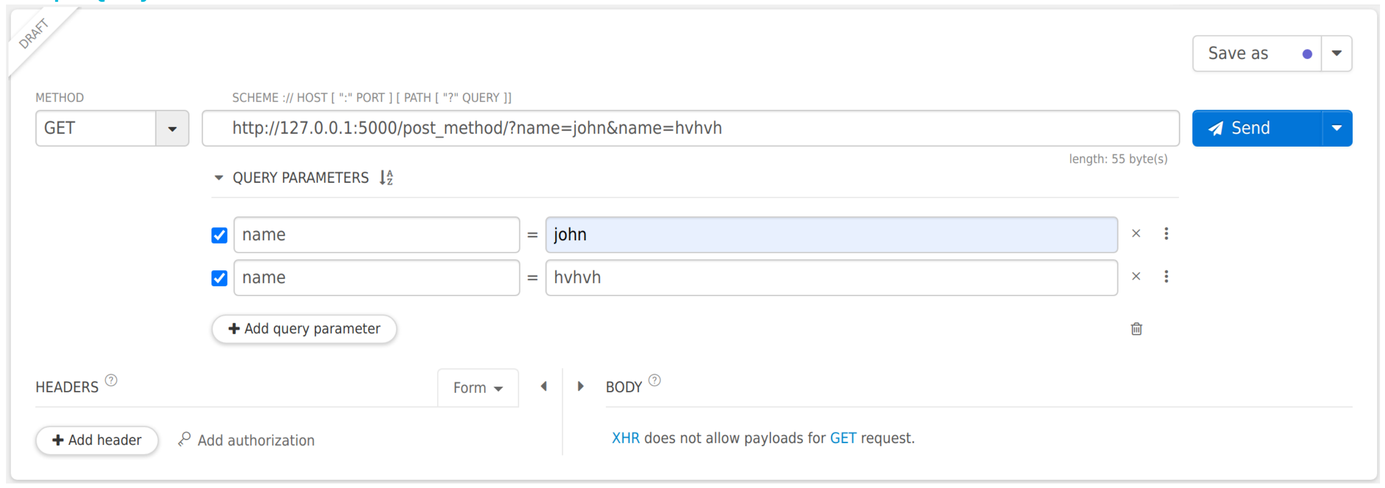
length: 44 byte(s)

name = john

XHR does not allow payloads for GET request.

@app.route('/post_method/', methods=['GET'])
def mew_va():
name = request.args.get('name')
print(name)

Multiple Query value



http://127.0.0.1:5000/post_method/?name=john&name=hvhvh

length: 55 byte(s)

name = john

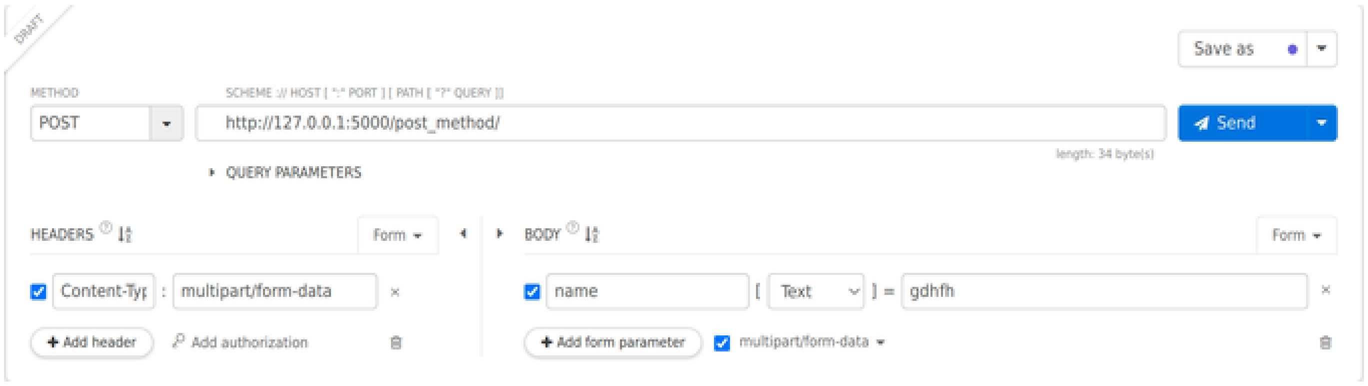
name = hvhvh

XHR does not allow payloads for GET request.

