

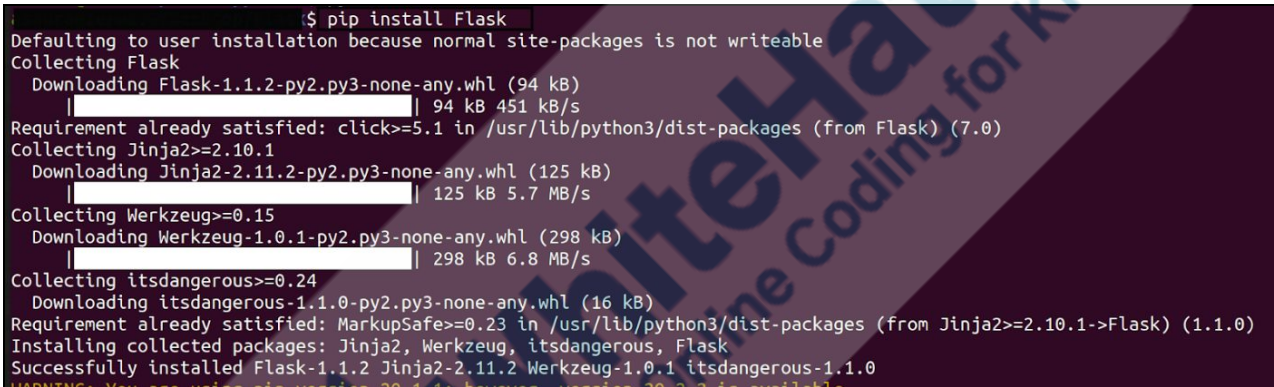
Topic	Flask	
Class Description	Students learn about the flask framework. Students create their first api.	
Class	C124	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Create a basic API • Use postman to test GET and POST Methods 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ VS Code ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources <ul style="list-style-type: none"> ○ VS Code ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 15 min 15 min 5 min
CONTEXT <ul style="list-style-type: none"> • Learning about the API methods and creating our own API 		
Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Hi <Student Name>. How are you doing today? Let's quickly go through what we did in last class.	<i>The student revises the things done in the previous class.</i>

	Have you wondered that whenever you search for something on google you get exactly what you wanted. And how does Google do that?	ESR: Varied!
	Google makes a request on the servers using an API (Application Programming Interface). Api servers as a waiter, it takes your request to the server and gets the information you want and brings it back to you.	-
	Won't it be interesting if we make our own API? We are just going to do that today using flask. And also learn about flask.	ESR: Varied!
Teacher Initiates Screen Share		
<p style="text-align: center;"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Creating a virtual environment. • Installing Flask inside the virtual environment. • Creating a Hello world program. 		
Step 2: Teacher-led Activity (15 min)	So today we are going to be learning about the flask framework. Before that let's understand what is a framework. Can you guess what a framework is?	ESR: Varied
	A framework, in the world of programming, is an additional layer of user written code on top of an already existing language, so that the	<i>Student observes and asks questions.</i>

	language can be used for application specific cases.	
	<p>Now let's understand what flask is. Flask is a framework written on Python. Its library equips us with a lot of useful methods and functionalities, which we can use to do web development. It is easy to get started with as a beginner in web development because you can build a simple application within a couple of lines of code.</p>	<i>Student listens and asks questions.</i>
	<p>Now let's set up a project for flask. To use flask we need:</p> <ol style="list-style-type: none"> 1) latest version of python installed 2) pip 3) Virtual environment 	<i>Student listens and asks questions.</i>
	<p>As we are using python we already have the latest version of python and pip installed and we need to install the virtual environment.</p> <p>Can you tell me what a virtual environment is?</p> <p>Okay so many times, you would find yourself working on different projects at the same time. Maybe one project is using Python3.6 while the other is using Python3.8. On top of that, there might be a different set of modules and libraries you are using in one application, and a different set in another. Managing all this can be confusing, maintaining different versions for different apps, etc. A</p>	<p>ESR: Varied</p> <p><i>Student observes and asks questions.</i></p>

