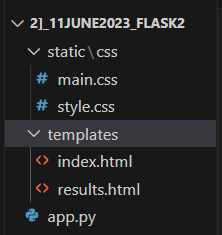
In the previous lecture, we studied how data can be taken from the user and to give results based on that data. This data can also be gathered with the help of **FORMS.**

* Open a Flask Project. And create a Folder named “templates” in the location where app.py is located.
* Inside the “templates” folder, create files “index.html. And “results.html”.
* Also create a folder “static” and place it where the “app.py” and “templates” folder is located.
* Inside the “static” folder, and inside the static folder, create another folder named “css”.
* Inside the “css” folder, create two files -> “main.css” and “style.css”

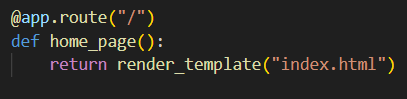


Now we are going to create a form. This form will allow users to input numbers and operations like add, subtract, divide, etc.

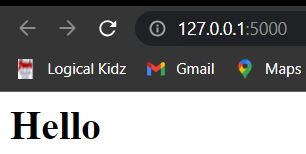
The required function should take place and the result should be displayed to the user.

* The index.html page will contain the main format of the FORM and the input will be taken with its help.



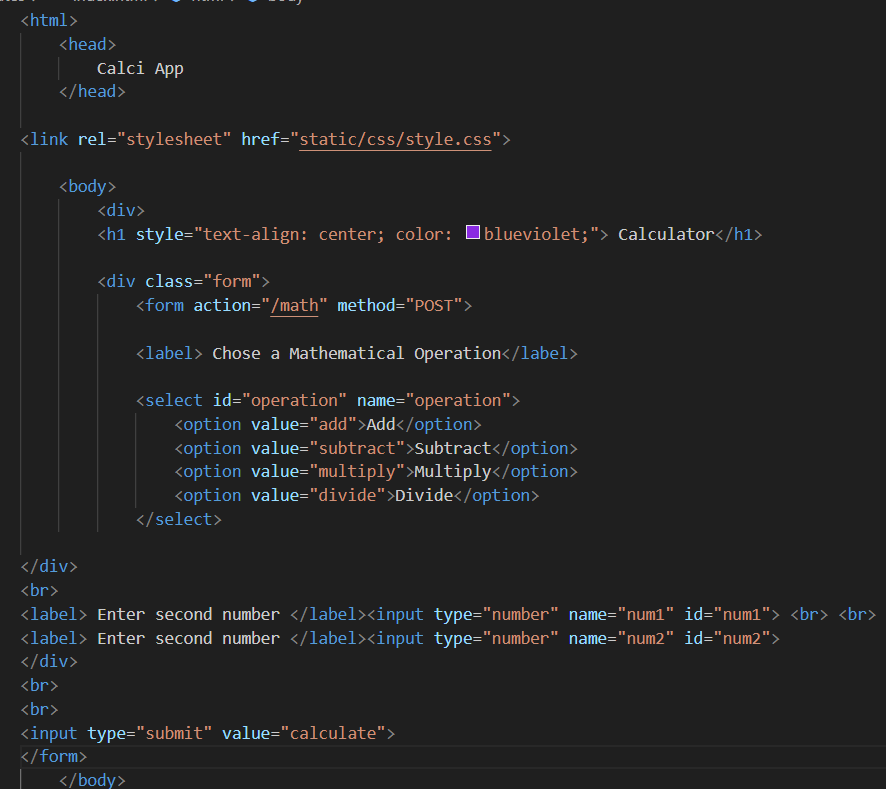


The above piece of code will display index.html on the webpage when the binded url is used.

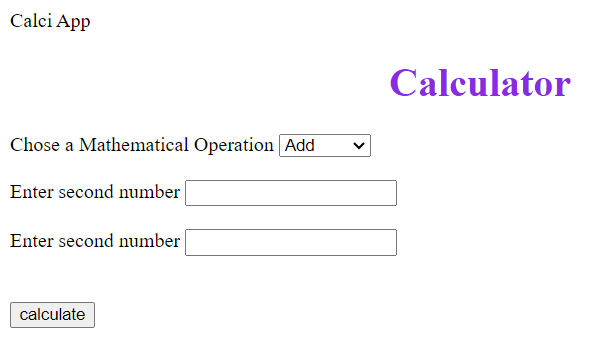
Hence, in this way, index.html will be displayed on web page

Now, we will create index.html in such a way that it represents a calculator type design.



Form action will contain the main action for the form.

method = POST means that we are trying to input data without showing it on the URL.



When the button “Calculate” is pressed, the form action will take place and hence, the function binded with “ \main” will be called.

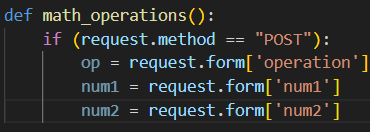
Now, we have successfully built the index.html page and also rendered it. Now, we want to work on the operation/ working.

* if (request.method == "POST"):

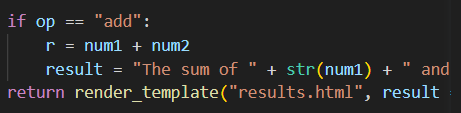
This means -> (If the data is being inputted using POST method i.e through the FORM and not through URL.

* For the GET method(unsecure) , we used request.args.get to accept the input from the user.
* For the POST method(secure), we will use request.form[‘operation’].