@app.route('/post_method/', methods=['GET'])
def mew_va():
name = request.args.getlist('name')
print(name[0], name[1])

Flask Form Request

Single Form


DRAFT

Save as

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

Send

length: 34 byte(s)

QUERY PARAMETERS

HEADERS: Content-Typ : multipart/form-data

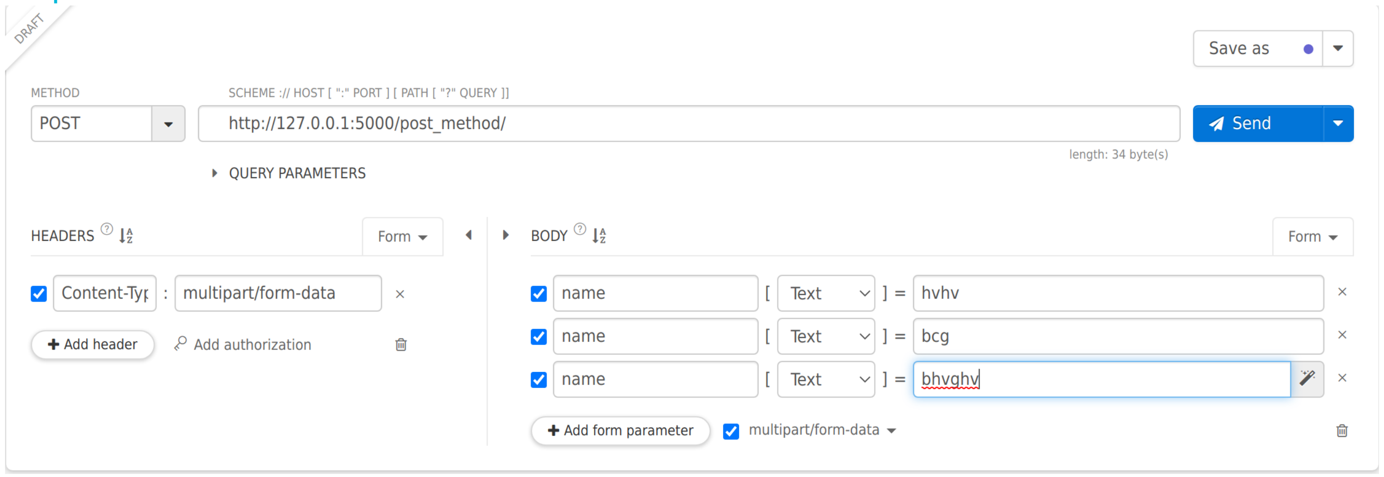
Add header · Add authorization

BODY: name [Text] = gdhfh

Add form parameter · multipart/form-data

request.form['name'] or request.form.get('name')

Multiple Form


DRAFT

Save as

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

Send

length: 34 byte(s)

