Web dev:

Cs5o slides and their videos related info like slides used,notes,subtitles etc

https://cs50.harvard.edu/x/2023/weeks/9/

pscale\_pw\_ujdR6u06sXoDQ29xSmKZUmVsR6qgtG5JMCphtPaMkUO

pscale\_pw\_78nZnFH7vyt4APvb4VcRV6XUozvYN0WOfO0l4ZKeFJ6

mysql -h aws.connect.psdb.cloud -D hansifdb -u i8jfld1x06ikjqyo93yt -ppscale\_pw\_78nZnFH7vyt4APvb4VcRV6XUozvYN0WOfO0l4ZKeFJ6 --ssl-mode=VERIFY\_IDENTITY --ssl-ca=/etc/ssl/cert.pem

For certification in responsive web design:

<https://www.freecodecamp.org/learn/2022/responsive-web-design/>

hosting links:

https://youtu.be/UwvAMkH9tTU?si=YsJxwdaPctyKBGQC

chagpt. ; https://chat.openai.com/share/6fe528fd-066b-4fb7-b57e-a9d52d028233

render: <https://www.youtube.com/watch?v=pg11wmj8LbY>

Heroku : <https://youtu.be/D2GLVoiEZyE?si=kjvkGY7hV_ZwgEu3>

Cyclic: <https://www.youtube.com/watch?v=q8GSWGu2roA>

imp command:

* flask run --port=PORT\_NUMBER 🡺changes the port number

ex:

flask run --port=30000

IMP POINTS:

1. Route() is called as decorator
2. export FLASK\_APP=flask\_app\_file\_name (syntax to tell the flask server to run the particular flask file when “flask run” command is executed
3. Usage of [“app.route('/hello/<name>')”](#_Usage_of_)
4. Can run flask app with following commands

11)python app\_name.py

22)export FLASK\_APP=app\_name

Flask run

1. DATATYPED SUPPORTED IN DYNAMIC URL:
2. [Then “topic of jinja2”](#_JINJA_2)

Notes:

@app.route('/')

*def* index():

*return* render\_template("index.html")

@app.route("/hello")

*def* abc():

name=request.args.get("hello")

render\_template("abc.html",x=name)

*if* \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

if u see above :

1)u have app.route(“/”) and app.route(/hello) here it doesn’t mean that if run the app and there in the url u explicity give like 5000/?hello=dsafas it doent go to abc() function it goes to index() function only bcz while u run by default url be 5000/ again if explicitly change url its doent change the route in order to change the route(in the change the set of things to be done or simply function) there should be some implicit action in the page that could change the url and navigate to specific route as in the below example where we will make use of action button which change the url implicitly

**ACTION ATTRIBUTE:**

It makes sure that on the form submit the section action to be done like if we give the app.route path then the specified route function is triggered

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  </head>  <body>  <form action="/fcuk" method="get">  <input type="text" name="hansi1111" value="das">  <input type="text" name="n2" value="n3">  <input type="submit">  </form>  </body>  </html> |

\*in code for every type we gave “name” na that will will used to form url let us see with

Snippet ex:

<form action="" method="get">

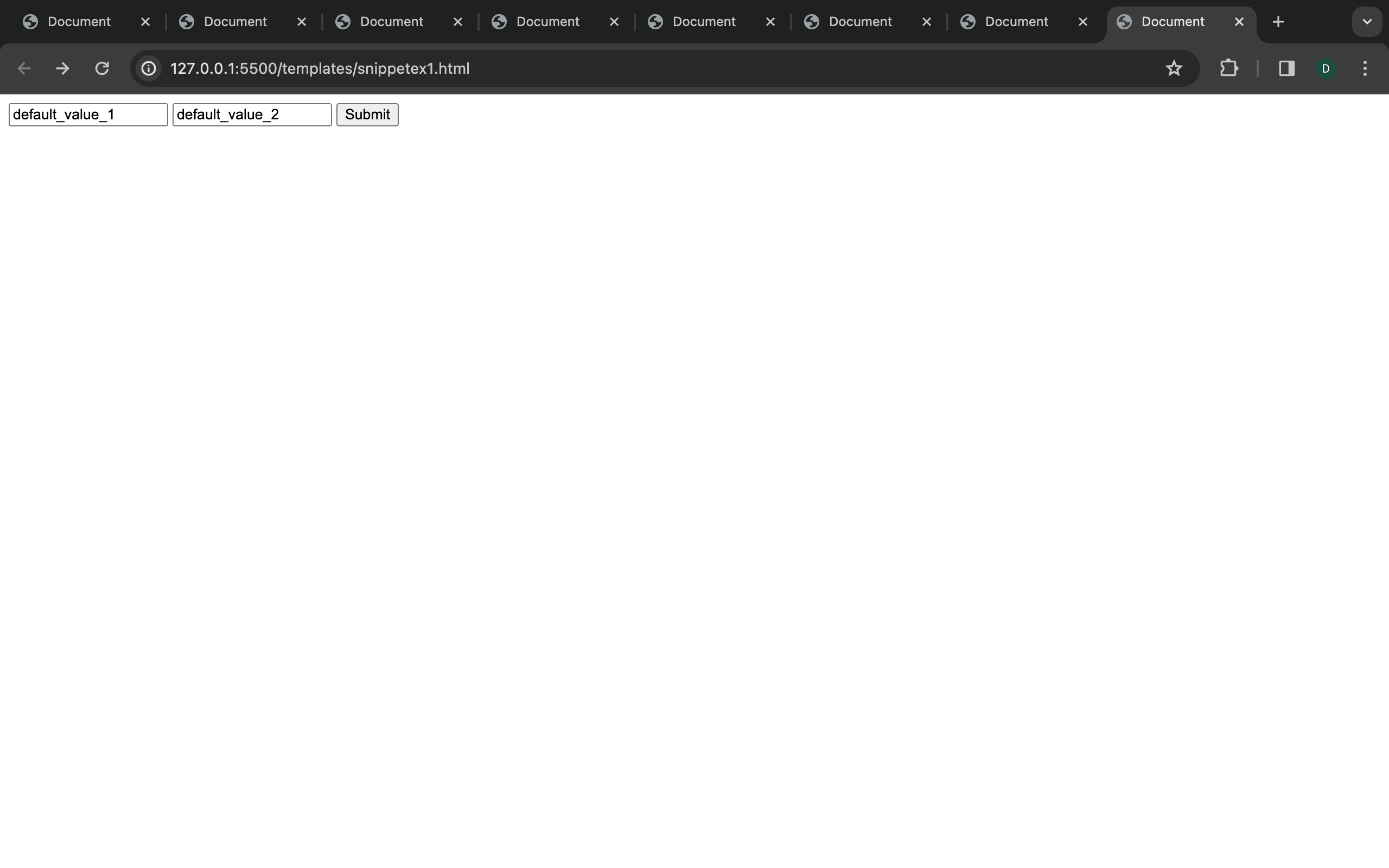
<input type="text" name="urlformationname\_1" value="default\_value\_1">

