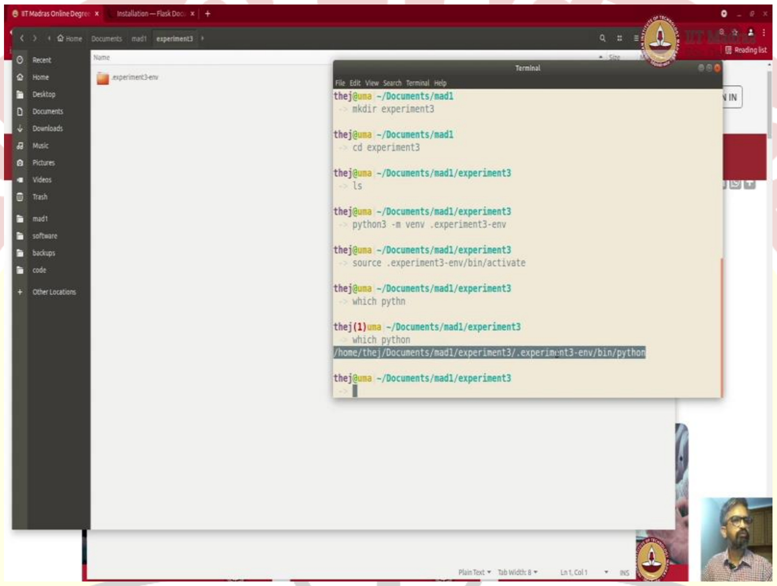
INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

IIT Madras

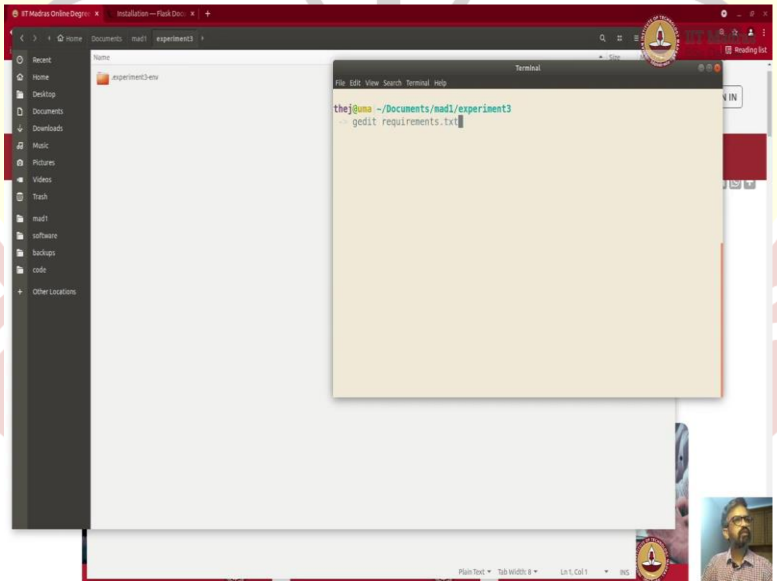
ONLINE DEGREE

Modern Application Development – I
Professor Thejesh G N
Software Consultant
IITM BSc Degree
Indian Institute of Technology, Madras
Introduction to Flask

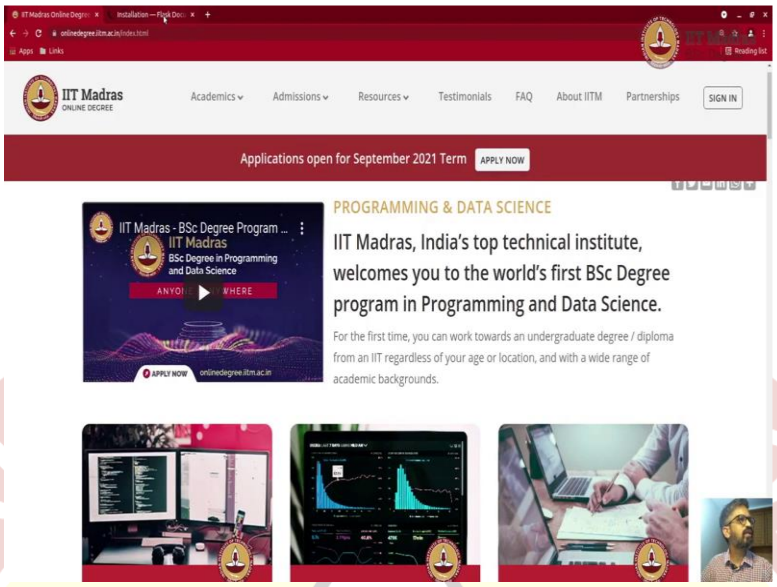
(Refer Slide Time: 00:19)



```
thej@uma ~/Documents/mad1
 -> mkdir experiment3

thej@uma ~/Documents/mad1
 -> cd experiment3

thej@uma ~/Documents/mad1/experiment3
 -> ls

thej@uma ~/Documents/mad1/experiment3
 -> python3 -m venv .experiment3-env

thej@uma ~/Documents/mad1/experiment3
 -> source .experiment3-env/bin/activate

thej@uma ~/Documents/mad1/experiment3
 -> which pythn

thej(1)uma ~/Documents/mad1/experiment3
 -> which python
/home/thej/Documents/mad1/experiment3/.experiment3-env/bin/python

thej@uma ~/Documents/mad1/experiment3
 ->
```



```
thej@uma ~/Documents/mad1/experiment3
 -> gedit requirements.txt
```


IIT Madras Online Degree
Installation — Flask Docs
onlinedegree.iitm.ac.in/index.html
Apps
Links
Reading list
IIT Madras
ONLINE DEGREE
Academics
Admissions
Resources
Testimonials
FAQ
About IITM
Partnerships
SIGN IN
Applications open for September 2021 Term
APPLY NOW
IIT Madras - BSc Degree Program ...
IIT Madras
BSc Degree in Programming and Data Science
APPLY NOW
onlinedegree.iitm.ac.in
PROGRAMMING & DATA SCIENCE

IIT Madras, India's top technical institute, welcomes you to the world's first BSc Degree program in Programming and Data Science.

For the first time, you can work towards an undergraduate degree / diploma from an IIT regardless of your age or location, and with a wide range of academic backgrounds.

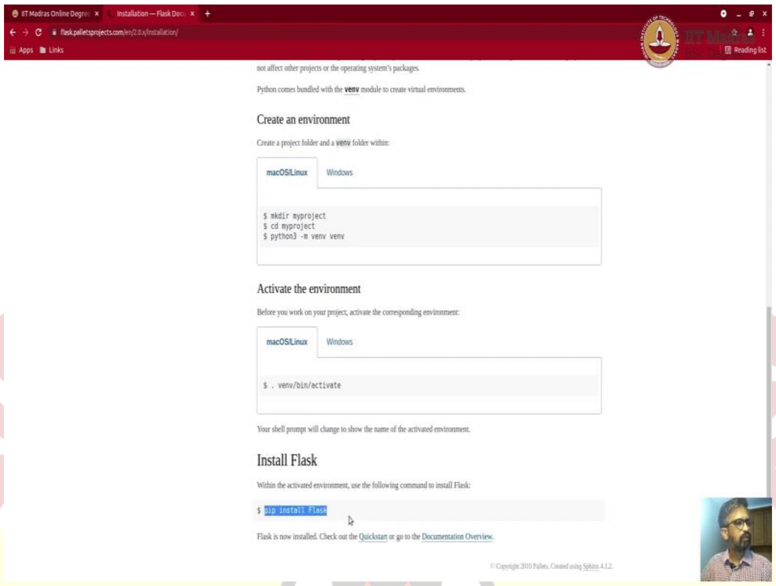
Welcome to the Modern Application Development Screencast. In this short screencast we will learn how to use flask to build a simple web application and run the same web application using built-in flash server. Before we start, open the following applications on your Ubuntu desktop so that you can work along with me.

Browser like your chrome built-in editor called gedit in Ubuntu you could use any editor whichever you are comfortable with; a terminal I am using a built-in terminal and then file browser. Now I am using a built-in file browser. So let us start by creating a folder called experiment3.

So, we are already in our work folder I am going to create another folder called experiment3 where we can work. And then I am going to cd experiment3, yes. So now to begin with we are going to create a virtual environment like we did previously, so that all our installations, all our dependencies are contained within this, yes. So that is done by running this python3 -m venv .experiment-env; this is where your virtual environment will be created including the dot.

Turn it okay it is created, now we can source from there or activate the virtual environment, bin activate; it is done you can check sorry as you can see it is using the local Python from our virtual environment so it is activated, it is going to clear I am just going to create a requirements.txt file and have flask as part of it just do gedit requirements.txt.

(Refer Slide Time: 03:18)



not affect other projects or the operating system's packages.

Python comes bundled with the **venv** module to create virtual environments.

Create an environment

Create a project folder and a `venv` folder within:

```
$ mkdir myproject
$ cd myproject
$ python3 -m venv venv
```

Activate the environment

Before you work on your project, activate the corresponding environment:

```
$ . venv/bin/activate
```

Your shell prompt will change to show the name of the activated environment.

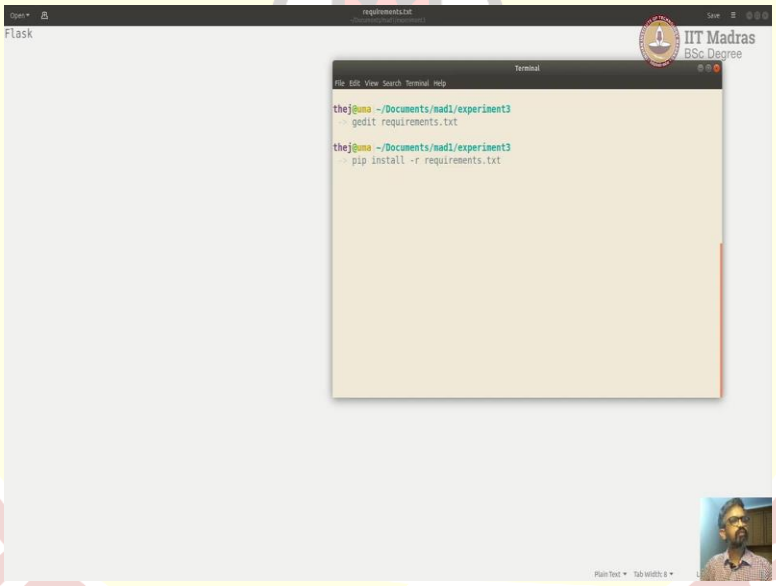
Install Flask

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
```

Flask is now installed. Check out the Quickstart or go to the Documentation Overview.

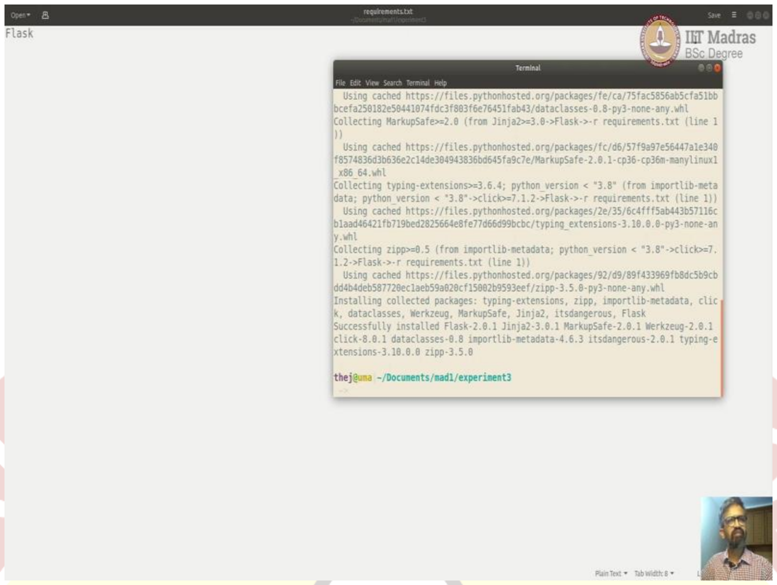
© Copyright 2010 Pallets. Created using Sphinx 4.1.2.



requirements.txt