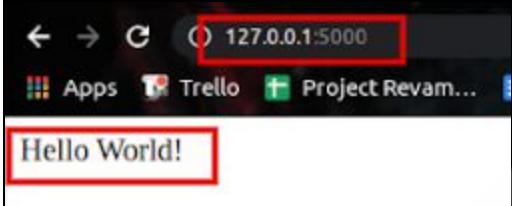
	virtual environment is used to create an isolated environment for your projects. Your projects can have its own set of dependencies, regardless of what other projects might need.	
	<p>Note: In python3.8 virtual environment is preinstalled. And you can directly use python3.8 -m venv venv to create a virtual environment in windows.</p> <p>In Linux and mac you'll have to run a command sudo apt-get install python3.8-venv to install the environment files.</p> <p>Now we will create a virtual environment for our project by running python3.8 -m venv venv.</p> <p>Now here, we are saying that we want to create a virtual environment, which we will call venv and we want this virtual environment to have Python3.8 by default.</p> <p>This -m tells python the name of the module we want to use to perform this action.</p> <p><i><Teacher codes to create a new folder called flask application and navigates inside the folder and creates a virtual environment using python3.8 -m venv venv.</i></p> <p><i>If using Mac or Linux run the</i></p>	-

	<p><i>command <code>sudo apt-get install python3.8-venv</code> and then <code>python3.8 -m venv venv</code>></i></p>	
 <pre>[sudo] password for ashura: Sorry, try again. [sudo] password for ashura: Reading package lists... Done Building dependency tree Reading state information... Done The following packages were automatically installed and are no longer required: libfprint-2-tod1 libllvm9 python3-click python3-colorama Use 'sudo apt autoremove' to remove them. The following NEW packages will be installed: python3.8-venv 0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded. Need to get 5,288 B of archives. After this operation, 27.6 kB of additional disk space will be used. Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python3.8-venv amd64 3.8.2-1ubuntu1.2 [5,288 B] Fetched 5,288 B in 0s (17.2 kB/s) Selecting previously unselected package python3.8-venv. (Reading database ... 411568 files and directories currently installed.) Preparing to unpack .../python3.8-venv_3.8.2-1ubuntu1.2_amd64.deb ... Unpacking python3.8-venv (3.8.2-1ubuntu1.2) ... Setting up python3.8-venv (3.8.2-1ubuntu1.2) ... \$ python3.8 -m venv venv</pre>		
	<p>Now that our virtual environment is set up, we will just activate this virtual environment, so that whatever dependency or module we install, we install it in this virtual environment only. Once we activate this, it will start taking the python and pip versions in this shell from whatever is installed inside this virtual environment we created.</p> <p>To activate it :-</p> <p>For linux and mac use:</p> <pre>source env/bin/activate</pre> <p>For windows use:</p> <pre>.\env\Scripts\activate</pre>	<p><i>Student observes and asks questions.</i></p>

	<p>We first need to run pip install flask to install Flask into the Virtual Environment that we just created.</p> <p><i><Teacher codes to install flask inside a Virtual Environment></i></p> <p>Code:</p> <p>pip install Flask</p>	
		
	<p>Now that flask is installed we'll create our first Hello World application using flask.</p> <p><i><Teacher opens the VS Code editor and opens the flask application folder inside it.></i></p> <p>Now, let's create a file called app.py, where we will write a basic Hello, World! Application, just to get started.</p> <p><i><Teacher creates a app.py file ></i></p> <p>The first thing we'll do in this file is that we will import Flask.</p>	-

	Code: from flask import Flask	
		
	<p><Teacher codes to write <code>app = Flask(__name__)></code> code: <code>app = Flask(__name__)</code></p> <p>Now when we say <code>app = Flask(__name__)</code>, the constructor of this class Flask that we imported takes the name of the current module, stored in <code>__name__</code> as an argument and the variable <code>app</code> is now a Flask object.</p>	<p><i>Student listens and asks questions.</i></p>
		
	<p>Now let's define a small route. Code: <code>@app.route("/")</code> <code>def hello_world():</code> <code>return "Hello World!"</code></p> <p>Here, the <code>route()</code> function of this Flask class is a decorator, which will tell our web app which URL endpoint should call the associated function, <code>hello_world</code> in this case.</p>	<p><i>Student observes and asks questions.</i></p>