<input type="text" name="urlformationname\_2" value="default\_value\_2">

<input type="submit">

</form>

**SCENARIO 1:**

IF U DIDN’T ENTER UR VALUES AND KEEPS DEFAULT VALUES ONL

Then if PRESS SUBMIT THE URL CHANGES AS BELOW

|  |
| --- |
| http://127.0.0.1:5500/templates/snippetex1.html?urlformationname\_1=default\_value\_1&urlformationname\_2=default\_value\_2 |

**SCENARIO 2:**

IF U ENTER DIFF VALEUS AND SUBMIT



Then

|  |
| --- |
| http://127.0.0.1:5500/templates/snippetex1.html?urlformationname\_1=X&urlformationname\_2=Y |

SCENARIO 3:

If the action button has specific route

|  |
| --- |
| <form action="action\_1" method="get">  <input type="text" name="urlformationname\_1" value="default\_value\_1">  <input type="text" name="urlformationname\_2" value="default\_value\_2">  <input type="submit">  </form> |

Then the path:

|  |
| --- |
| http://127.0.0.1:5500/templates/action\_1?urlformationname\_1=x&urlformationname\_2=y |

NOW LET US GO BACK TO THE EXAMPLE OF IMPLICIT CHANGING OF URL BY ACTION PAGE

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/fcuk" method="get">

<input type="text" name="hansi" value="das">

<input type="text" name="n2" value="n3">

<input type="submit">

</form>

</body>

If we see here on submitting the form the action done is triggering the function in app.py associated with “/fcuk” route as in below ex where then name hansi value is being caught by request.args.get() method and and assigned to a variable where that variable is passed on to other html page and utilized there

If in request.args.get() method if u try to get other than name of input tag like request.args.get(“dsfds”) where name of input tag is “hansi” in the html for above u might not get value

*from* flask *import* Flask,render\_template,request

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

@app.route("/")

*def* basic():

*return* render\_template("index1.html")

*#ensure that the form has built in the html page with action as below route path anem*

@app.route("/fcuk")

*def* hello():

name=request.args.get("hansi")

*return* render\_template("onsubmitindex1.html",hansi=name)

the other html page is onsubmitindex1.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

hello buddy {{hansi}}

</body>

</html>

Ex of getting two inputs from form and displaying those values in other html page:

Firsr. Index1.html;

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/fcuk" method="get">

<input type="text" name="hansi" value="das">

<input type="text" name="n2" value="n3">

<input type="submit">

</form>

</body>

</html>

Next the app1.py

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/fcuk" method="get">

<input type="text" name="hansi" value="das">

<input type="text" name="n2" value="n3">

<input type="submit">

</form>

</body>

</html>

The page that is going to display on pressing submit button in index1.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