```
Flask
```

```
thej@uma ~/Documents/mad1/experiment3
-> gedit requirements.txt

thej@uma ~/Documents/mad1/experiment3
-> pip install -r requirements.txt
```



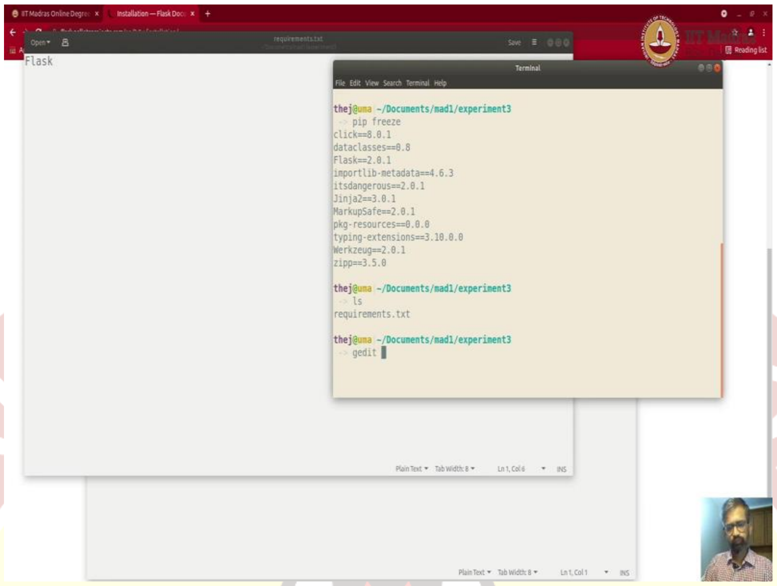
```
Flask
```

```
  Using cached https://files.pythonhosted.org/packages/fe/ca/75fac5856ab5cfa51bb
bcefa250182e50441074fdc3f803f6e76451fab43/dataclasses-0.8-py3-none-any.whl
Collecting MarkupSafe>=2.0 (from Jinja2>=3.0->Flask->-r requirements.txt (line 1
))
  Using cached https://files.pythonhosted.org/packages/fc/d6/57f9a97e56447a1e340
f8574836d3b636e2c14de304943836bd645fa9c7e/MarkupSafe-2.0.1-cp36-cp36m-manylinux1
_x86_64.whl
Collecting typing-extensions>=3.6.4; python_version < "3.8" (from importlib-meta
data; python_version < "3.8"->click>=7.1.2->Flask->-r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/2e/35/6c4fff5ab443b57116c
b1aad46421fb719bed2825664e8fe77d66d99bcbc/typing_extensions-3.10.0.0-py3-none-an
y.whl
Collecting zipp>=0.5 (from importlib-metadata; python_version < "3.8"->click>=7.
1.2->Flask->-r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/92/d9/89f433969fb8dc5b9cb
dd4b4deb587720ec1aeb59a020cf15002b9593eef/zipp-3.5.0-py3-none-any.whl
Installing collected packages: typing-extensions, zipp, importlib-metadata, clic
k, dataclasses, Werkzeug, MarkupSafe, Jinja2, itsdangerous, Flask
Successfully installed Flask-2.0.1 Jinja2-3.0.1 MarkupSafe-2.0.1 Werkzeug-2.0.1
click-8.0.1 dataclasses-0.8 importlib-metadata-4.6.3 itsdangerous-2.0.1 typing-e
xtensions-3.10.0.0 zipp-3.5.0

thej@uma ~/Documents/mad1/experiment3
```

Now, to install flask you can either do directly like this or I can, you can just add the name of the library flask into your requirements.txt this is one way. And you can include the version information by adding whether you are using 1.0.0 or whichever version that you want to use usually the website will tell you what version it is.

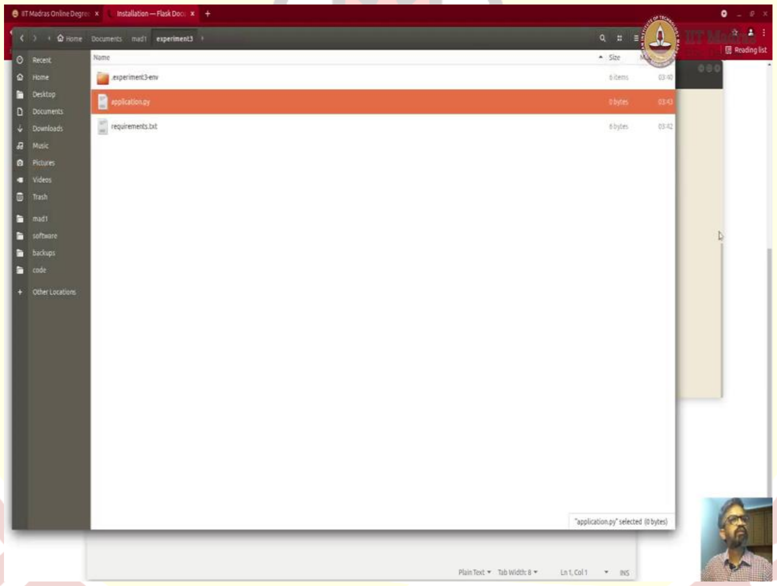
If you do not give any version, it will install the latest, so I am just going to leave it as the latest. So, we have requirements.txt. Now I am going to install everything from requirements.txt which is just flask, okay minus pip install - r requirements.txt. So it is going to read from this file, it is read and it is installing flask, it is installed flask and any other dependency flask depends on. Could be many other.

(Refer Slide Time: 04:23)



Flask

```
thej@uma ~/Documents/mad1/experiment3
 -> pip freeze
click==8.0.1
dataclasses==0.8
Flask==2.0.1
importlib-metadata==4.6.3
itsdangerous==2.0.1
Jinja2==3.0.1
MarkupSafe==2.0.1
pkg-resources==0.0.0
typing-extensions==3.10.0.0
Werkzeug==2.0.1
zipp==3.5.0

thej@uma ~/Documents/mad1/experiment3
 -> ls
requirements.txt

thej@uma ~/Documents/mad1/experiment3
 -> gedit
```



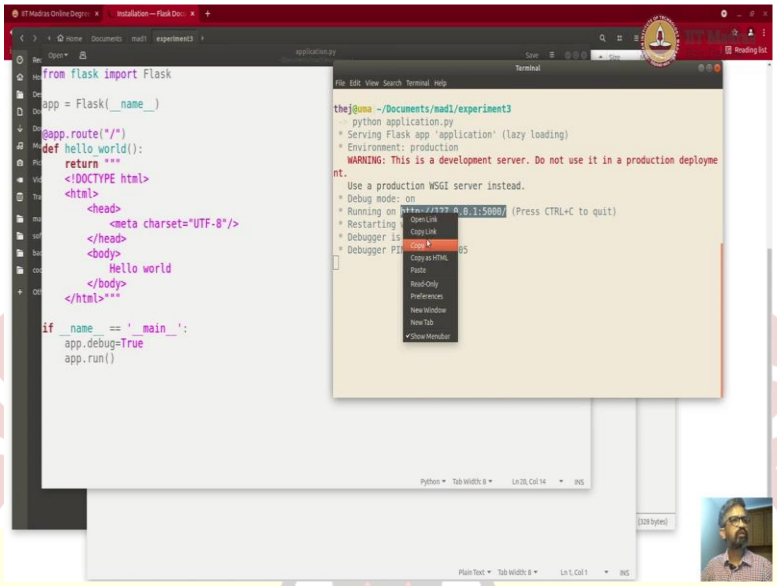
```
experiment3-env      6 items   03:40
application.py       0 bytes   03:43
requirements.txt     6 bytes   03:42
```

"application.py" selected (0 bytes)

So, just to check what all it got installed you can do pip freeze and it should show you all the installed things it has installed flask obviously and flask depends on all these other packages including jinja2 markup space and all other things and install all of them. So now we have installed flask, we have set up a virtual environment, okay.

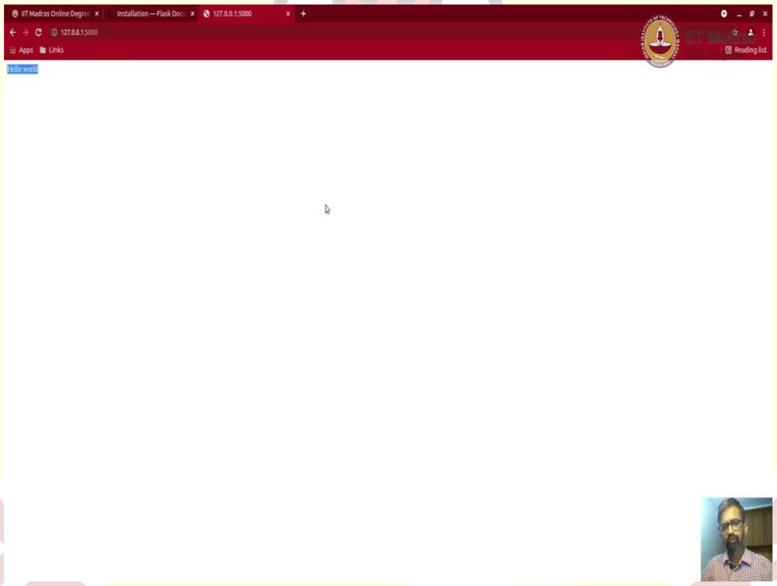
Now, we will create an application.py file this is where we are going to do all our work. I am just going to create it, just directly in gedit, so here it is application I can just save it just to make it, make sure it got saved here it is, okay. Now let us open the editor again here and I am going to close this requirements.txt, this is no more useful.

(Refer Slide Time: 05:30)



```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return """
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8"/>
        </head>
        <body>
            Hello world
        </body>
    </html>"""

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

```
thej@uma ~/Documents/mad1/experiment3
 -> python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting
 * Debugger is
 * Debugger PI                05
```



Hello world

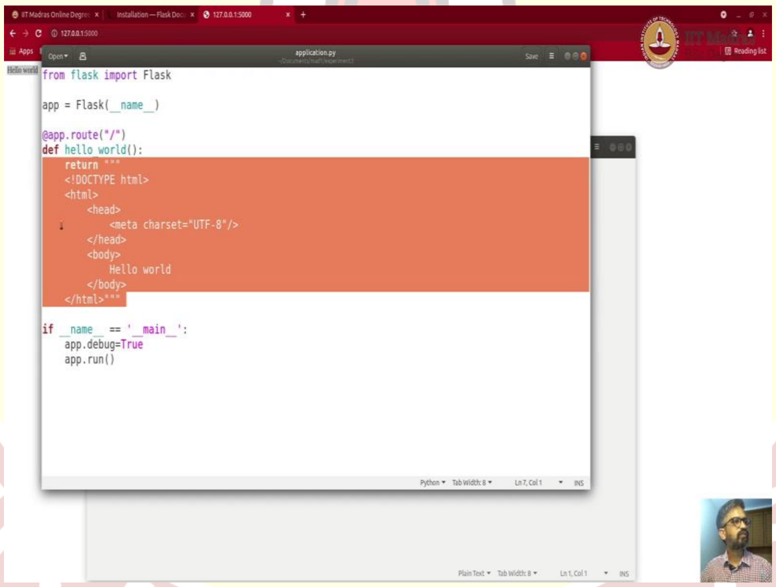
Now, let us create a simple flask app which has one page and which prints a hello world html, so this is the most simplest flask we are using a flask from the flask class from the package flask or module flask and then we are creating the app; then we are creating one function which returns an html content as a string and it is routed to slash(/).

In the sense this is what tells the web server serve it at the base route you could change it I will show it to you. Then in our main which is when the script runs it runs sets in debugging mode so we get some error messages if it is; if there are errors, we probably do not use it in production or also done app.run(). It will run the flask app.

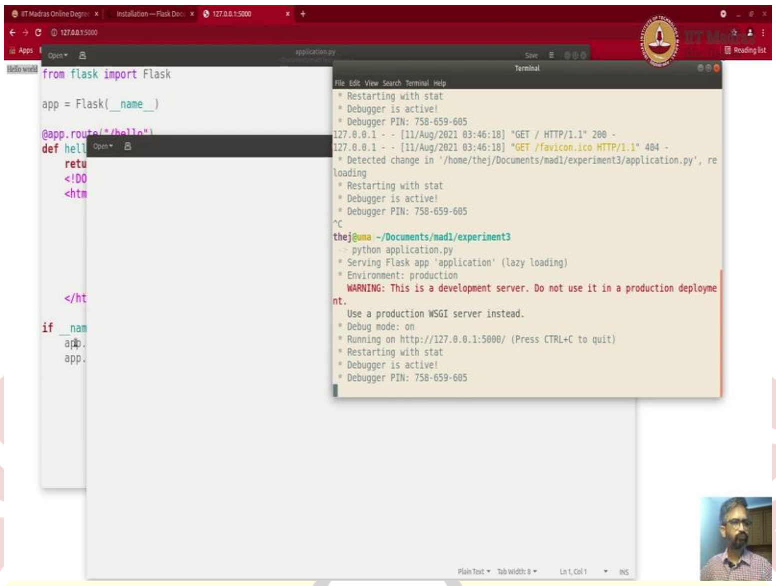
Now, we are going to run this using built-in http server which is actually not used in the production, it is not production ready but it is very good to run locally and do all our testing. So we can just run by Python. So as you can see the server has started it is also warning you whatever I just said that this is on development server cannot, should not be used in production.