QUERY PARAMETERS

HEADERS: Content-Typ : multipart/form-data

Add header · Add authorization

BODY:
name [Text] = hvhv
name [Text] = bcg
name [Text] = bhvghv

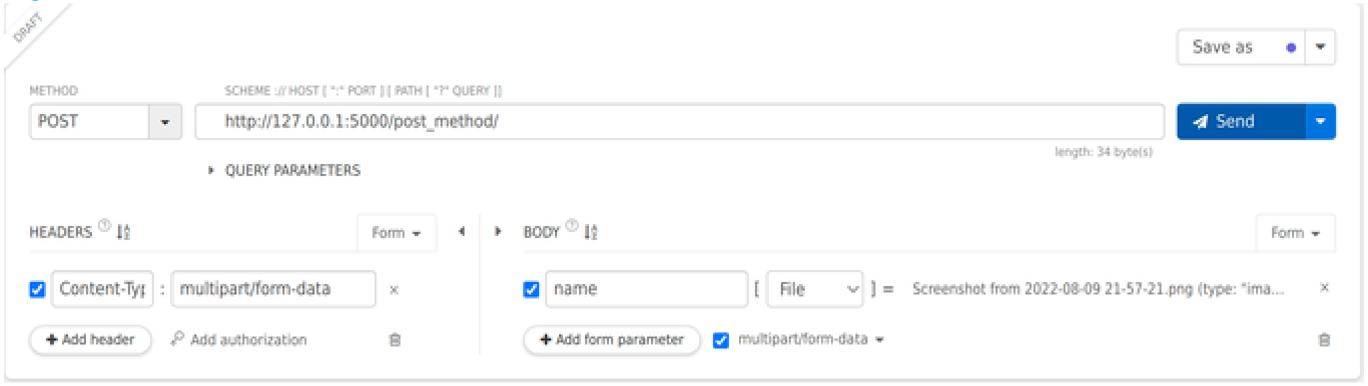
Add form parameter · multipart/form-data

data = request.form.getlist("name")

print(data[0])

Flask File Request

Single File


DRAFT

Save as

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

Send

length: 34 byte(s)

QUERY PARAMETERS

HEADERS: Content-Typ : multipart/form-data

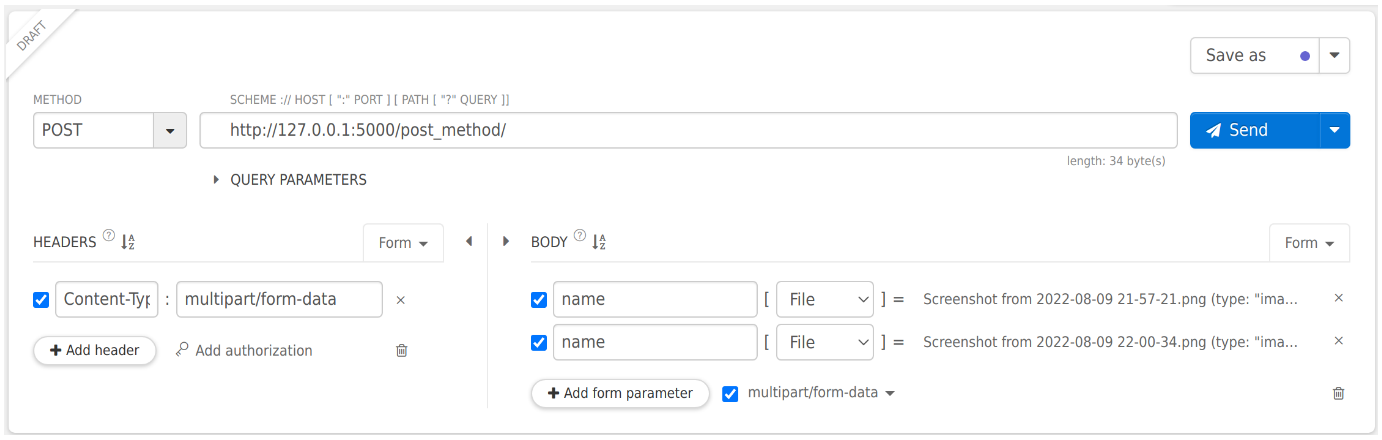
Add header · Add authorization

BODY: name [File] = Screenshot from 2022-08-09 21-57-21.png (type: "ima...

Add form parameter · multipart/form-data

imagefile = flask.request.files.get('name') or imagefile = request.files['name']