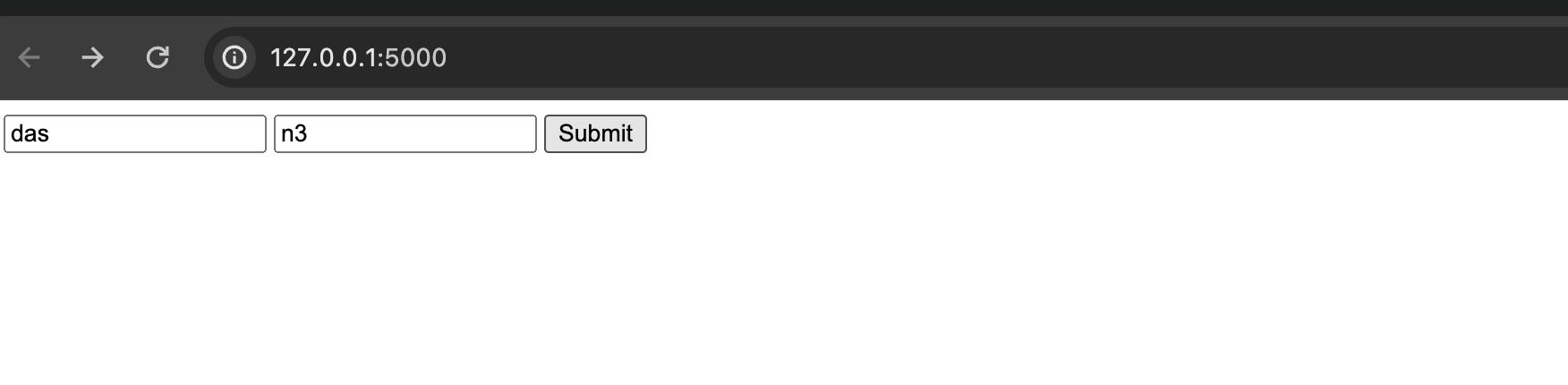
<body>

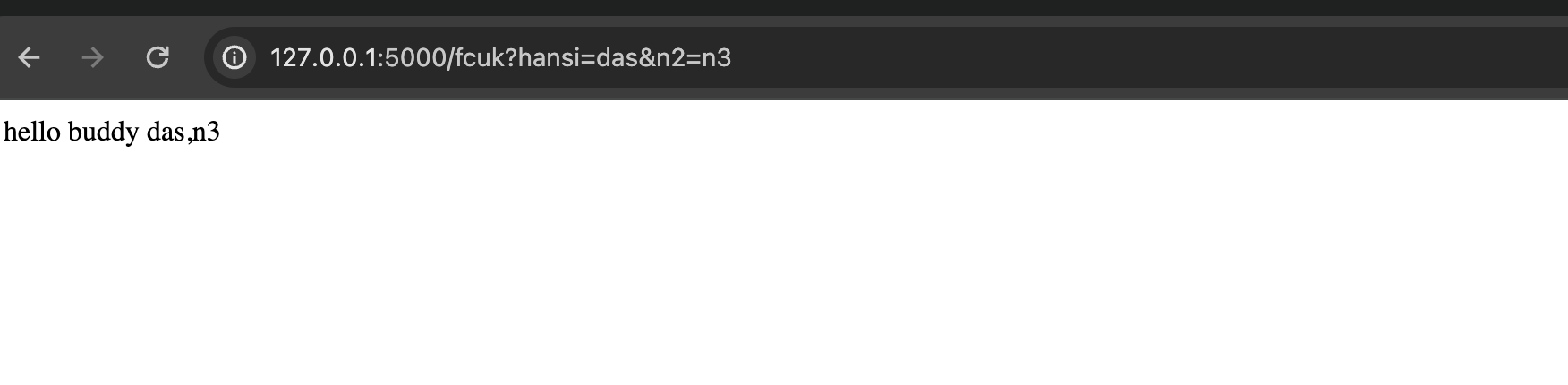
hello buddy {{hansi}},{{passwd}}

</body>

</html>

Output





ERROR WHEN: we use post method directly in index to hide credential without adding POST to route methods

Index2\_POST.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form action="/fcuk" method="post">

<input type="text" name="hansi" value="das">

<input type="text" name="n2" value="n3">

<input type="submit">

</form>

</body>

</html>

App3\_post.py

*from* flask *import* Flask,render\_template,request

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

@app.route("/")

*def* basic():

*return* render\_template("index2\_POST.html")

*#ensure that the form has built in the html page with action as below route path anem*

@app.route("/fcuk")

*def* hello():

name=request.args.get("hansi")

pwd=request.args.get("n2")

*return* render\_template("onsubmitindex1.html",hansi=name,passwd=pwd)

and the output page onsubmitinputpage1.html

*from* flask *import* Flask,render\_template,request

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

@app.route("/")

*def* basic():

*return* render\_template("index2\_POST.html")

*#ensure that the form has built in the html page with action as below route path anem*

@app.route("/fcuk")

*def* hello():

name=request.args.get("hansi")

pwd=request.args.get("n2")

*return* render\_template("onsubmitindex1.html",hansi=name,passwd=pwd)

the error got is



The error is bcz:

In

@app.route("/fcuk")

What it is happening is flask is supporting only get method by defalult so if u want the post method to be support on the particular method then change the syntax or the snippet as follow

@app.route("/fcuk",methods=[“POST”])

So the modified app3\_post.py be

*from* flask *import* Flask,render\_template,request

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

@app.route("/")

*def* basic():

*return* render\_template("index2\_POST.html")

*#ensure that the form has built in the html page with action as below route path anem*