	<p>We can also provide this route function with different kinds of request methods, but we will get into that later.</p> <p>Here, the URL '/' is now bound with the function <code>hello_world()</code></p>	
<pre>@app.route("/") def hello_world(): return "Hello World!"</pre>		
	<p>Finally we write the following code to run our web application.</p> <p>Code:</p> <pre>If __name__ == '__main__': app.run()</pre> <p>Here, when we run this file now, it will run this web application we just built on our local server on port 5000, and when we go to the homepage of the web server, the output of the <code>hello_world</code> function will be rendered.</p> <p>We can run this file the same way as any other file. We just have to write <code>python app.py</code> and voila, our server is up and running!</p>	<p><i>Student observes and asks questions.</i></p>
<pre>if (__name__ == "__main__"): app.run(debug=True)</pre>		

	<p>Now let's check the output. <i><Teacher runs the code normally on the shell></i> We can see that a development server gets opened. And our code is running in port 5000 by default. <i><Teacher clicks on the link ></i> What happened when I clicked on the link?</p>	<p>ESR: A tab in the browser opened and we could see hello world written there. And the code was running on port 5000.</p>
		
	<p>Perfect. So as we saw here we created our own web API which returned "Hello World!". Now let's create an API which will keep a record of our tasks and also where we can add them and view them. Are you up for it?</p>	<p>ESR: Yes!</p>
	<p>Let's get started.</p>	
<p>Teacher Stops Screen Share</p>		
	<p>Now it's your turn. Please share your screen with me.</p>	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 		

ACTIVITY

- Learning about the POST Method
- Using postman to check the API Methods

Step 3: Student-Led Activity (15 min)

As we know API is Application Programming interface, which is a software intermediary that allows two applications to talk to each other. All the websites, mobile apps, etc. that you may use, or instant messaging apps to send messages, use an API.

All APIs use an HTTP method. Let's see what they are:

The **GET** Method -
GET is used to request data from a specified resource. When you access a website's page, your browser makes a get request to your API and your API is returning the front-end that is displayed in the browser.

All the APIs that you create in Flask, by default, use GET Request. If you start your hello_world server and go to localhost:5000, you can see in your terminal that the browser made a GET request on the server/API.

The **POST** Method -
POST is used to send data to the server to create/update a resource.

A user wants to sign up from the signup page? A user wants to change

Student listens and asks questions.

	<p>their password? That would be a post request.</p> <p>The PUT Method - PUT is used to send data to a server to create / update a resource.</p> <p>The basic difference between PUT and POST is that a POST request is when you can create multiple copies of the same resource.</p> <p>A PUT request, on the other hand means that you want to create only one copy of the resource, i.e. Signing Up a unique user.</p> <p>The DELETE Method - DELETE is used to delete a resource.</p> <p>The default method used by flask is GET.</p>	
	<p>So now let's start building our API. Before that let's quickly set up our project file.</p> <p><i><Teacher helps student to create a folder called flask.</i></p> <p><i>The teacher helps the student to create a virtual environment inside the folder.</i></p> <p><i>The teacher helps the student to install flask inside the folder.</i></p> <p><i>Teacher helps student to open the folder in VSCode and create a app.py file></i></p>	<p><i>Student creates a folder called flask.</i></p> <p><i>The student creates a virtual environment inside the folder.</i></p> <p><i>Student installs flask inside the folder.</i></p> <p><i>Student opens the folder in VSCode and creates an app.py file.</i></p>

```
[sudo] password for ashura: [sudo apt-get install python3.8-venv]
[sudo] password for ashura:
Sorry, try again.
[sudo] password for ashura:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm9 python3-click python3-colorama
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  python3.8-venv
0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded.
Need to get 5,288 B of archives.
After this operation, 27.6 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python3.8-venv amd64 3.8.2-1ubuntu1.2 [5,288 B]
Fetched 5,288 B in 0s (17.2 kB/s)
Selecting previously unselected package python3.8-venv.
(Reading database ... 411568 files and directories currently installed.)
Preparing to unpack .../python3.8-venv_3.8.2-1ubuntu1.2_amd64.deb ...
Unpacking python3.8-venv (3.8.2-1ubuntu1.2) ...
Setting up python3.8-venv (3.8.2-1ubuntu1.2) ...
```

```
$ python3.8 -m venv venv
```

```
$ pip install Flask
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
    | 94 kB 451 kB/s
Requirement already satisfied: click>=5.1 in /usr/lib/python3/dist-packages (from Flask) (7.0)
Collecting Jinja2>=2.10.1
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
    | 125 kB 5.7 MB/s
Collecting Werkzeug>=0.15
  Downloading Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
    | 298 kB 6.8 MB/s
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/lib/python3/dist-packages (from Jinja2>=2.10.1->Flask) (1.1.0)
Installing collected packages: Jinja2, Werkzeug, itsdangerous, Flask
Successfully installed Flask-1.1.2 Jinja2-2.11.2 Werkzeug-1.0.1 itsdangerous-1.1.0
```