Multiple File


DRAFT

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS

Content-Typ	multipart/form-data

BODY

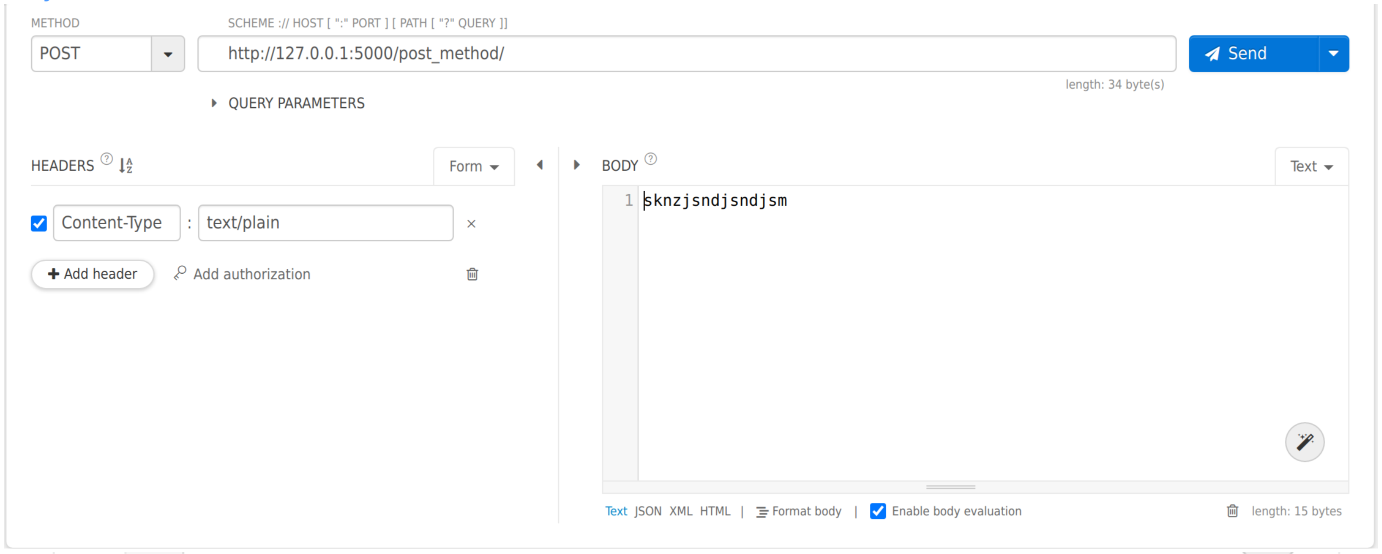
name	File	Screenshot from 2022-08-09 21-57-21.png (type: "ima...
name	File	Screenshot from 2022-08-09 22-00-34.png (type: "ima...

multipart/form-data

```
uploaded_files = request.files.getlist('file')
print(uploaded_files[0].filename)
print(uploaded_files[0].name)
```

Flask Text Request

Body Text


METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

length: 34 byte(s)