*#befor modification where we are not specified to support the post method*

*#@app.route("/fcuk")*

*#to avodi error of method not allowed (ERROR 405) we use the following app.route syntax*

@app.route("/fcuk",methods=["POST"])

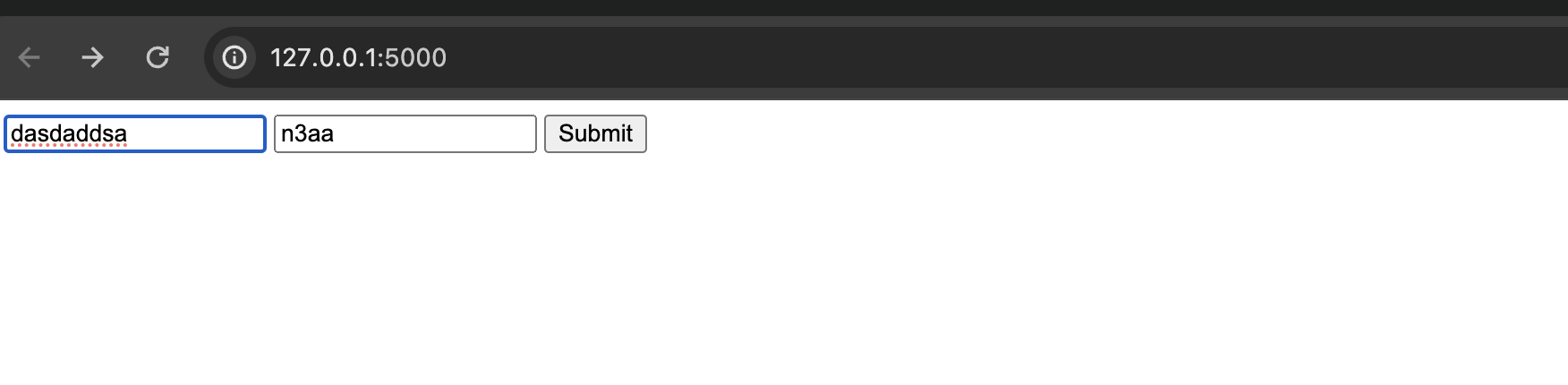
*def* hello():

name=request.args.get("hansi")

pwd=request.args.get("n2")

*return* render\_template("onsubmitindex1.html",hansi=name,passwd=pwd)

Page after running flasK:

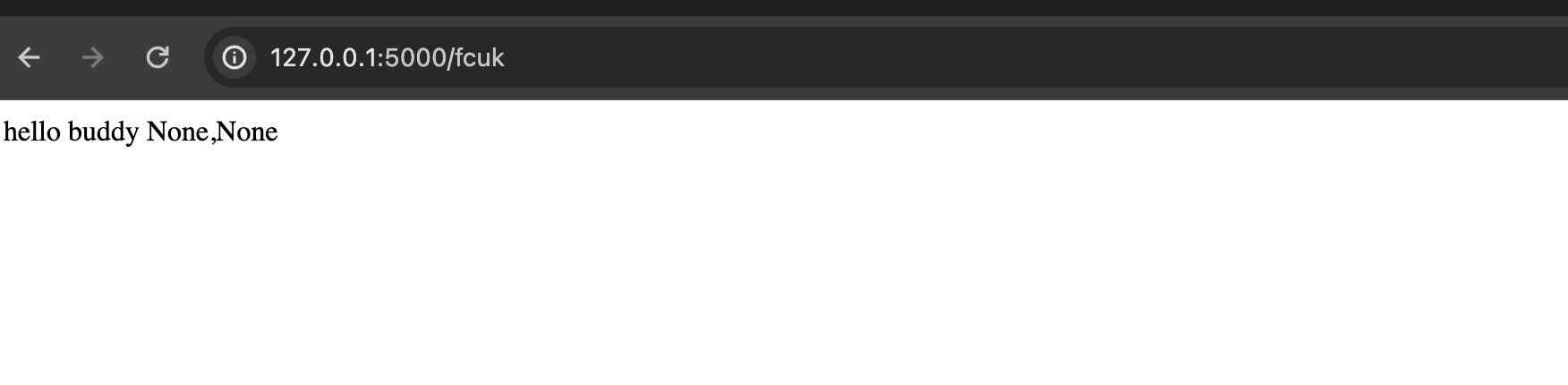


SEE HERE THE URL IS NOT LIKE 5000/fcuk?name=value ,pwd=value

Bcz POST method hides those credentials to maintain the privacy.but if u see u are not getting printed with values u entered let us discuss that below



OUTPUT AFTER SUBMIT:





To solve the error of not getting enterd values in for there come the diff b/w

request.args is for get request

request.form. is for post request

So after changing app3\_post.py with request.form.get instead of request.args.get

*from* flask *import* Flask,render\_template,request

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

@app.route("/")

*def* basic():

*return* render\_template("index2\_POST.html")

*#ensure that the form has built in the html page with action as below route path anem*

*#befor modification where we are not specified to support the post method*

*#@app.route("/fcuk")*

*#to avodi error of method not allowed (ERROR 405) we use the following app.route syntax*