So, we have there are many other ways to set up the production anyway the server has started and it is serving at 127.0.0.1 which is local host and at the port 5000. Remember by default flask runs at the port 5000 you can change it, you can look it up or you can change the port if you want. Now, I am just going to keep it with the default port. I am going to copy the link, go to the browser, open a new tab, paste on things. As you can see hello world.

(Refer Slide Time: 07:45)



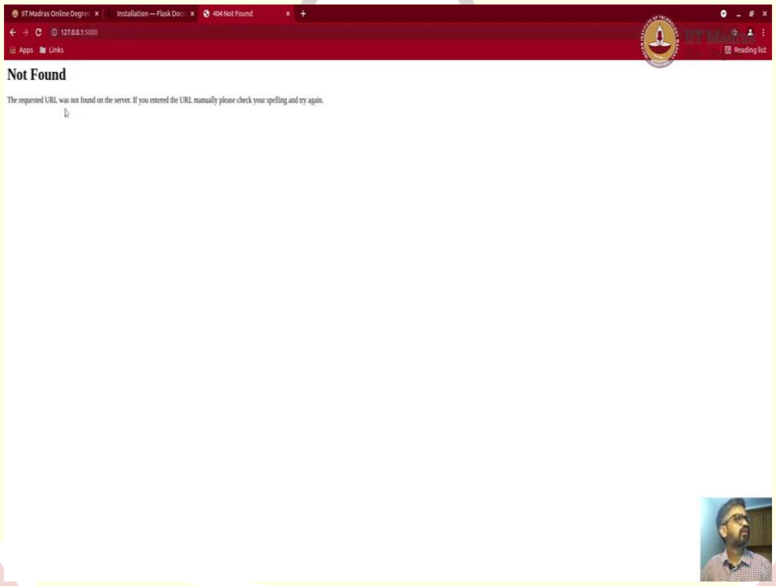
```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return """
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8"/>
        </head>
        <body>
            Hello world
        </body>
    </html>"""

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



Hello world

```
from flask import Flask

app = Flask(__name__)

@app.route("/hello")
def hell
    retu
    <!DO
    <htm




    </ht

if __nam
    app.
    app.
```

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
127.0.0.1 - - [11/Aug/2021 03:46:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 03:46:18] "GET /favicon.ico HTTP/1.1" 404 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
thej@uma ~/Documents/mad1/experiment3
 -> python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
```



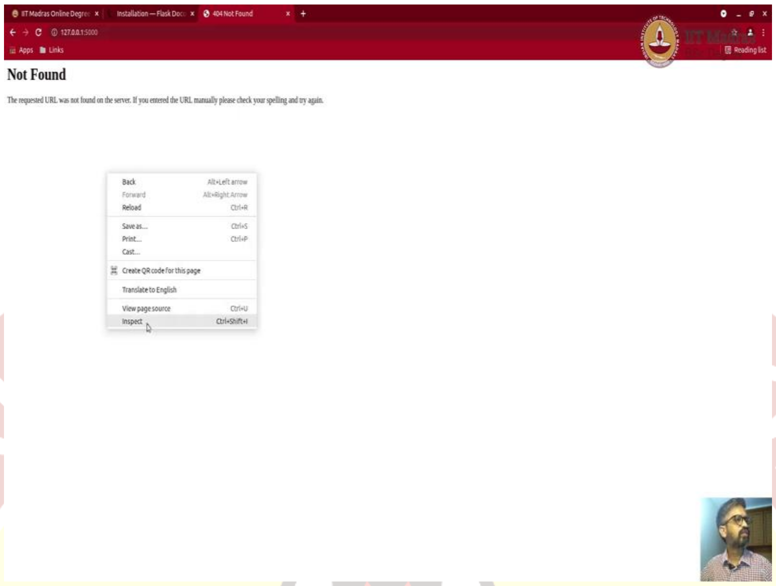
Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

Now, this is what is getting served because I have written slash here, now if I change this to let us say hello and save it that is the URL path now in flask most of the times you do not need to restart it, it automatically restarts and it will tell you whenever the file changes.

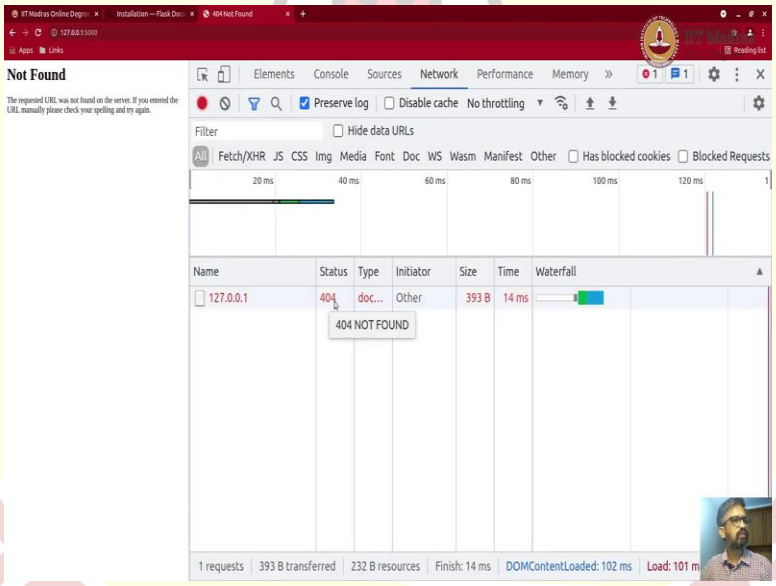
It is restarting but if you are not sure you can stop it by doing control C and start again. Most of the time it should restart but if it does not then you can stop and restart, it is also telling debug mode on because we said debug is equal to true. Anyway, now if I go back and say 5000 so you can see that nothing is getting served at a slash(/) it is saying 404 not found error.

(Refer Slide Time: 08:44)



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.



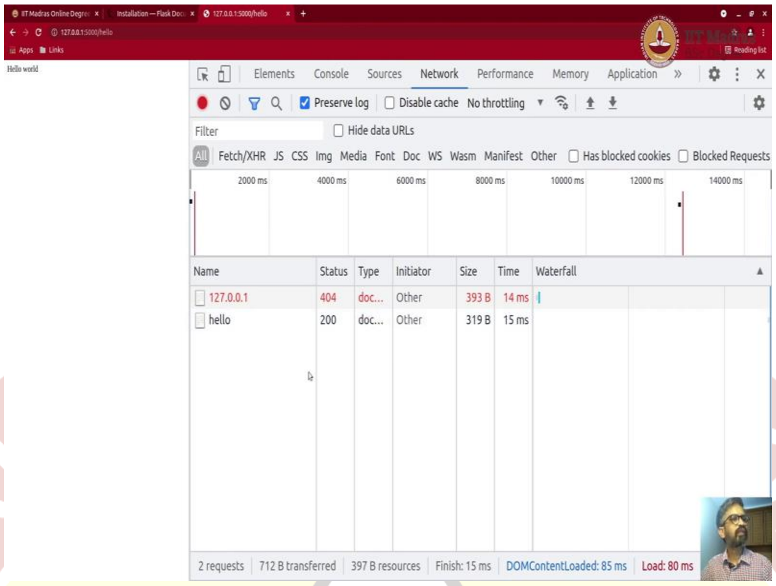
Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

[✓ Preserve log] [ ] Disable cache No throttling
[ ] Hide data URLs
[ ] Has blocked cookies [ ] Blocked Requests

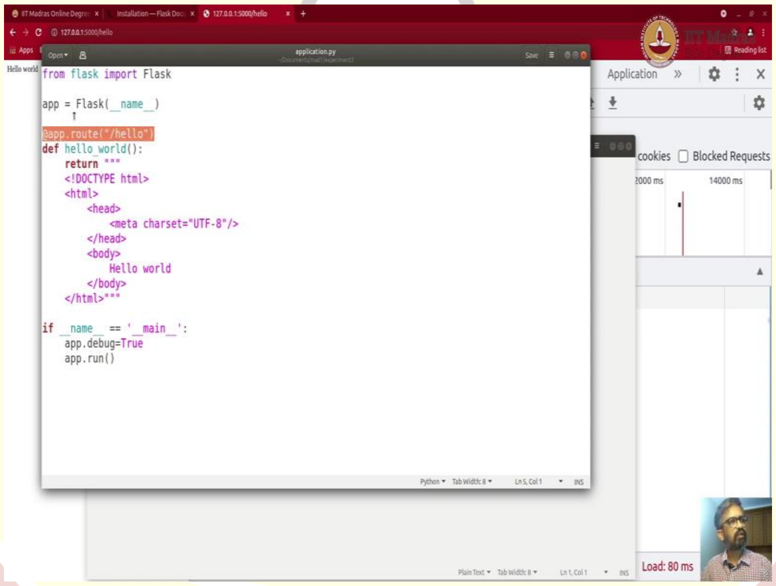
Name	Status	Type	Initiator	Size	Time	Waterfall
127.0.0.1	404	doc...	Other	393 B	14 ms	

1 requests | 393 B transferred | 232 B resources | Finish: 14 ms | DOMContentLoaded: 102 ms | Load: 101 m



Name	Status	Type	Initiator	Size	Time	Waterfall
127.0.0.1	404	doc...	Other	393 B	14 ms	
hello	200	doc...	Other	319 B	15 ms	

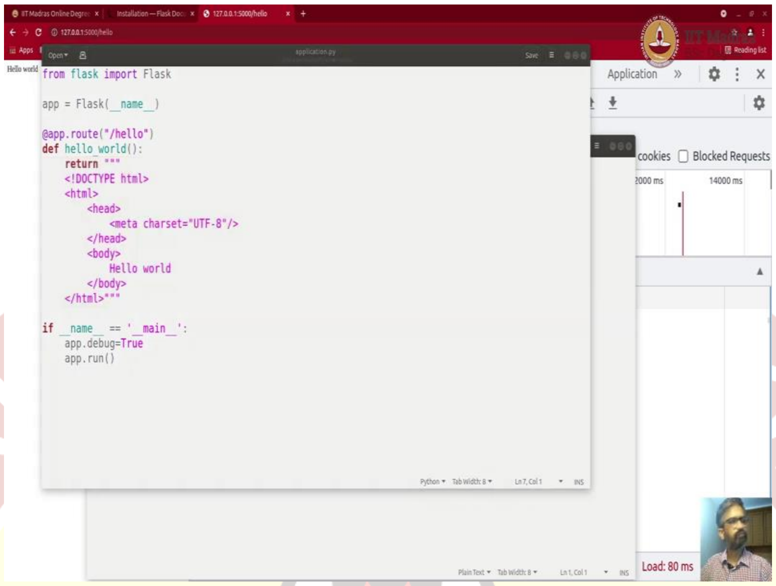
2 requests | 712 B transferred | 397 B resources | Finish: 15 ms | DOMContentLoaded: 85 ms | Load: 80 ms