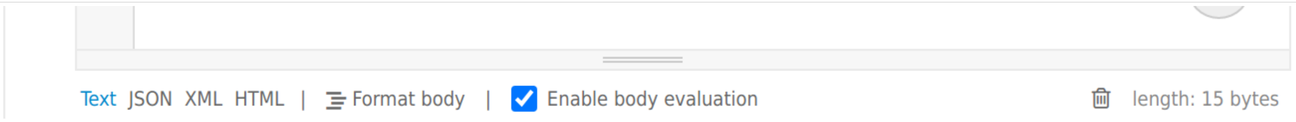
HEADERS

Content-Type	text/plain

BODY (Text)

```
sknzjsndjsndjsm
```

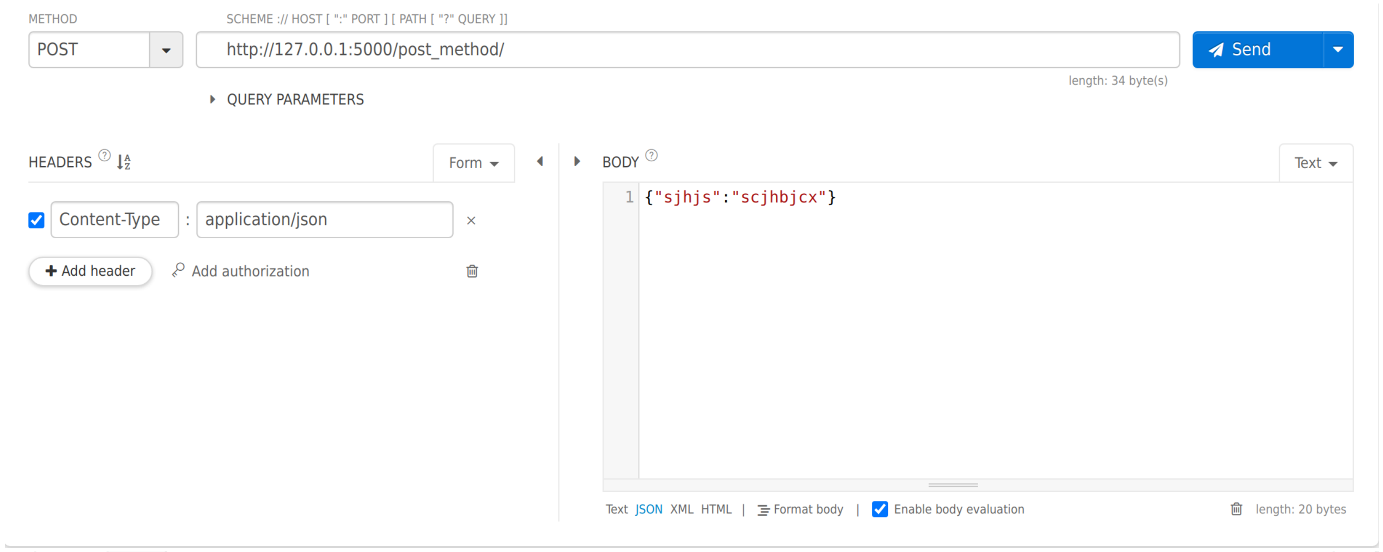
Text JSON XML HTML | Format body | Enable body evaluation length: 15 bytes


Text JSON XML HTML | Format body | Enable body evaluation length: 15 bytes

```
data = request.get_data()

print(data)
```

Flask json Request

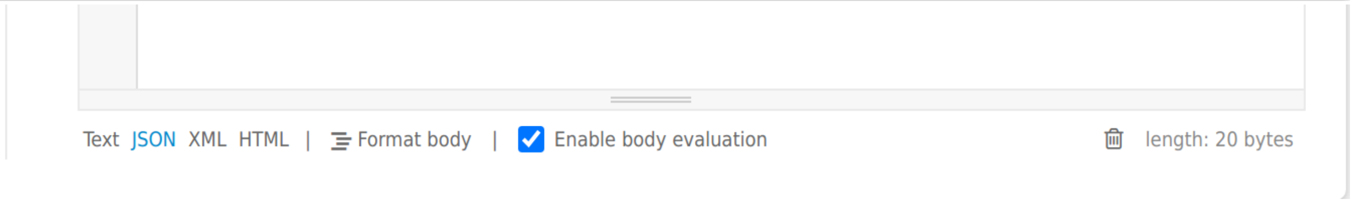
Body Json


METHOD
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
Send
length: 34 byte(s)
QUERY PARAMETERS
HEADERS
Form
BODY
Text
Add header
Add authorization
Text JSON XML HTML | Format body | Enable body evaluation
length: 20 bytes

POST http://127.0.0.1:5000/post_method/

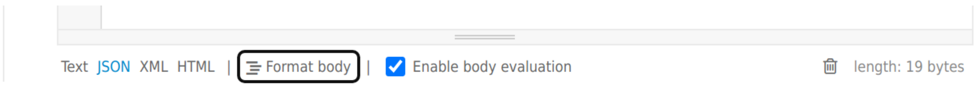
Content-Type : application/json

```
{"sjhjs":"scjhbjcx"}
```



data = request.get_json()
print(data)

click format body button it shows better view



Flask With Mysql RestfulApi

what restful api

https://realpython.com/api-integration-in-python/

HTTP method	API endpoint	Description
`GET`	`/customers`	Get a list of customers.
`GET`	`/customers/<customer_id>`	Get a single customer.
`POST`	`/customers`	Create a new customer.
`PUT`	`/customers/<customer_id>`	Update a customer.
`PATCH`	`/customers/<customer_id>`	Partially update a customer.
`DELETE`	`/customers/<customer_id>`	Delete a customer.