@app.route("/fcuk",methods=["POST"])

*def* hello():

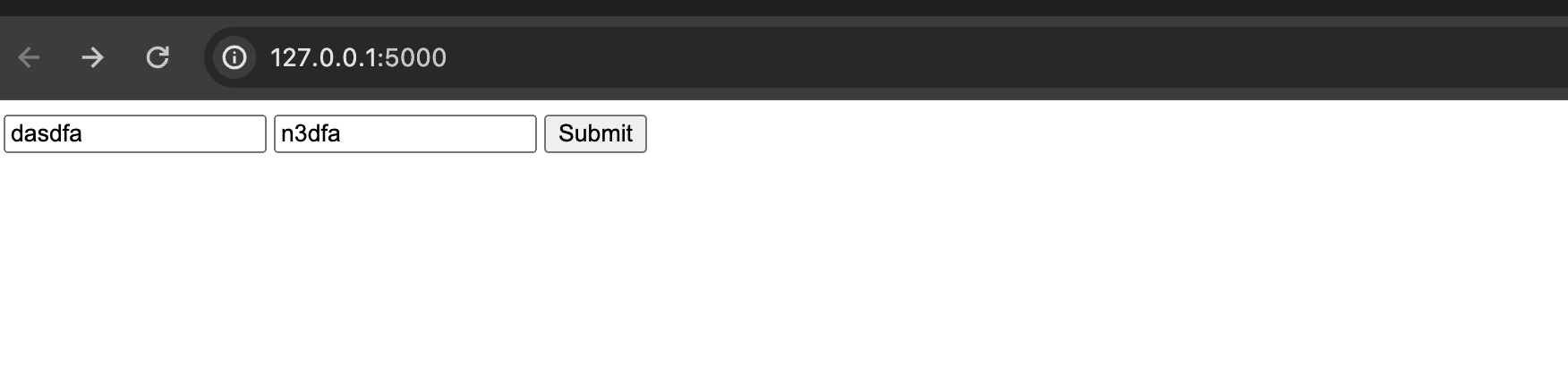
name=request.args.get("hansi")

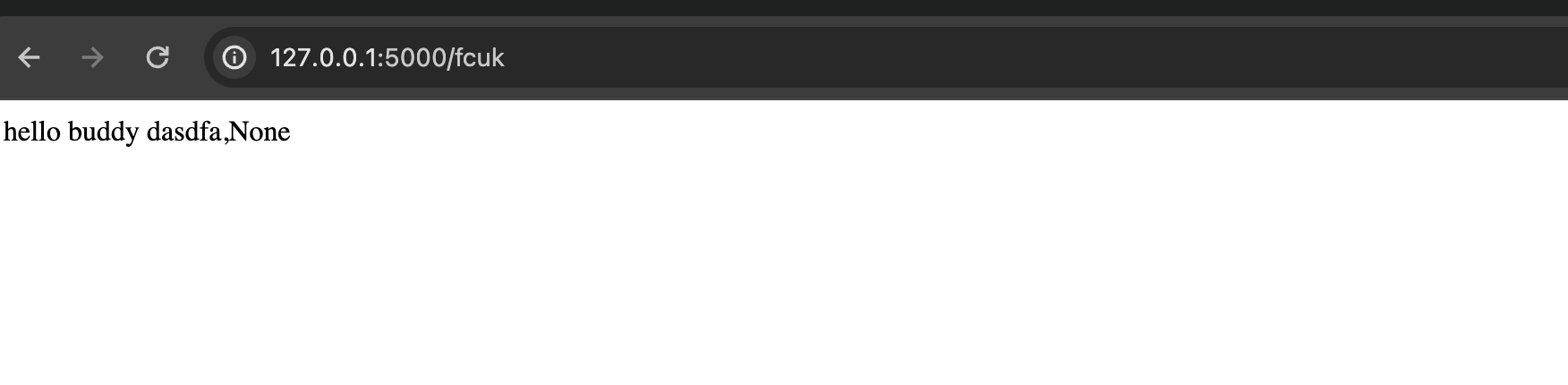
pwd=request.args.get("n2")

*return* render\_template("onsubmitindex1.html",hansi=name,passwd=pwd)

output:







Finally we got url without credentials and the form with one value as entered and other with None bcz for that we didn’t change to request.form

The code is as follows:

@app.route("/fcuk",methods=["POST"])

*def* hello():

name=request.form.get("hansi")

pwd=request.args.get("n2")

*return* render\_template("onsubmitindex1.html",hansi=name,passwd=pwd)



Why we use if \_ \_ name\_ \_ =\_ \_ main\_ \_

This is to ensure the whenever the program with this condition can execute the statements in the condition only if the current program is running directly by flask server not like

The program with this condition is imported into other program and there it is called and if that programmed is runned by flask server the statements In this method will not be executed.

Chatgpt explanation:

In Python, a script can be either the main program that is executed or it can be imported as a module into another script. The **if \_\_name\_\_ == '\_\_main\_\_':** block is a way to determine whether the current script is being run directly as the main program or if it's being imported as a module.Let's illustrate this with an example. Consider you have two Python scripts: **main\_script.py** and **module\_script.py**.