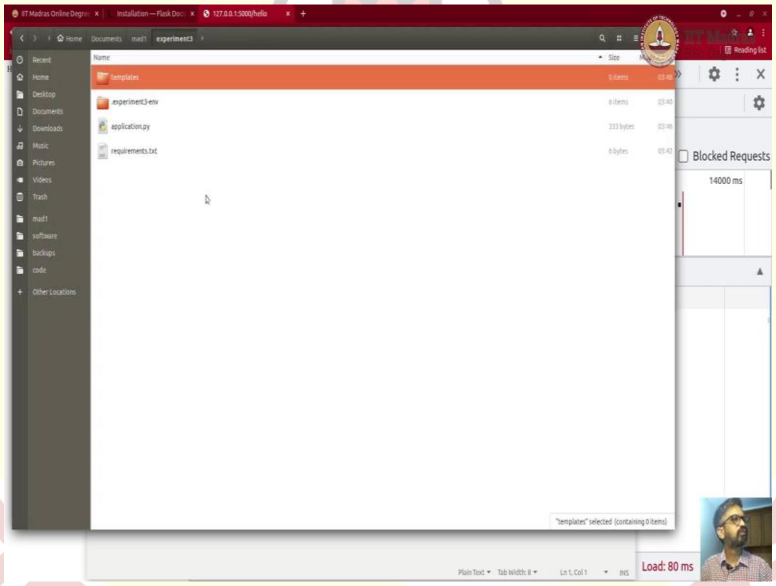
```
from flask import Flask

app = Flask(__name__)

@app.route("/hello")
def hello_world():
    return """
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8"/>
        </head>
        <body>
            Hello world
        </body>
    </html>"""

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

If you see a inspect network, let us reload it again just to make sure you can see, it is giving 404 not found. Now, the path which we had given hello you can check there now we can get 200 and it is printing. So this is the path that is getting set this is what a route uses is a http path, the path of the resource.

(Refer Slide Time: 09:27)

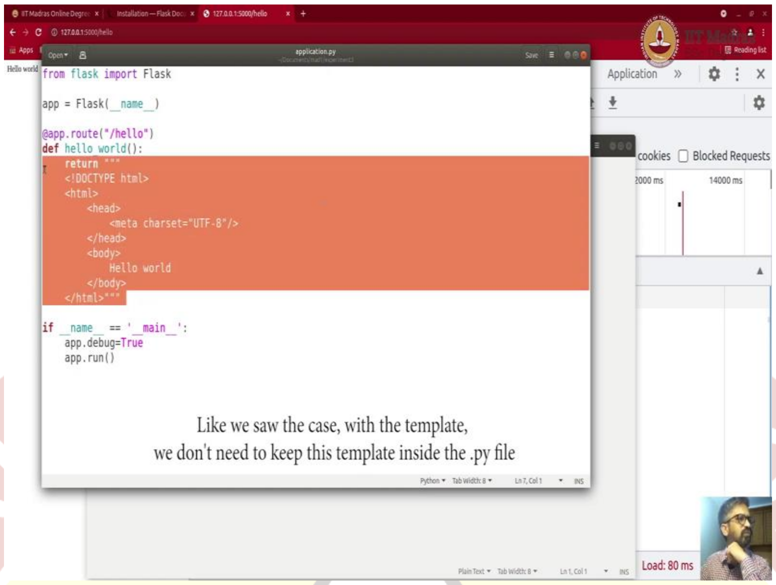


```
from flask import Flask

app = Flask(__name__)

@app.route("/hello")
def hello_world():
    return """
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8"/>
        </head>
        <body>
            Hello world
        </body>
    </html>"""

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



```
templates            0 items
experiment3-env      0 items
application.py       333 bytes
requirements.txt     0 bytes
```

"templates" selected (containing 0 items)



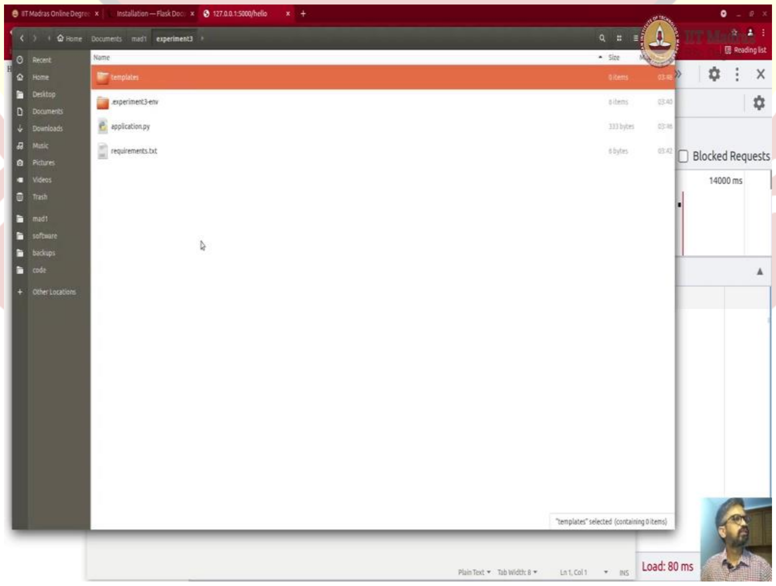
```
from flask import Flask

app = Flask(__name__)

@app.route("/hello")
def hello_world():
    return """
    <!DOCTYPE html>
    <html>
        <head>
            <meta charset="UTF-8"/>
        </head>
        <body>
            Hello world
        </body>
    </html>"""

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

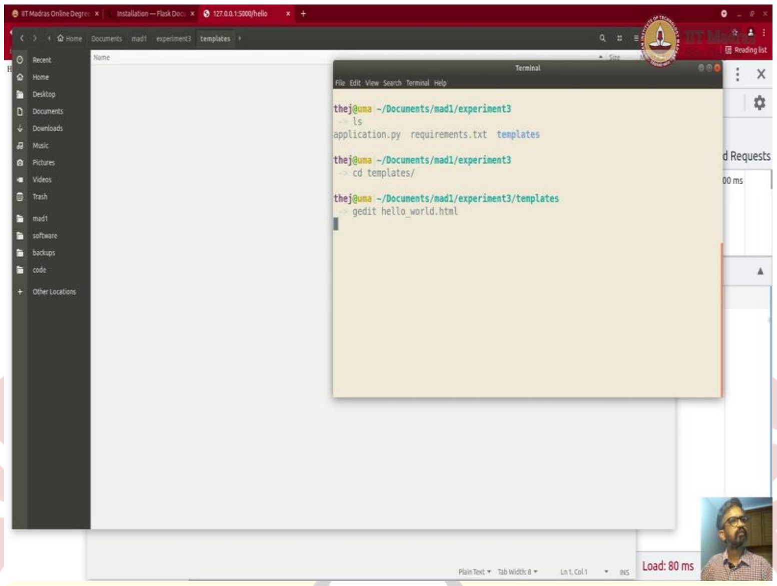
Like we saw the case, with the template,
we don't need to keep this template inside the .py file

Now, you can do one more thing to make it better we can actually externalize the html content like we saw, so we have a template, we do not need to keep this template inside the python file we can keep it outside.

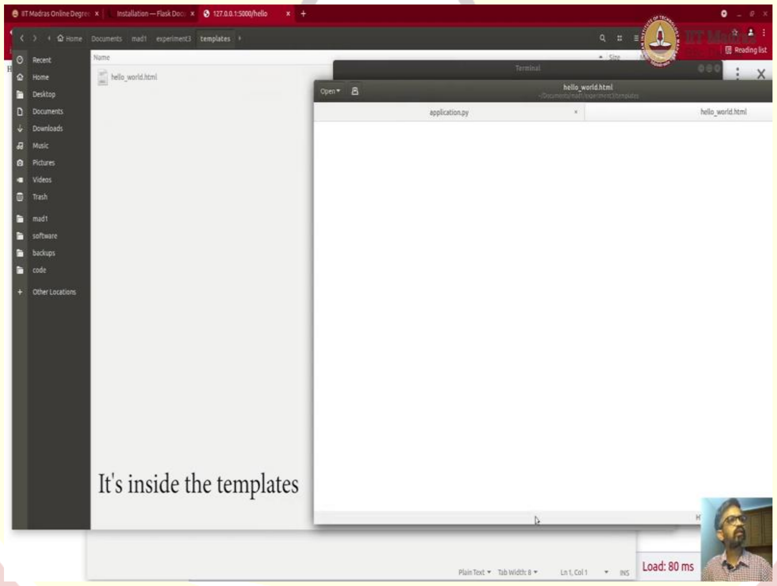
Now, flask indirectly or inside uses jinja2 and it all loads all the jinja templates from a folder called templates. So let us create a folder called templates in the same folder experiment3.

(Refer Slide Time: 10:02)





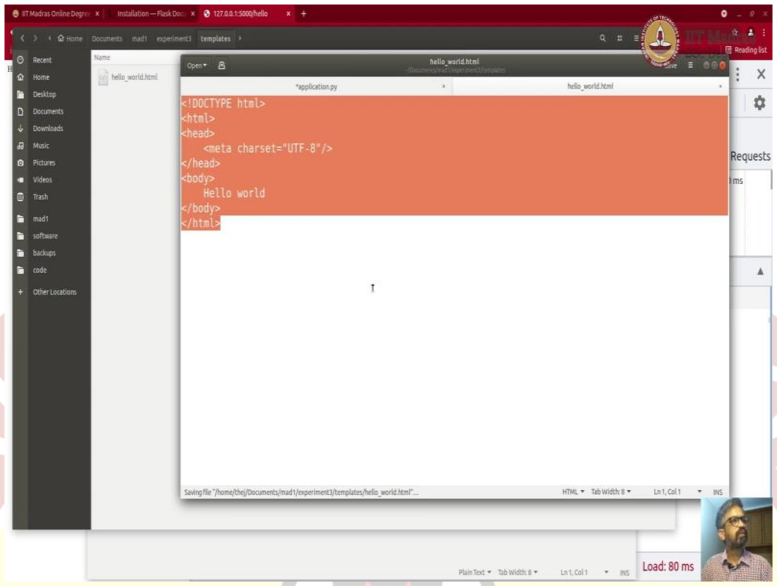
```
thej@uma ~/Documents/mad1/experiment3
-> ls
application.py  requirements.txt  templates

thej@uma ~/Documents/mad1/experiment3
-> cd templates/

thej@uma ~/Documents/mad1/experiment3/templates
-> gedit hello_world.html
```



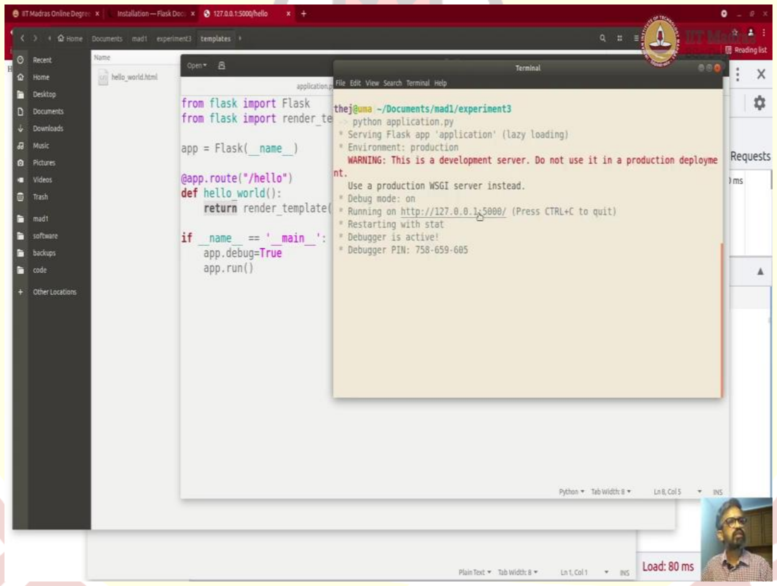
It's inside the templates

So, do they create a new folder templates inside the template since we are using a hello world program let us create a file called hello_world.html. I am going to stop the server cd template, I am just going to create a file. The file is here if I save it you can see it is inside the hello world.

(Refer Slide Time: 10:43)

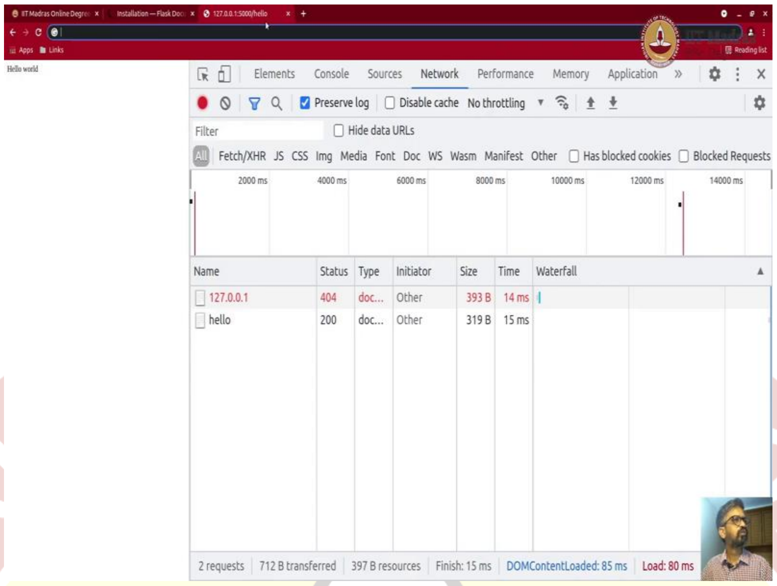


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
</head>
<body>
    Hello world
</body>
</html>
```