First we'll need to import Flask, jsonify and request from flask to our code file.

Teacher helps the student to import Flask, jsonify and request from flask.

Code:

```
from flask import Flask,jsonify,
request
```

The student imports Flask, jsonify and requests from flask.

```

app.py
1  from flask import Flask,jsonify, request
2

```

Then we say `app = Flask(__name__)`.
And create dummy tasks data with an id, title, description and done value which would be true or false.

<Teacher helps student with the code and to create a dummy data.>

Code:

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

#creating a array of tasks with each task as a different object in it.

```

tasks = [
    {
        'id': 1,
        'title': u'Buy groceries',
        'description': u'Milk, Cheese,
Pizza, Fruit, Tylenol',
        'done': False
    },
    {
        'id': 2,
        'title': u'Learn Python',
        'description': u'Need to find a
good Python tutorial on the web',
        'done': False
    }
]

```

Students code to create a flask object and create a dummy data.

```

2
3 app = Flask(__name__)
4
5 tasks = [
6     {
7         'id': 1,
8         'title': u'Buy groceries',
9         'description': u'Milk, Cheese, Pizza, Fruit, Tylenol',
10        'done': False
11    },
12    {
13        'id': 2,
14        'title': u'Learn Python',
15        'description': u'Need to find a good Python tutorial on the web',
16        'done': False
17    }
18 ]
19

```

Now, we will create a similar looking function as our `hello_world()`, and we would name it as `add_data()`.

In the `add_data()` function, we will change the route from `'/'` to `'add-data'`. This would mean that this function would only be available at route `localhost:5000/add-data`.

We'll also specify the method that we want to have on the route. We can do that by adding `methods=["POST"]` to the line.

Here, we are explicitly telling Flask that we want this function on this route to only allow POST Request.

Code:

```
@app.route("/add-data",
methods=["POST"])
```

Student codes to create a route with a post method.


```
@app.route("/add-data", methods=["POST"])
```

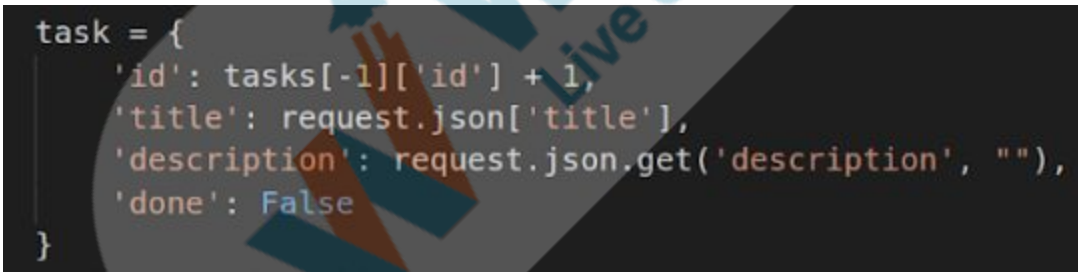
In this post request we'll create an `add_task()` function which will check for the data and if it doesn't find any then it'll show the 400 error with a message to provide data and allow us to add data to the API and to the tasks array of objects.