**module\_script.py:**

# module\_script.py

def hello():

print("Hello from module\_script!")

# This block won't be executed if the script is imported as a module

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

print("This won't be printed if module\_script is imported.")

**main\_script.py:**

# main\_script.py

import module\_script

print("This is the main program.")

# Call the hello function from module\_script

Now, if you run **main\_script.py**, you'll see the following output:

This is the main program.

Hello from module\_script!

The block of code under **if \_\_name\_\_ == '\_\_main\_\_':** in **module\_script.py** is not executed when **module\_script.py** is imported into **main\_script.py**. It only runs when **module\_script.py** is executed directly.

This distinction is useful when you want certain code (like script initialization, configuration, or standalone tests) to run only when the script is executed directly and not when it's imported as a module in another script.

# Usage of “app.route('/hello/<name>')”

What this does is anything u enter in the url after /hello/ the value is taken and stored in the variable name as we used <name> that we can use anywhere in the the followed function of the route

Ex:

App1.py

|  |
| --- |
| *from* flask *import* Flask, render\_template,request  app = Flask(\_\_name\_\_)  @app.route('/')  *def* x():  *return* render\_template("index1.html")    @app.route('/hello/<name>')  *def* hello\_name(name):  *return* render\_template("index.html",abc=name)  *if* \_\_name\_\_ == "\_\_main\_\_":  app.run(debug=True) |

Index1.html :

Has a sinle statement

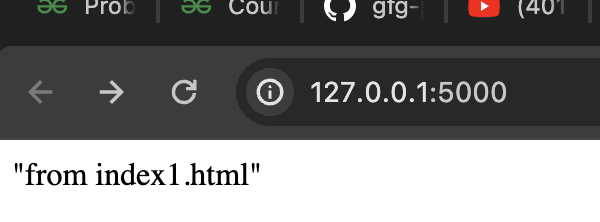
Index.html :

Has.

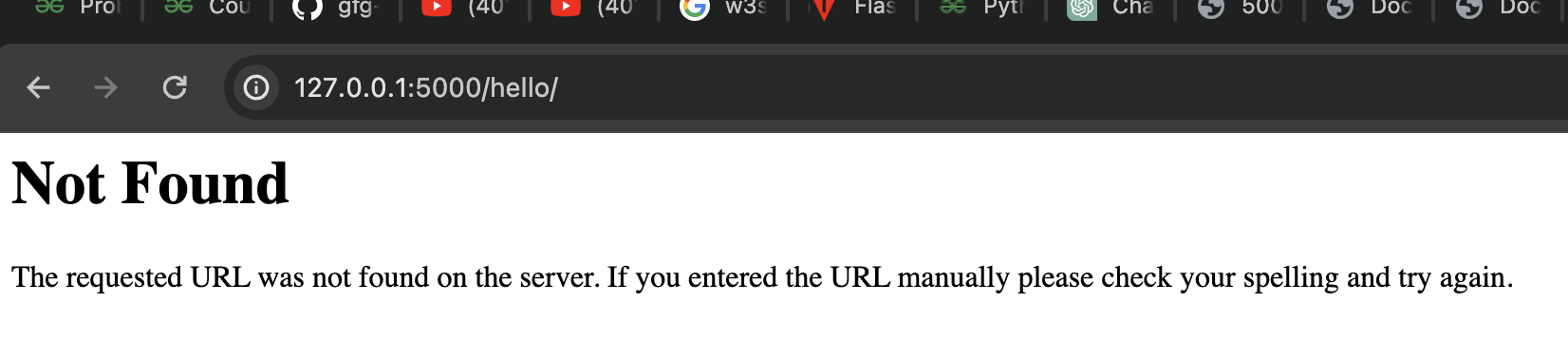
hello,{{abc}}

now for the following url we will get following response:

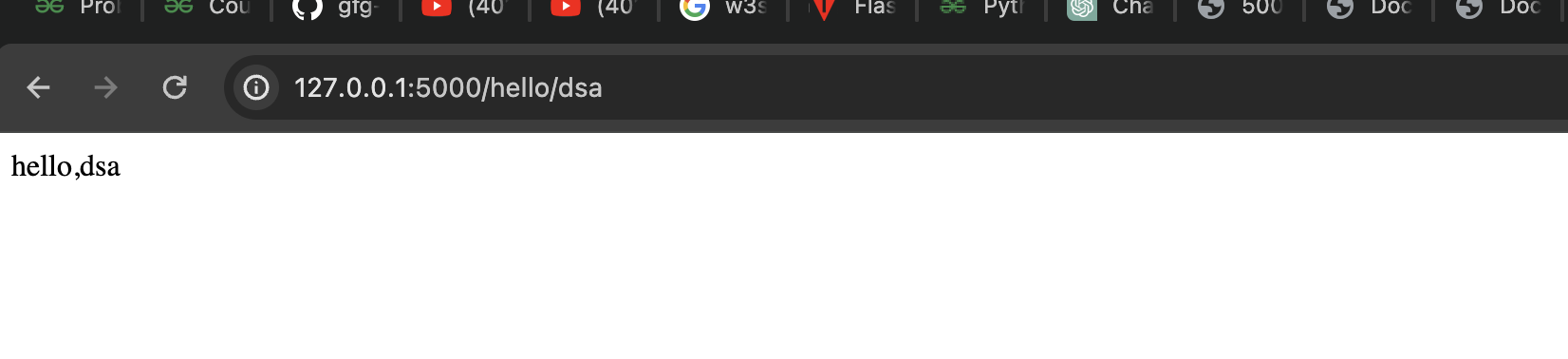
1)<http://127.0.0.1:5000/>



2) <http://127.0.0.1:5000/hello/> 🡺not like if app.route(/hello) and getting keyvalue pairswe will get hello,None as in index.html it is hello,{{abc}}