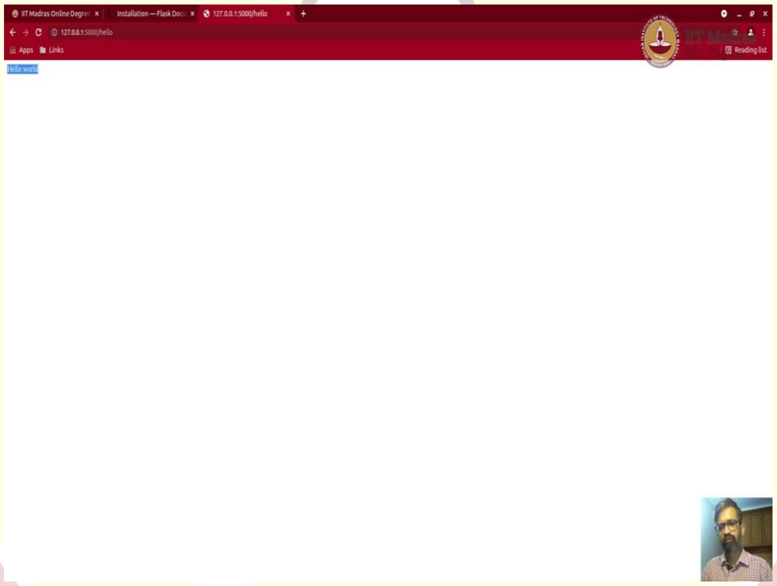
```
from flask import Flask
from flask import render_te

app = Flask(__name__)

@app.route("/hello")
def hello_world():
    return render_template(

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

```
thej@uma ~/Documents/mad1/experiment3
 > python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
```


Hello world

Name	Status	Type	Initiator	Size	Time	Waterfall
127.0.0.1	404	doc...	Other	393 B	14 ms	
hello	200	doc...	Other	319 B	15 ms	

```
2 requests | 712 B transferred | 397 B resources | Finish: 15 ms | DOMContentLoaded: 85 ms | Load: 80 ms
```

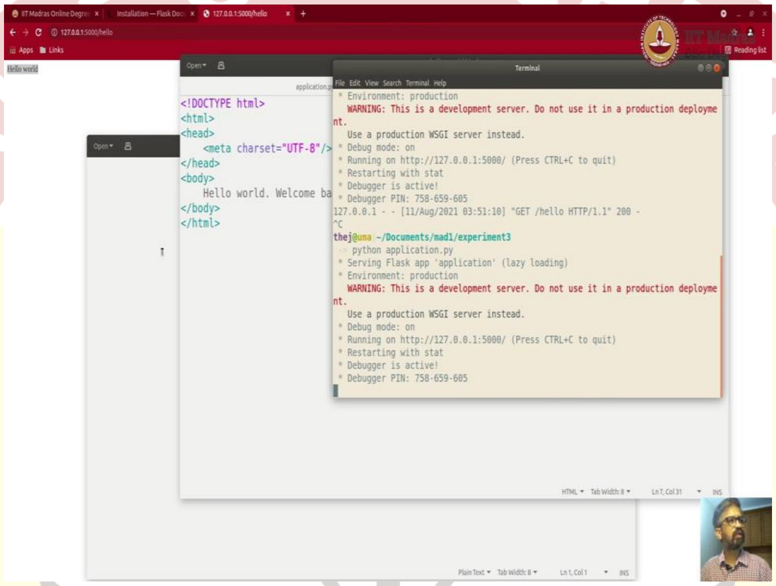

Hello world

Now, what I am going to do is I am going to go back to my application.py, take away this thing and paste it into this. So I am just removing html content from Python file and putting into its own template file. You can call the template file with any extension but I am going with .html here.

Now, we still have this hello_world() now we need to read the template and render it, that is easy flask has built in you do not need to use jinja directly, flask has wrapped around that jinja and it exposes a function called render template which actually takes the template file's name and renders it.

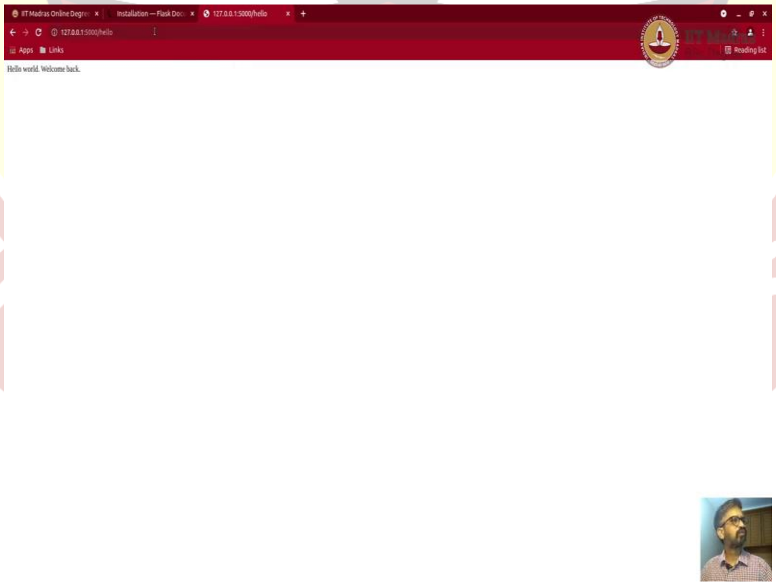
Like for example I can just do this and then I can just give hello_world.html, okay and save. So now what I am doing is instead of actually returning the content directly I am actually reading this and rendering it and return the rendering text, rendered text. Let us run the application, I am going to do python application.py, starts the server again same place. I am just going to close this tab and re-open it; we have hello world again.

(Refer Slide Time: 12:34)



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
</head>
<body>
    Hello world. Welcome ba
</body>
</html>
```

```
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
127.0.0.1 - - [11/Aug/2021 03:51:10] "GET /hello HTTP/1.1" 200 -
^C
thej@uma ~/Documents/madl/experiment3
 -> python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
```

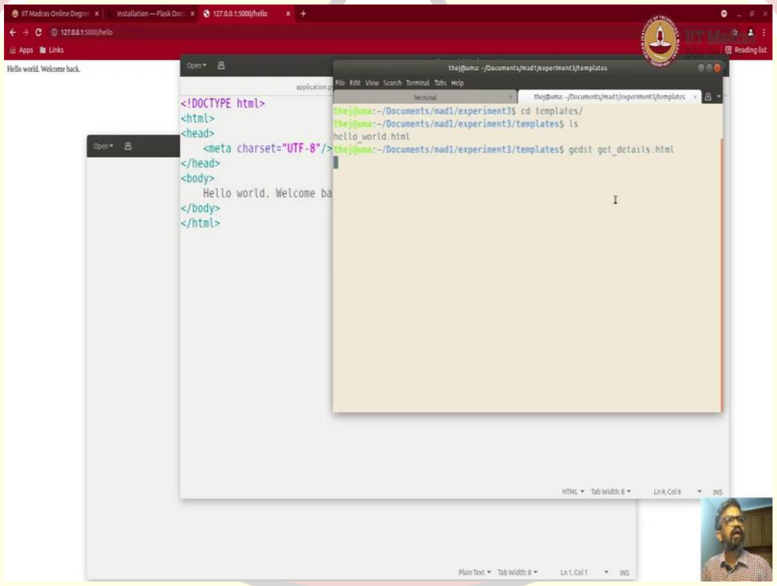


Hello world. Welcome back.

But this time it is coming from here. You can see hello world welcome back, when you save it the server should restart if it does not restart you can always start again, stop and start again just to make sure and then refresh, again hello world welcome back. Now this is useful what we are just displaying the data on the server.

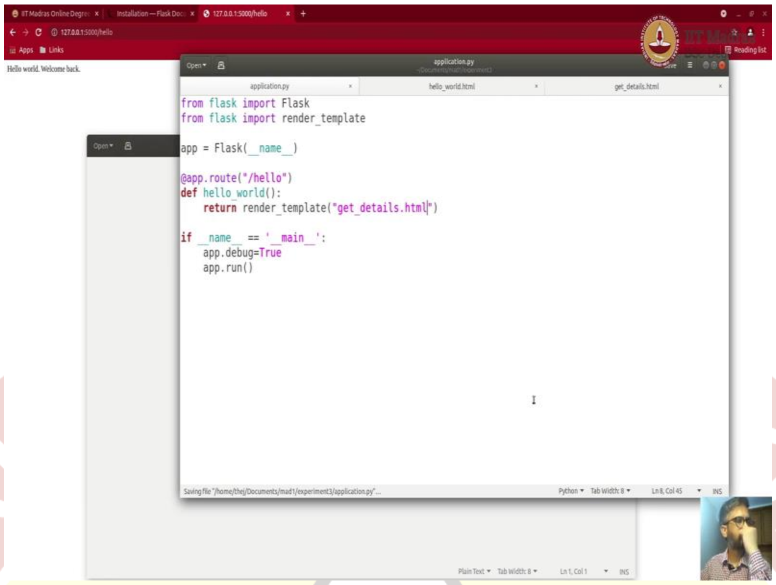
Now, what we want is if a person or a user should be able to submit something, let us say his name and then we should be able to use the name he submitted and welcome him with that name. So which needs, which means we need a form where the user can submit his name. And then use that name to show him the welcome message. Let us try and do that. We will use the same URL path hello, we have to create in a form to get the details of the user.

(Refer Slide Time: 13:55)


Hello world. Welcome back.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8"/>
</head>
<body>
    Hello world. Welcome ba
</body>
</html>
```

```
thej@uma:~/Documents/mad1/experiment3$ cd templates/
thej@uma:~/Documents/mad1/experiment3/templates$ ls
hello_world.html
thej@uma:~/Documents/mad1/experiment3/templates$ gedit get_details.html
```
सिद्धिर्भवति कर्मजा



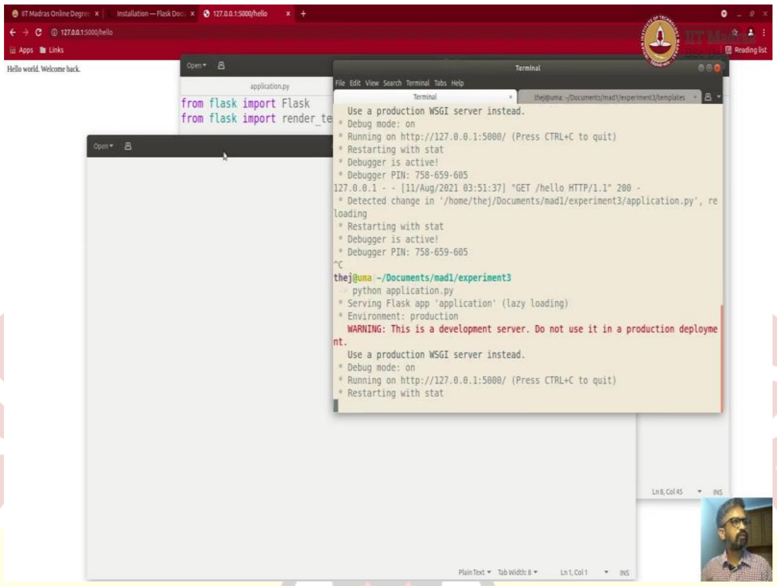
Hello world. Welcome back.

```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/hello")
def hello_world():
    return render_template("get_details.html")

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

So, I will just create another template, I will leave this as it is, I will create a new file called get details. Let me just open another tab in the terminal so that it is easy to work with yeah, cd template I will create another file called get_details.html.

In that template we will create an html form and get the user input. So let us I am just going to paste it and explain it to you. Simple html document it just has a form and it has two fields; one is an input text, other one is the submission. So I am just going to put, it is going to be the simplest form you will ever see. I will save this. Now, let me go back to your application instead of serving this I want to actually serve this get_details.html so we can get the user's name. get_details.html, okay.