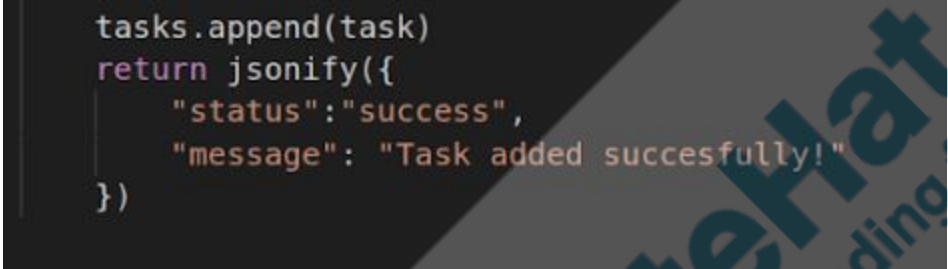
Code:

```
def add_task():
    if not request.json:
        return jsonify({
            "status": "error",
            "message": "Please provide
the data!"
        },400)
```

The student codes to create a function which checks for the data and shows the 400 error with a message to provide data.

```
@app.route("/add-data", methods=["POST"])
def add_task():
    if not request.json:
        return jsonify({
            "status": "error",
            "message": "Please provide the data!"
        },400)
```


	<p>Now that we have to add a task, we'll create a skeleton/ structure of how the task will look like.</p> <p>Our task will be an object/dictionary which will have multiple keys such as Id, title of the task, description of the task and the status of the task.</p> <p>We'll automate the id to increment by 1 every time a new task is added and keep the initial status of the task as false.</p> <p>Title and description will be provided by the user.</p> <p>Code:</p> <pre>task = { 'id': tasks[-1]['id'] + 1, 'title': request.json['title'], 'description': request.json.get('description', ""), 'done': False }</pre>	<p><i>Student codes to create the structure/ skeleton of the task.</i></p>
		
	<p>As the user provides the title and description we have to add the task to the main list of tasks and then show the message - "task added successfully".</p> <p>Code:</p> <pre>tasks.append(task) return jsonify({</pre>	<p><i>Student codes to add the task to the main tasks list and return a message.</i></p>

	<pre>"status": "success", "message": "Task added successfully!")</pre> <p>Here we are converting the message to a json object and then showing it. We are doing it because most of the time the data we get is in a Json format.</p>	
		
	<p>So now our post method is ready. We also have to create a get method to see all the available tasks. We have already seen how to create a GET method, this time we'll add a route name to it.</p> <p>Code:</p> <pre>@app.route("/get-data") def get_task(): return jsonify({ "data" : tasks)</pre>	<p><i>Student codes to create a GET method with route name which will return all the tasks in the list.</i></p>

```
@app.route("/get-data")
def get_task():
    return jsonify({
        "data" : tasks
    })
```

And finally we'll code to run our web application.

Code:

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

Here we are keeping debug=True so that the server will reload every time you make some change to the code.

Student codes to reload the server every time a change is being made and run the web application.

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

To test our APIs excluding GET i.e (POST, PUT, and DELETE) we need to install a software known as POSTMAN.

Teacher helps student to install postman using command sudo snap install postman or install it from Student Activity 1. And create an account on it.

Student installs the postman using command sudo snap install postman.

Or install it using Student Activity 1

Student creates an account on the postman.

```
~$ sudo snap install postman
[sudo] password for ashura:
postman 7.32.0 from Postman, Inc. (postman-inc*) installed
```

Or download from the site.

Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

[Download the App](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

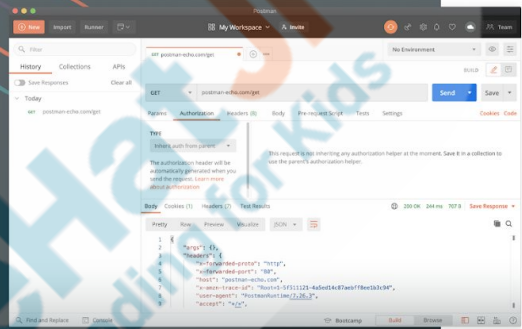
Version 7.32.0 | [Release Notes](#) | [Product Roadmap](#)

Not your OS? Download for Windows (x32 / x64) or Mac (macOS)

Postman on the web

You can now access Postman through your web browser. Simply create a free Postman account, and you're in.