3) <http://127.0.0.1:5000/hello/dsa>



Summary:

What happening is that the value in the place of <name> is been taken and stored in the name and that is passed as an argument to the function hello\_name() where it is again passed to index file and in index file it is being used.

Other examples of above scenarios

S1)

|  |
| --- |
| *def* hello\_name(name):  *return*("hello %s" %name)  *#return render\_template("index.html",abc=name)* |

Prints hello and value entered in name

S2)

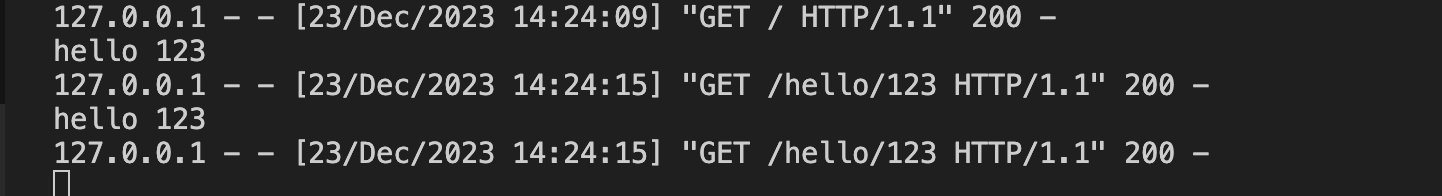
If u try

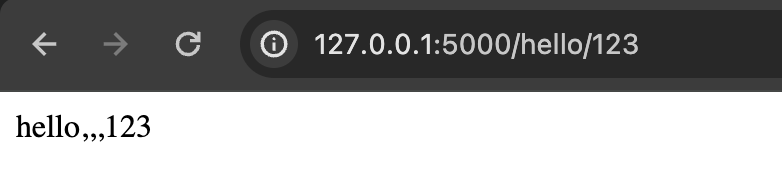
print("hello %s" %name)

*return render\_template("index.html",abc=name)*

When a request is made to the "/hello/<name>" route, the message "hello %s" % name will be printed to the console where your Flask server is running.

To see the output, you need to look at the terminal or command prompt where you started your Flask application. You should see the print statement output there.





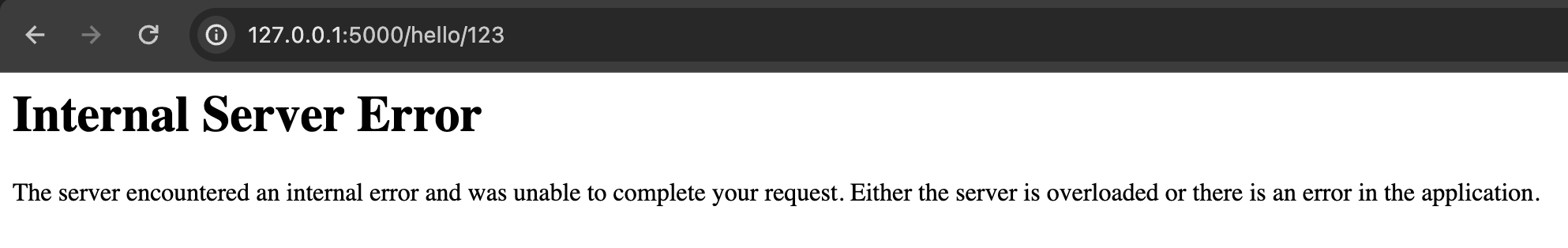
S3)

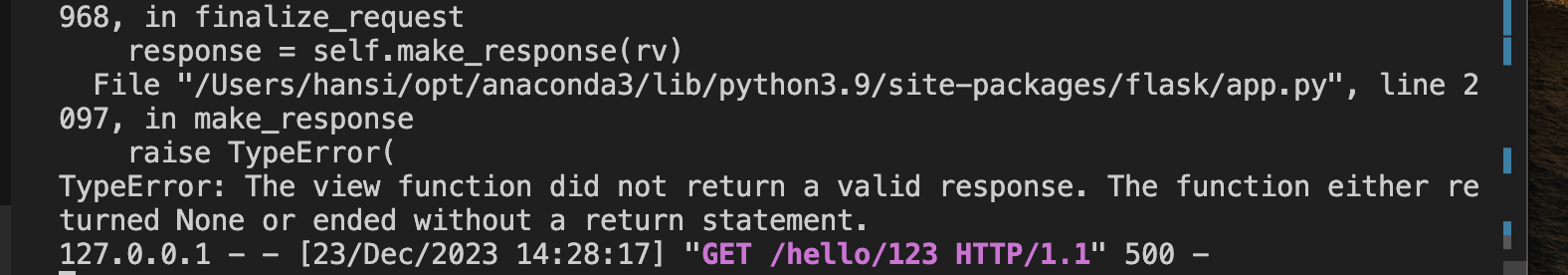
But if u try to execute

print("hello %s" %name)

alone it shows error bcz the function isn’t returning anything:

like bleow





S4)