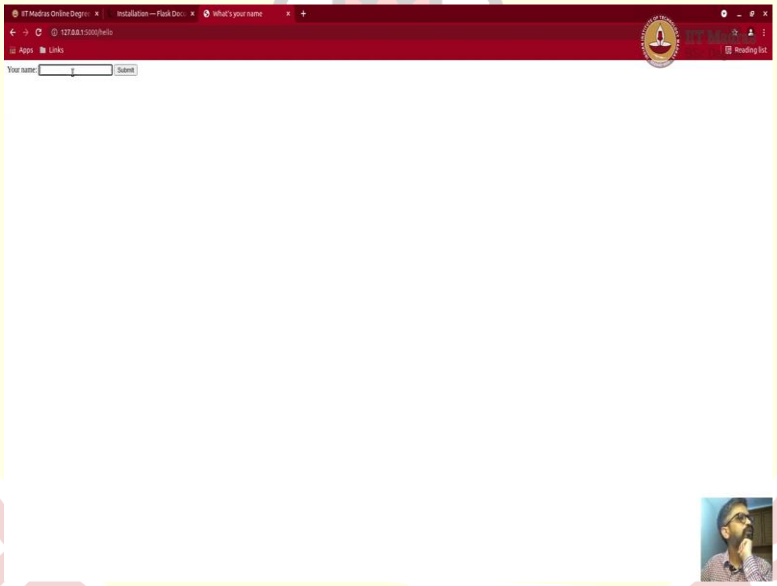
(Refer Slide Time: 15:36)



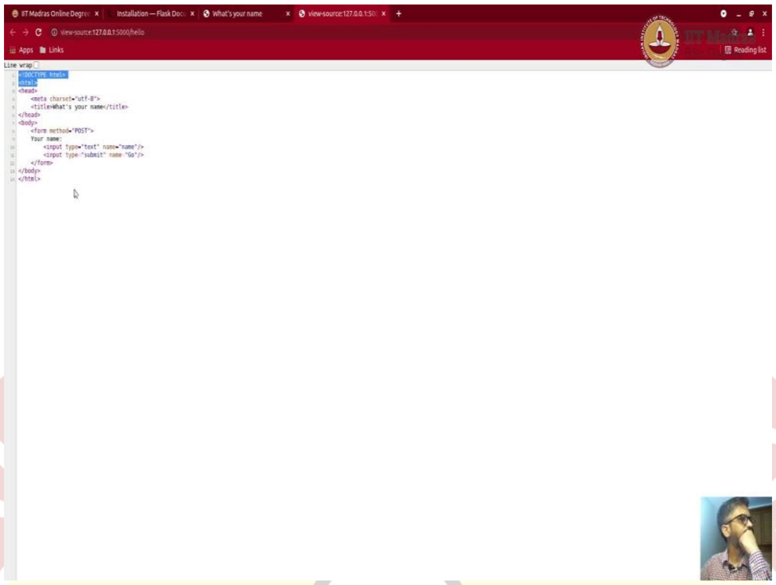
Hello world. Welcome back.

```
from flask import Flask
from flask import render_te
```

```
   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: 758-659-605
127.0.0.1 - - [11/Aug/2021 03:51:37] "GET /hello HTTP/1.1" 200 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
thej@uma ~/Documents/mad1/experiment3
  > python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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
```



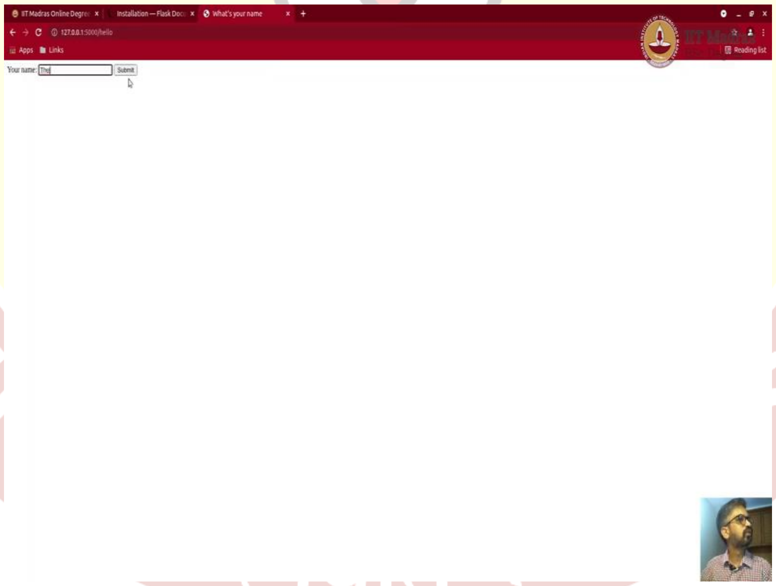
Your name: Submit

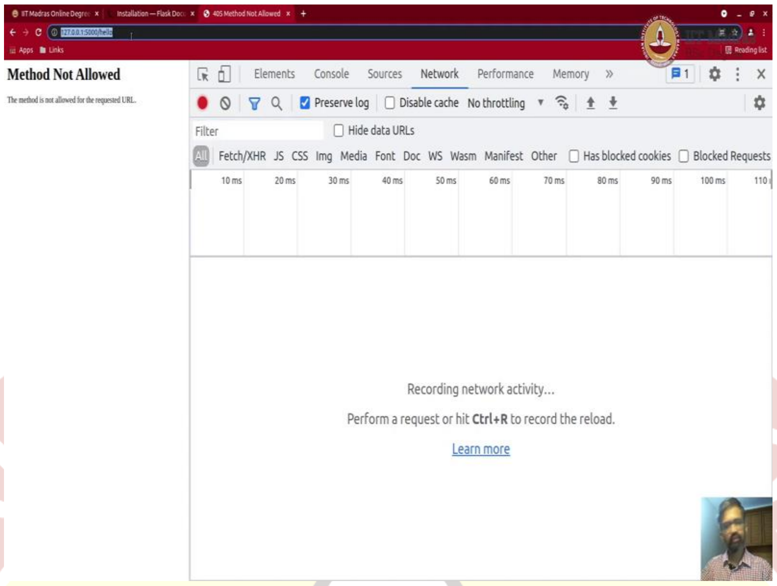


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>What's your name</title>
</head>
<body>
    <form method="POST">
    Your name:
        <input type="text" name="name"/>
        <input type="submit" name="Go"/>
    </form>
</body>
</html>
```

Now, let us start or restart the server and try it, I am just going to stop it, start it again and go back here and refresh. Now you can see that we have a form with a text and a submit button you can see the same thing. Now what happens if you put a name?

(Refer Slide Time: 16:03)



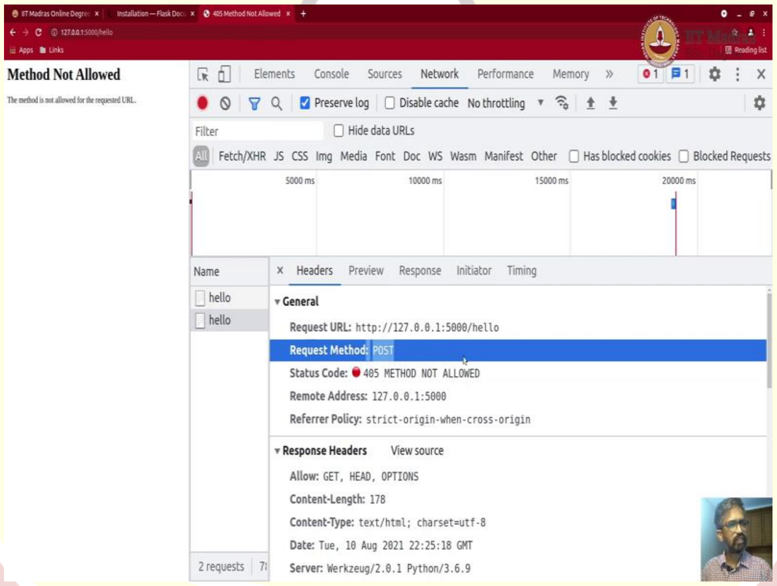

Method Not Allowed

The method is not allowed for the requested URL.

Recording network activity...

Perform a request or hit **Ctrl+R** to record the reload.

Learn more

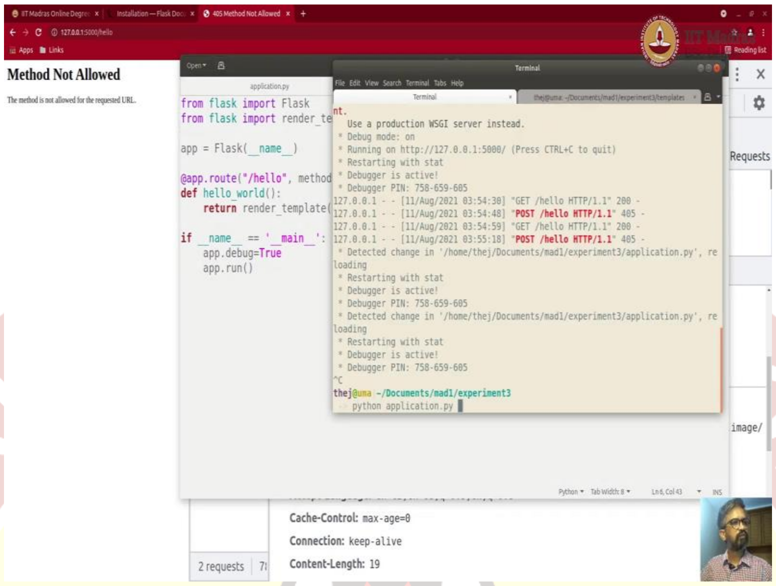

Method Not Allowed

The method is not allowed for the requested URL.

```
Name        Headers  Preview  Response  Initiator  Timing
hello       ▾ General
hello         Request URL: http://127.0.0.1:5000/hello
              Request Method: POST
              Status Code: ● 405 METHOD NOT ALLOWED
              Remote Address: 127.0.0.1:5000
              Referrer Policy: strict-origin-when-cross-origin
            ▾ Response Headers     View source
              Allow: GET, HEAD, OPTIONS
              Content-Length: 178
              Content-Type: text/html; charset=utf-8
              Date: Tue, 10 Aug 2021 22:25:18 GMT
              Server: Werkzeug/2.0.1 Python/3.6.9
2 requests
```

Let us say I am going to put my name Thej click on submit it is trying to submit it is doing something but we are getting error. Let us check the inspect network, let us reload it again. It is getting hello, getting the form, headers and there is 200 okay I am using the request method get and when I click Thej you enter Thej click submit checks are trying to do a post but the server is telling me there is no post allowed that is because we have not added what are the types of methods allowed.

(Refer Slide Time: 16:56)



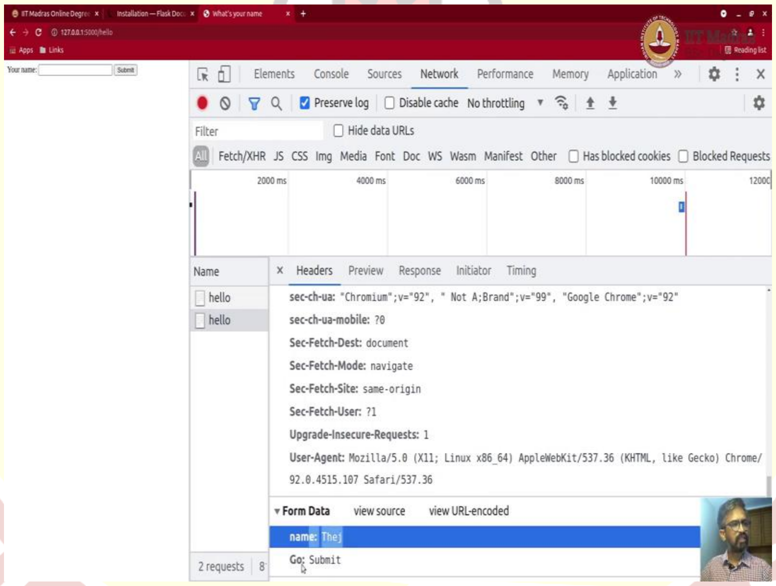
Method Not Allowed

The method is not allowed for the requested URL.