HTTP method	API endpoint	Description
`GET`	`/events/<event_id>/guests`	Get a list of guests.
`GET`	`/events/<event_id>/guests/<guest_id>`	Get a single guest.
`POST`	`/events/<event_id>/guests`	Create a new guest.
`PUT`	`/events/<event_id>/guests/<guest_id>`	Update a guest.

PATCH	/events/<event_id>/guests /<guest_id>	Partially update a guest.
DELETE	/events/<event_id>/guests /<guest_id>	Delete a guest

Flask get post put delete code website

https://www.bogotobogo.com/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php

Flask get post put delete with mysql

https://webdamn.com/create-restful-api-using-python-mysql/

Working Code

```
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json, yaml

app = Flask(__name__)

# Database Connection Information
mydb = mysql.connector.connect(host="localhost",
                               port="3306",
                               user="root",
                               password="root@123",
                               database="face")

# GET REQUEST 1 Method
@app.route('/get_method/', methods=['GET'])
def home():
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor(
            # query = "SELECT * FROM test;"
            # df = pd.read_sql(query, mydb)
            mycursor.execute("SELECT * FROM test;")
            df = pd.DataFrame(mycursor.fetchall())
            jsonfiles = json.loads(df.to_json(orient='records'))
            mycursor.close()
            mydb.close()
            return jsonify(jsonfiles)
        else:
            return jsonify({'data': "no"})

# GET REQUEST 2 Method
@app.route('/get_method/<int:num>', methods=['GET'])
def id_value(num):
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            # query = f"select * from test where
id='{num}';"
            # df = pd.read_sql(query, mydb)
            mycursor.execute(f"select * from test where id='{num}';")
            df = pd.DataFrame(mycursor.fetchall())
            jsonfiles = json.loads(df.to_json(orient='records'))
            mycursor.close()
            return jsonify(jsonfiles)
        else:
            return jsonify({'data': "no"})
# POST REQUEST Method
@app.route('/post_method/', methods=['POST'])
def mew_va():
    mydb.connect()
    if (request.method == 'POST'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"
            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
# PUT REQUEST Method
@app.route('/put_method/', methods=['PUT'])
def put_va():
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            # sql = "INSERT INTO test (id, name, intime, outtime)
VALUES (%s, %s, %s, %s)"
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"
            val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "Employee updated
successfully!"})
            respone.status_code = 200            return respone
        else:
            return jsonify({'data': "no"})
# DELETE REQUEST Method
@app.route('/del_method/<int:num>', methods=['DELETE'])
def del_value(num):
    mydb.connect()
    if (request.method == 'DELETE'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "Deleted successfully!"})
            respone.status_code = 200            return respone
        else:
            return jsonify({'data': "no"})

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

Flask Restful API

```
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json

app = Flask(__name__)

mydb = mysql.connector.connect(host="
localhost",                              port="
3306",                                user="
root",                                password="
root@123",                                 database="face")
@app.route('/all_api/', methods=['GET', 'POST', 'PUT', 'DELETE'])
def new_va():
    if (request.method == 'GET'):
        if not request.args.get('name'):
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute("SELECT * FROM test;")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                mydb.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
        else:
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"select * from test where
id='{num}';")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
    if (request.method == 'POST'):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
    if (request.method == 'PUT'):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"            val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
```

```
                mydb.close()
                respone = jsonify({'data': "updated successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})
        if (request.method == 'DELETE'):
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
                mydb.commit()
                mydb.close()
                respone = jsonify({'data': "Deleted successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})