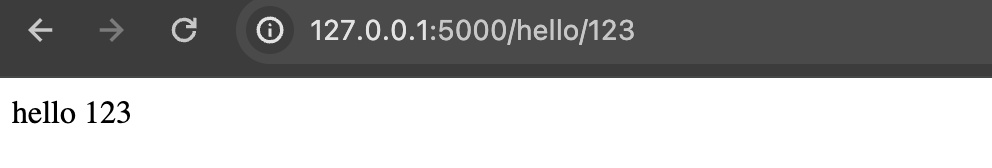
If u try

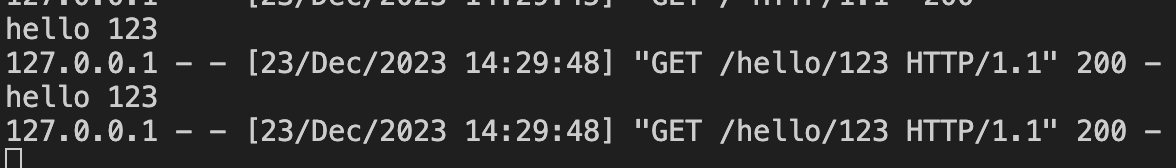
*def* hello\_name(name):

print("hello %s" %name)

*#return render\_template("index.html",abc=name)*

*return*" hello %s" %name



In terminal u will get the following output:

We can even print the other datatypes toooo:

@app.route('/blog/<postID>')

**def** show\_blog(postID):

**return** 'Blog Number %d' **%** postID

@app.route('/rev/<revNo>')

**def** revision(revNo):

**return** 'Revision Number %f' **%** revNo

DATATYPED SUPPORTED IN DYNAMIC URL:

|  |
| --- |
| @app.route('/post/<int:id>')  **def** show\_post(id):      # Shows the post with given id.  **return** f'This post has the id {id}' |

# JINJA 2

Certainly! Jinja2 is a powerful and expressive templating engine used in Flask for generating dynamic HTML content. Here are the key concepts and syntax elements you'll commonly encounter when working with Jinja2:

### 1. Variable Substitution:

In Jinja2, you can insert variables into your templates using the `{{ ... }}` syntax. For example:

```html

<p>Hello, {{ username }}!</p>

```

Here, the `username` variable will be dynamically replaced with its actual value when rendering the template.

### 2. Control Structures:

Jinja2 provides control structures similar to Python, such as `for` loops and `if` statements.

- \*\*For Loop:\*\*

```html

<ul>

{% for item in items %}

<li>{{ item }}</li>

{% endfor %}

</ul>

```

- \*\*If Statement:\*\*

```html

{% if condition %}

<p>Condition is true</p>

{% else %}

<p>Condition is false</p>

{% endif %}

```

### 3. Filters:

Filters in Jinja2 modify the data that is being output. They are applied to variables using the `|` symbol.

```html

<p>{{ text\_variable|capitalize }}</p>

```

In this example, the `capitalize` filter is used to capitalize the first letter of the text.

### 4. Macros:

Macros in Jinja2 are similar to functions. They allow you to define reusable chunks of template code.

```html

{% macro render\_item(item) %}

<li>{{ item }}</li>

{% endmacro %}

<ul>

{% for item in items %}

{{ render\_item(item) }}

{% endfor %}

</ul>

```

### 5. Template Inheritance:

Jinja2 supports template inheritance, allowing you to create a base template and extend it in other templates.

\*\*Base Template (`base.html`):\*\*

```html

<!DOCTYPE html>

<html>

<head>

<title>{% block title %}My Website{% endblock %}</title>

</head>

<body>

<div id="content">

{% block content %}{% endblock %}

</div>

</body>

</html>

```

\*\*Child Template:\*\*

```html

{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block content %}

<h1>Welcome to my website!</h1>

{% endblock %}

```

In this example, the child template (`home.html`) extends the base template (`base.html`) and overrides specific blocks.

### 6. Comments:

Comments in Jinja2 are enclosed within `{# ... #}`.

```html

{# This is a comment #}

```

### 7. Template Environment Configuration (Optional):

You can configure the Jinja2 environment in Flask, but this is typically done in the background by Flask itself. However, it's good to be aware that you can customize the Jinja2 environment if needed.

```python

app.jinja\_env.globals.update(my\_variable='some\_value')

```

These are some fundamental concepts of Jinja2. As you work with Flask, you'll become more familiar with its features and syntax. The official [Jinja2 documentation](https://jinja.palletsprojects.com/) is an excellent resource for in-depth learning and reference.