```
from flask import Flask
from flask import render_te
 
app = Flask(__name__)
 
@app.route("/hello", method
def hello_world():
    return render_template(
 
if __name__ == '__main__':
    app.debug=True
    app.run()
```

```
nt.
   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: 758-659-605
127.0.0.1 - - [11/Aug/2021 03:54:30] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 03:54:48] "POST /hello HTTP/1.1" 405 -
127.0.0.1 - - [11/Aug/2021 03:54:59] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 03:55:18] "POST /hello HTTP/1.1" 405 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
thej@uma ~/Documents/mad1/experiment3
   python application.py
```

```
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 19
```



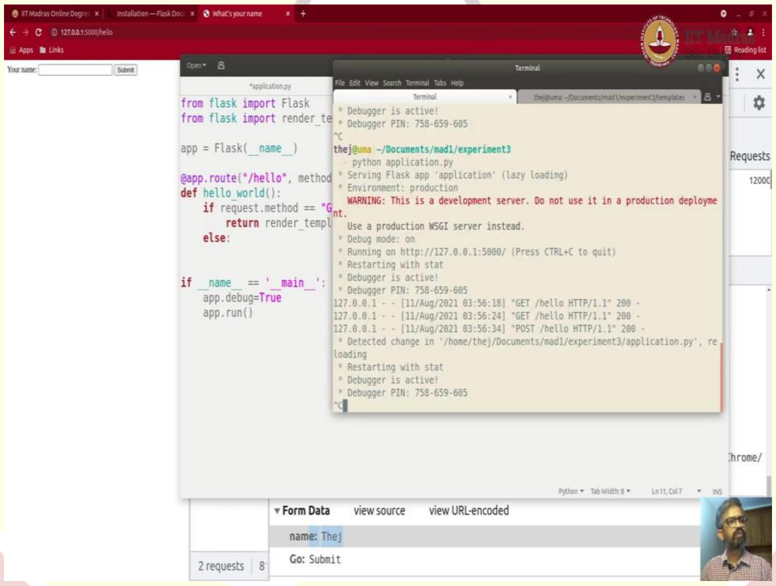
```
sec-ch-ua: "Chromium";v="92", " Not A;Brand";v="99", "Google Chrome";v="92"
sec-ch-ua-mobile: ?0
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
92.0.4515.107 Safari/537.36

Form Data
name: Thej
Go: Submit
```

Usually by default when you do @app.route() it allows only get methods. Now, we can add other methods here I mean by default I mean we have not added get thing but it would have looked like this if I add it. But we want post as well, so we will add post. So now what happens; let us stop and restart and see what happens. It is going to refresh, let me just clear this and then refresh.

And then put Thej here we got the get request going, we got the response everything now we entered the data and then submit. Now if you see the second one the post happened and the post you can see all the details and the form data is also been sent as part of post you can see your name is Thej and go is submit. But nothing is happening it is not showing anything that is because whether to get a post we are just returning the same html, so basically it is reading the same thing again after the submit two.

So, what we could do is we could do if request.method==GET then show the user a form else do something. What we want to do is actually read the value of submitted form element and then display to him. Now, let us create another template to display it, or to keep it simple yes I am going to call that as display_details.html. I am going to create another template file, I am going to go to my command prompt, stop the server and create another template.

(Refer Slide Time: 19:28)



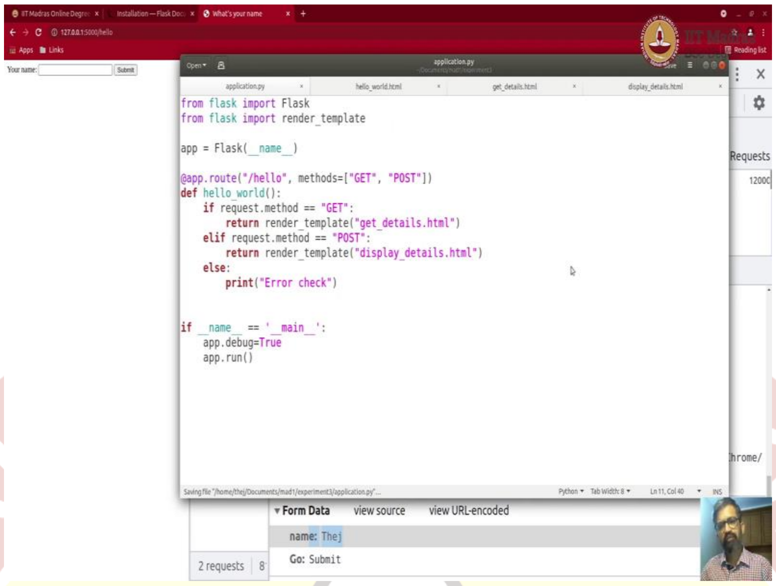
```
from flask import Flask
from flask import render_te

app = Flask(__name__)

@app.route("/hello", method
def hello_world():
    if request.method == "G
        return render_templ
    else:


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

```
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
thej@uma ~/Documents/mad1/experiment3
 python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
127.0.0.1 - - [11/Aug/2021 03:56:18] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 03:56:24] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 03:56:34] "POST /hello HTTP/1.1" 200 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
```

Form Data

name: Thej

Go: Submit

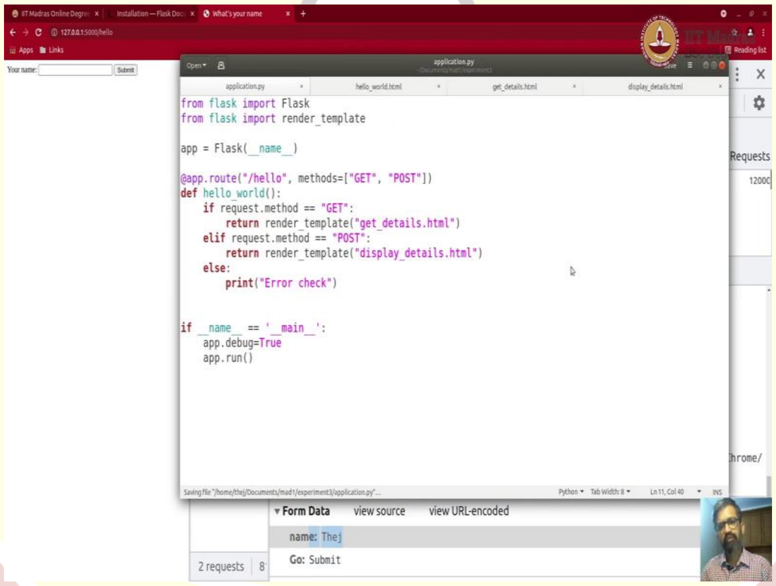


```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        return render_template("display_details.html")
    else:
        print("Error check")


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

Form Data view source view URL-encoded

name: Thej

Go: Submit



```
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        return render_template("display_details.html")
    else:
        print("Error check")


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

Form Data view source view URL-encoded

name: Thej

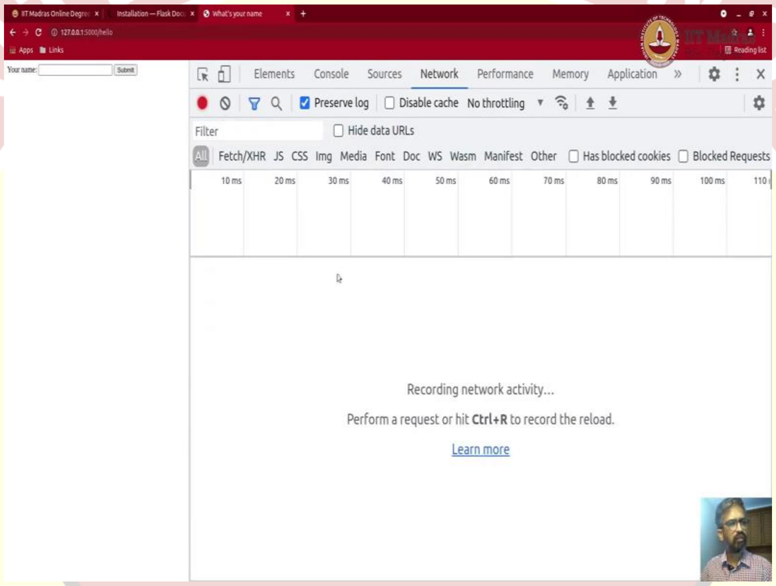
Go: Submit

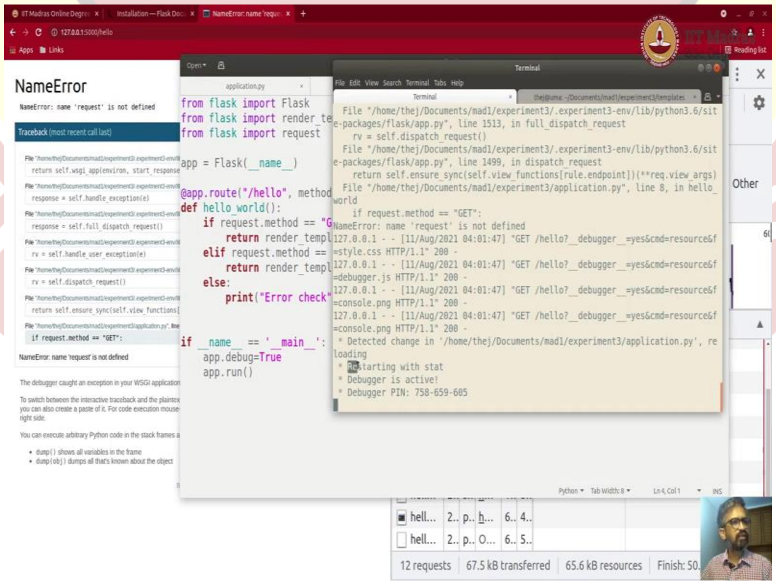
Now, in this template I am going to do simple html as well, I am going to say hello here is the jinja template it is like I said flask internally uses jinja, so I am just printing the name variable display name and I am going do. I have this template ready here let us use the template here, else we can do else but there are many methods like this is, this allows only get and post so simple you can do.

Either if it is gets or it will be post but if you had many methods like let us say delete as well then you have to do if method is called to get, if method is called to post, if method is called to delete. So you could do that but I am just going to use else if so that it is simple and it is always clear and understandable.

If somebody watches it, they will clearly know something is going on here, what is going on here. So if it is else then I do not know what to do because it should never happen in our case so I am just going to print to my log for my own in usage, this would not be shown to user this is for my own use. So, if it is post I am going to display details. Let us stop and run the server restarting, oh there is some error here, so I am going to use spaces everywhere instead of tab. So it started.

(Refer Slide Time: 22:12)

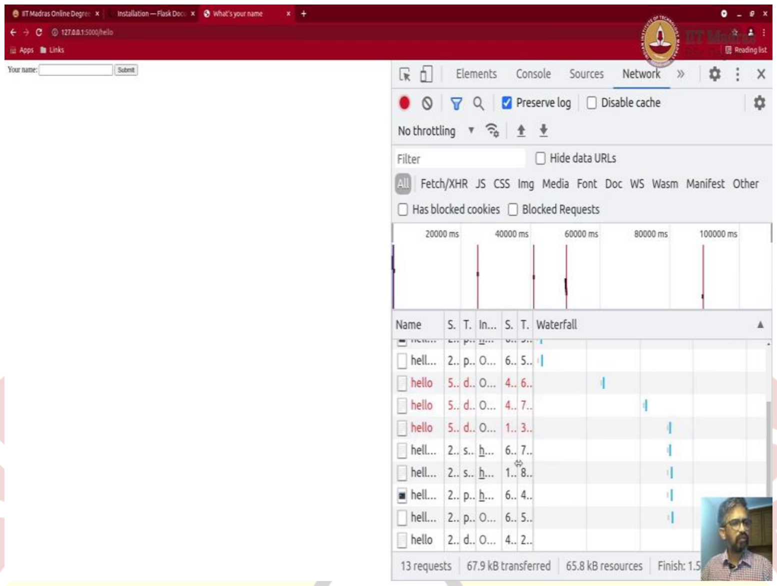



NameError

NameError: name 'request' is not defined

```
from flask import Flask
from flask import render_te
from flask import request

app = Flask(__name__)

@app.route("/hello", method
def hello_world():
    if request.method == "G
        return render_templ
    elif request.method ==
        return render_templ
    else:
        print("Error check"


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