We use cookies to analyze traffic, track and improve your experience, and assist in our marketing efforts. Our Cookie



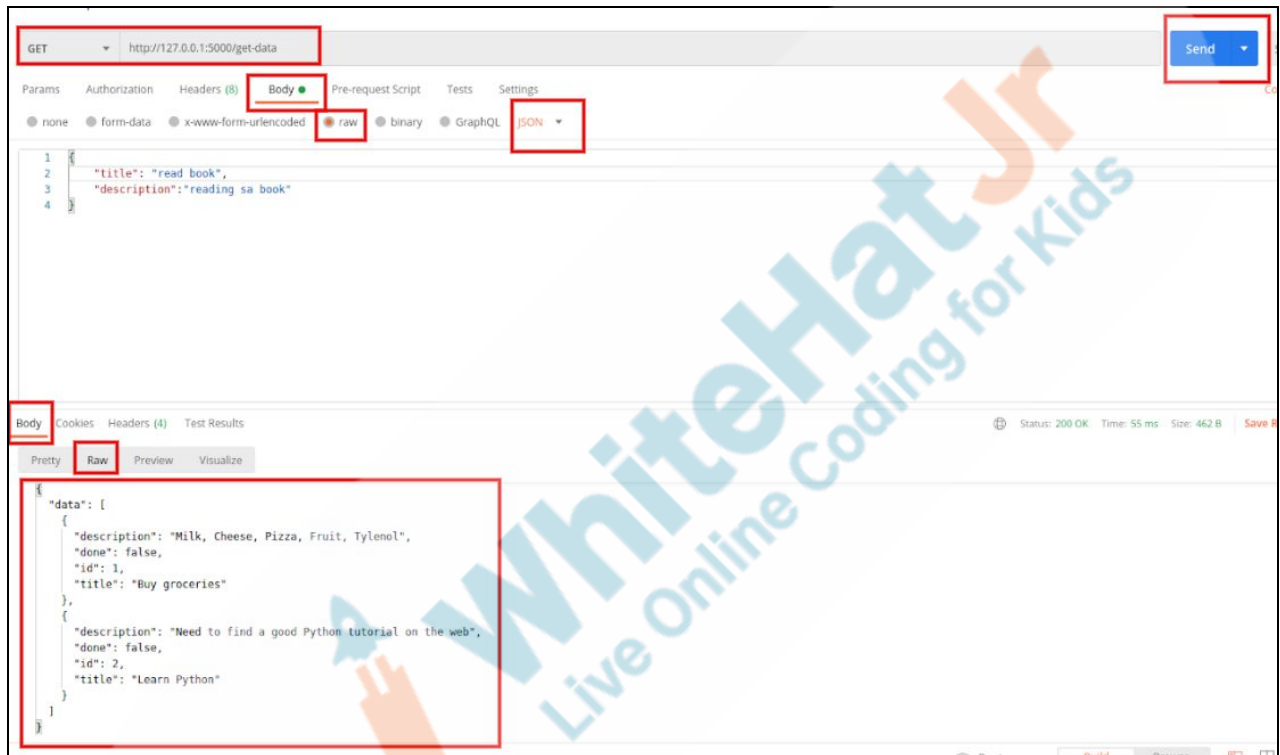
Alright now let's test our GET and POST methods.
Teacher helps the student to run the code and copy the link.