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

Method 2

Flask Restful API Using from flask_restful import Resource, Api

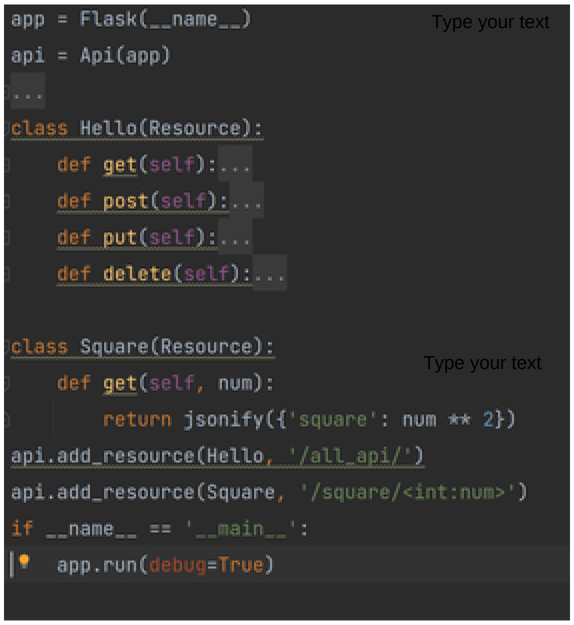
from flask import Flask, jsonify, request
from flask_restful import Resource, Api

```
from flask import Flask, jsonify, request
from flask_restful import Resource, Api
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json

app = Flask(__name__)
api = Api(app)

mydb = mysql.connector.connect(
            host="localhost", port="3306",            user="
root",            password="root@123",            database="
face"        )

class Hello(Resource):
    def get(self):
        if not request.args.get('name'):
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute("SELECT * FROM test;")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                mydb.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
        else:
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"select * from test where
id='{num}';")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
    def post(self):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
    def put(self):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"           val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
```

```
                mydb.close()
                respone = jsonify({'data': "updated successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})
        def delete(self):
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
                mydb.commit()
                mydb.close()
                respone = jsonify({'data': "Deleted successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})

    class Square(Resource):
        def get(self, num):
            return jsonify({'square': num ** 2})

    api.add_resource(Hello, '/all_api/')
    api.add_resource(Square, '/square/<int:num>')

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

Both method different syntax

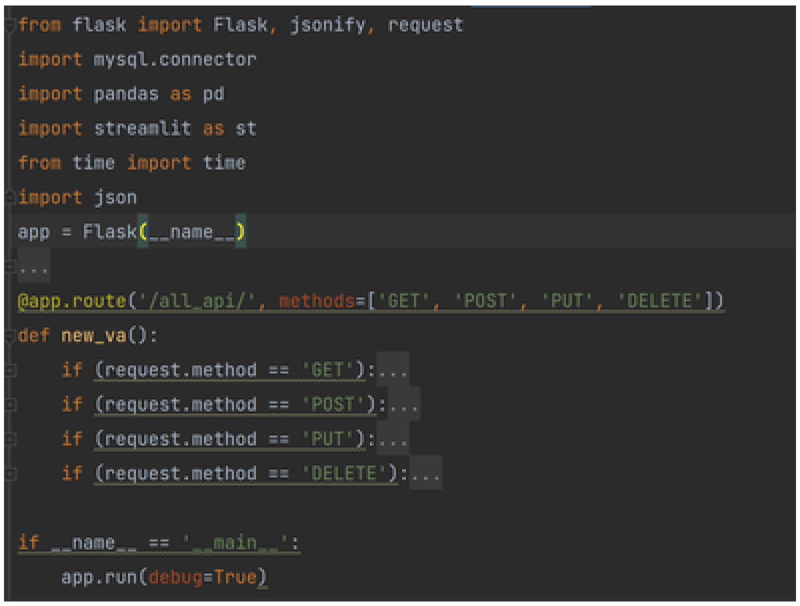
1. Method 2. Method



```
app = Flask(__name__)
api = Api(app)
...
class Hello(Resource):
    def get(self):...
    def post(self):...
    def put(self):...
    def delete(self):...

class Square(Resource):
    def get(self, num):
        return jsonify({'square': num ** 2})
api.add_resource(Hello, '/all_api/')
api.add_resource(Square, '/square/<int:num>')
if __name__ == '__main__':
    app.run(debug=True)
```

Type your text

Type your text



```python
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json
app = Flask(__name__)
...
@app.route('/all_api/', methods=['GET', 'POST', 'PUT', 'DELETE'])
def new_va():
    if (request.method == 'GET'):...
    if (request.method == 'POST'):...
    if (request.method == 'PUT'):...
    if (request.method == 'DELETE'):...

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

Type your text