Here operation is the id of a field.



Now, after accepting the input, we will perform necessary actions.



Do similar for subtraction, multiplication and division.



1. GET(unsecure) -> passing data through URL. Data is public as it is visible in URL

Eg -> Google Search

1. POST(secure) -> passing data through FORM. Data is private as it is not visible in URL

Eg -> Email Login

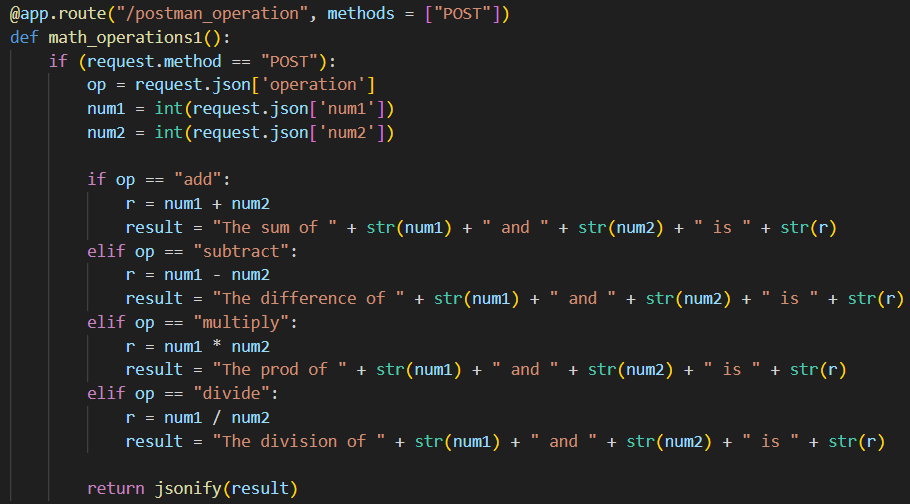
**POSTMAN** is a software used for API testing. The API can be POST or GET.

Now, convert your operation function() and take the input not in the form of form, But in the form of json.

Eg -> op = request.json[‘operation’]

operation will be key and value will be the value inputted in op.

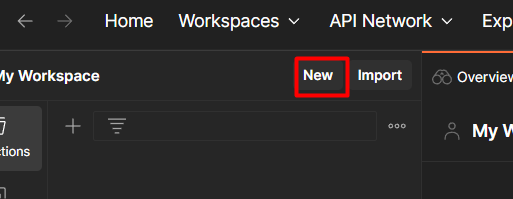
Also, we will try to obtain the final result in the form of json instead of rendering results.html. Hence, we use the jsonify() method.



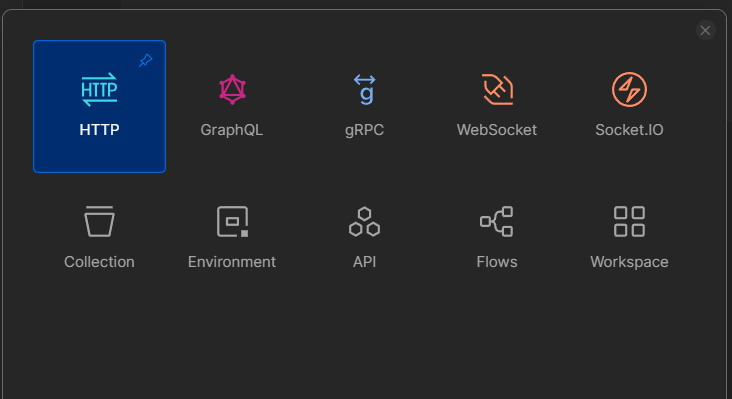
In this way, we can edit our function so as to perform api testing.

Now open the postman software.

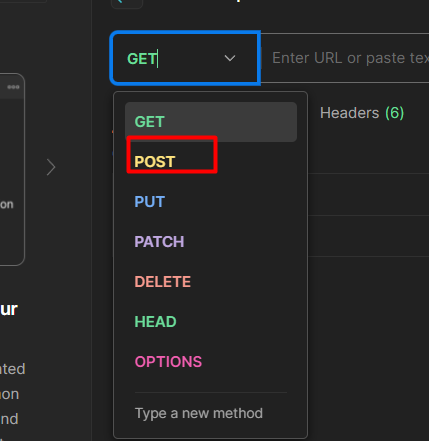
Click on “New”



After that, we have to select which type of API we have to test. Now, we want to test http. Hence, we select Http.



After that, select your method .i.e GET or POST. We select POST here.



After that, we have to put the url in the postman. Hence, copy the local url <http://127.0.0.1:5000/> and then append the text you have given in @app.route() which is binded with the function which you want to test. Here, it is @app.route(/postman\_operation, methods = [“POST”]).

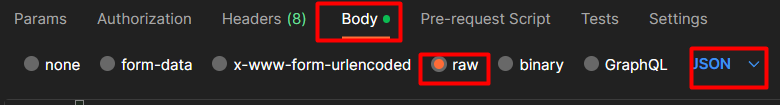
Hence, the final URL to put in postman here will be ->

<http://127.0.0.1:5000/postman_operation>

But, our work is not over here. We also have to give input .i.e operation and numbers.

As, we can see that in the new code, we have taken input in the form of json, we have to give input in postman in the form of json too .i.e in the form of dictionary.

Hence, click on Body and select option Raw and then from drop\_down option, select JSON.



Then, click on send to test our API. In this way, our API testing was successful.