Student runs the code and copies the link.

```
~/Desktop/Flask$ python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 115-451-367
```

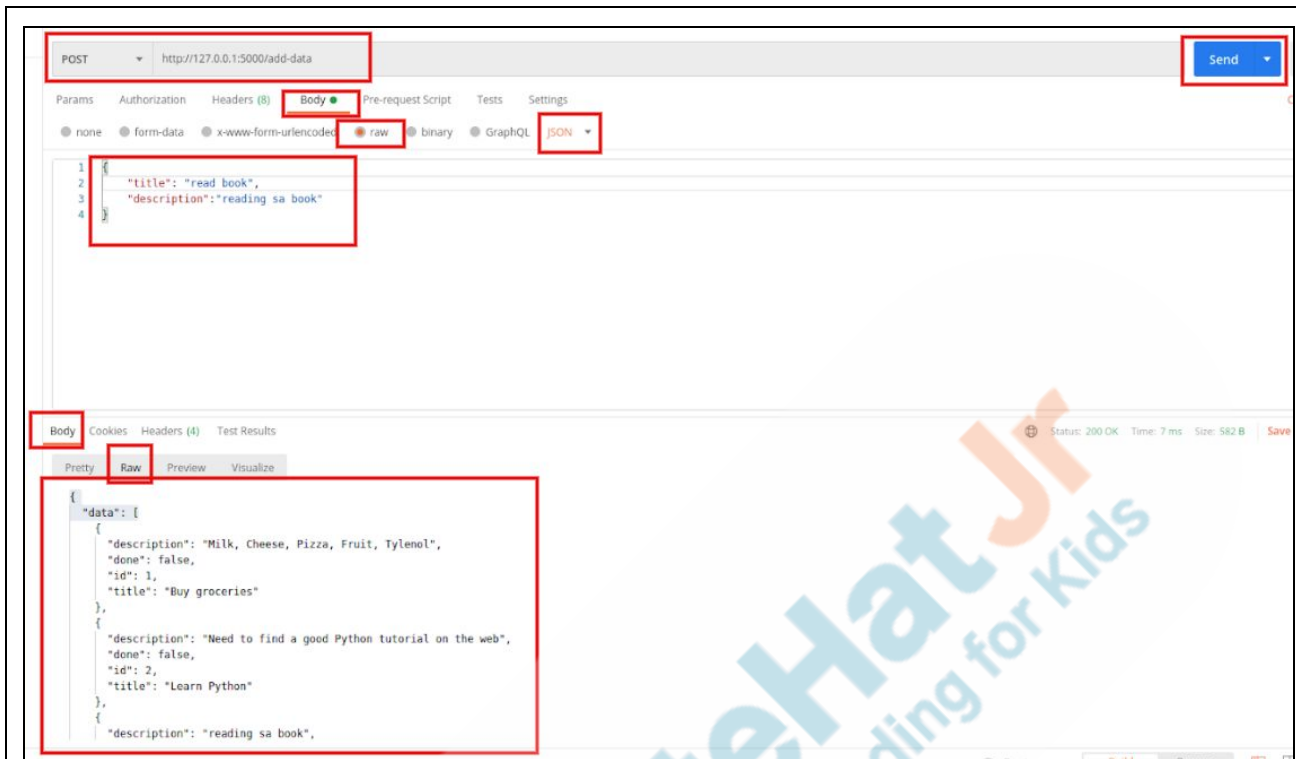
Now we paste the link in the postman and set the method to GET, add /get-data to the route and send the request.
Below we should be able to see the output.

Student pastes the link in the postman and sets the method to GET, add /get-data to the route and sends the request and checks the output below.



Now let's check our POST method. We'll set the method to POST and add /add-data to the route and send the request.

Student codes to check the POST method.



Teacher Guides Student to Stop Screen Share

FEEDBACK

- Appreciate the student for their efforts
- Identify 2 strengths and 1 area of progress for the student

Step 4: Wrap-Up (5 min)

Let's quickly revise what we did today.

The student revises the concepts covered in the class.

	Now that we have learned to create the API we'll integrate it to our react native app to create an awesome project. Sounds interesting ?	ESR: Yes
	See you in the next class then.	
<div>Teacher Clicks</div> <div>✕ End Class</div>		
Additional Activities	<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> - Describe what happened - Code I wrote • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	<p><i>The student uses the markdown editor to write her/his reflection in a reflection journal.</i></p>

Activity	Activity Name	Links
Teacher Activity 1	Solution	https://github.com/whitehatjr/Flask
Student Activity 1	Postman link	https://www.postman.com/download/s/