```
  File "/home/thej/Documents/mad1/experiment3/.experiment3-env/lib/python3.6/sit
e-packages/flask/app.py", line 1513, in full_dispatch_request
    rv = self.dispatch_request()
  File "/home/thej/Documents/mad1/experiment3/.experiment3-env/lib/python3.6/sit
e-packages/flask/app.py", line 1499, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
  File "/home/thej/Documents/mad1/experiment3/application.py", line 8, in hello_
world
    if request.method == "GET":
NameError: name 'request' is not defined
127.0.0.1 - - [11/Aug/2021 04:01:47] "GET /hello?__debugger__=yes&cmd=resource&f
=style.css HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:01:47] "GET /hello?__debugger__=yes&cmd=resource&f
=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:01:47] "GET /hello?__debugger__=yes&cmd=resource&f
=console.png HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:01:47] "GET /hello?__debugger__=yes&cmd=resource&f
=console.png HTTP/1.1" 200 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
```

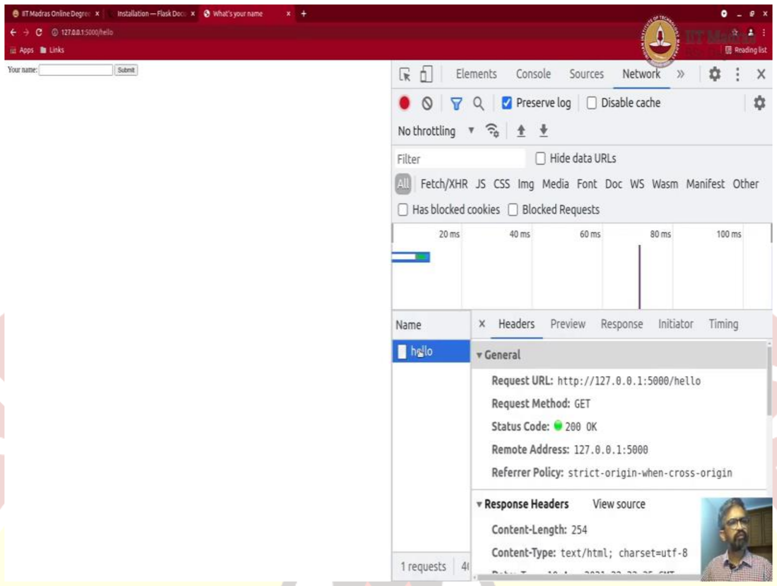


Now, let us go back here clear all the requests to here, oh there is some error in the flask. So this is why it is important to have this debug as true. If this is not there then it would not show this error you can always see. Let us say if I do that this is how it will be in the production because you do not want to show any internal error in the production because it could leak some sensitive information. But when you are running locally you can set debug to true, so that you can see the actual error.

Let me restart the server, it is saying the request is not defined; saying request.method in this part of the file application dot.py in line 8, here it is throwing error here. And it is saying request is not defined where is this request coming from? Okay what we have missed is importing the request from flask. Now, let us save it, let us see if the server has restarted, it has restarted, so we can refresh. Now, let me just clear everything and refresh again.

INDIAN INSTITUTE OF TECHNOLOGY MADRAS
सिद्धिर्भवति कर्मजा

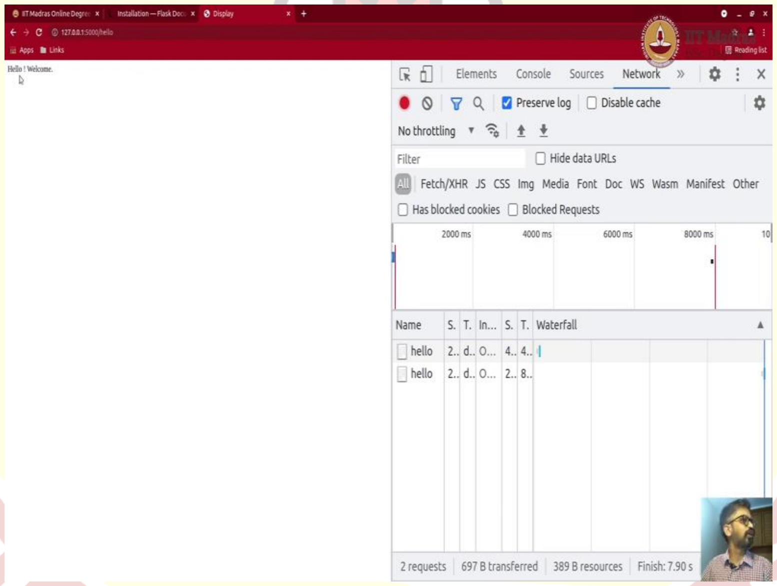
(Refer Slide Time: 23:55)



Your name: [Submit]

```
Request URL: http://127.0.0.1:5000/hello
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:5000
Referrer Policy: strict-origin-when-cross-origin
```

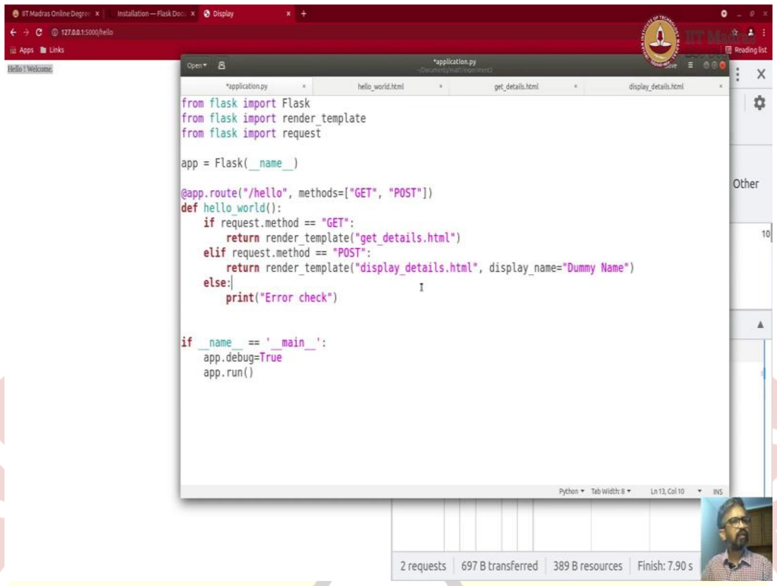
```
Response Headers
Content-Length: 254
Content-Type: text/html; charset=utf-8
```



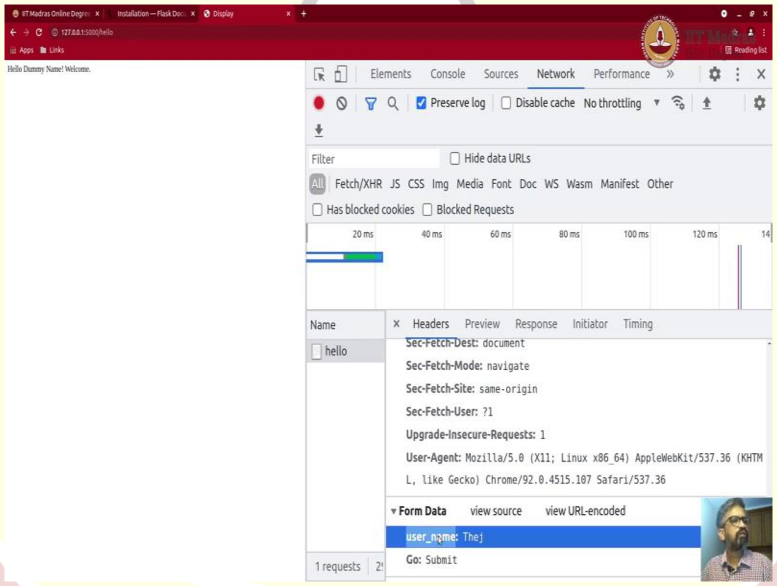
Hello ! Welcome.

Name	S.	T.	In...	S.	T.	Waterfall
hello	2..	d..	O...	4..	4..	
hello	2..	d..	O...	2..	8..	

```
2 requests | 697 B transferred | 389 B resources | Finish: 7.90 s
```



```python
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        return render_template("display_details.html", display_name="Dummy Name")
    else:
        print("Error check")


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



Hello Dummy Name! Welcome.

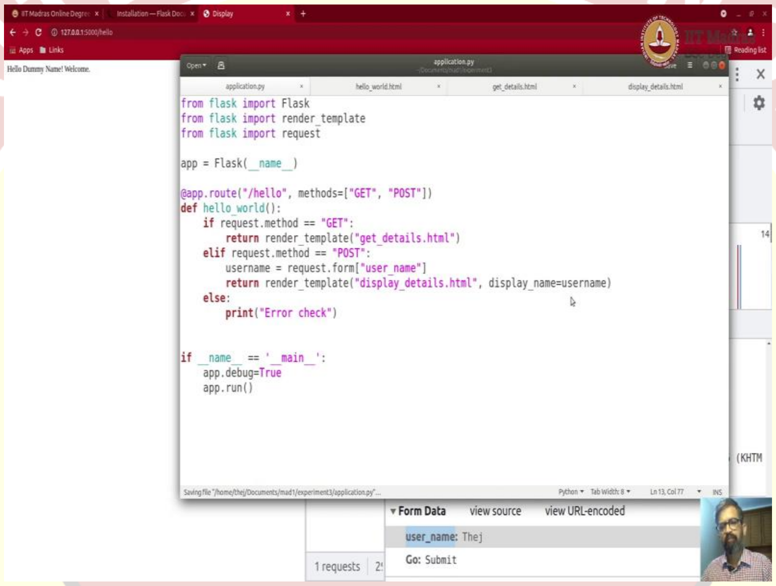
```
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36

Form Data
user_name: Thej
Go: Submit
```

Okay here it is, so 200 get everything is fine, now if you do Thej do submit oh, something is wrong here, it says hello and welcome which means let us come to the post and come to the display_details.html. Oh! it is not showing this display name that is because we are not passing any values to the template.

We can I put a dummy value there now let us see dummy name and if I save it and if I run it, do it shows dummy name that is correct, now the template is working but the thing is we have to get actually the name from the form that got submitted. Let us go back, clear this, enter the value; submit.

Here in the post, we are actually submitting form data the key is name; value is Thej which is same as here the key is actually the name; name of the input. Well, you can also change to your name just to make it a clear username, let us make it username. Now just go back, refresh it, I will put Thej and submit it. So the username is Thej, so we have to pull out this username from the form data and then use it our template, that is the next step. So that will make it very clear.

(Refer Slide Time: 26:16)

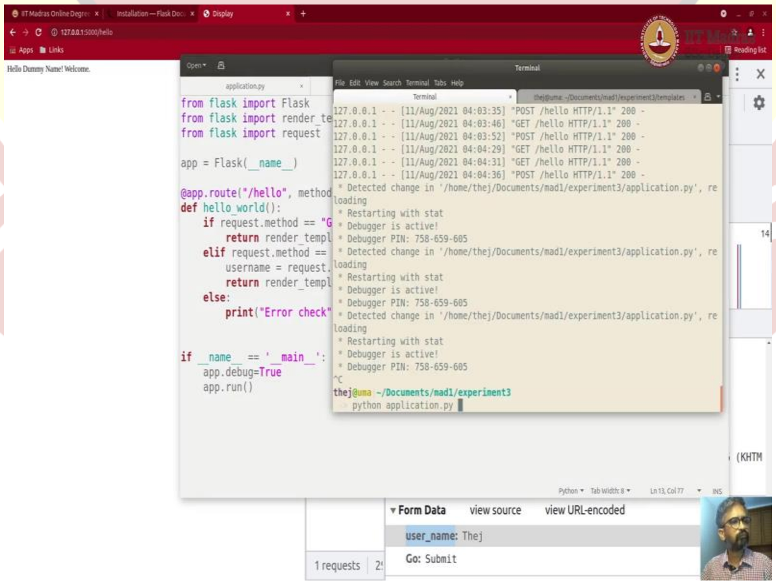


```
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        username = request.form["user_name"]
        return render_template("display_details.html", display_name=username)
    else:
        print("Error check")


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

Form Data

user_name: Thej

Go: Submit

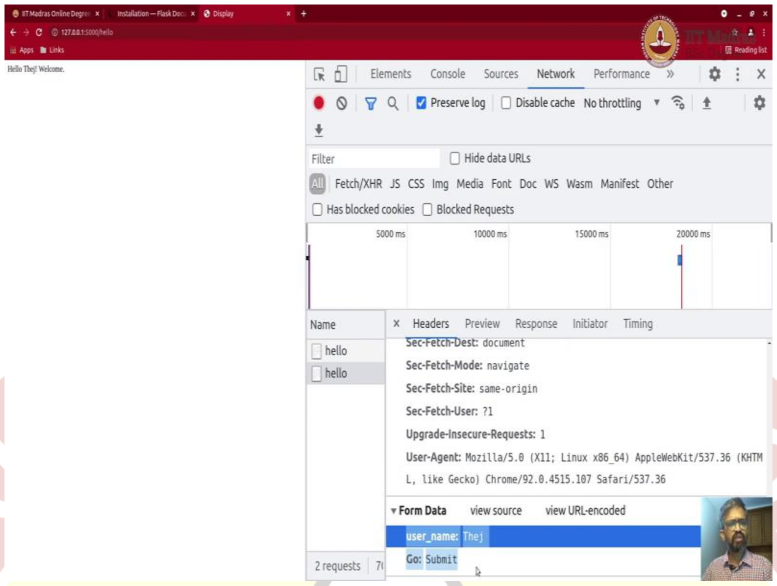


```
127.0.0.1 - - [11/Aug/2021 04:03:35] "POST /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:03:46] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:03:52] "POST /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:04:29] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:04:31] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:04:36] "POST /hello HTTP/1.1" 200 -
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
 * Detected change in '/home/thej/Documents/mad1/experiment3/application.py', re
loading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 758-659-605
^C
thej@uma ~/Documents/mad1/experiment3
   python application.py
```

Form Data

user_name: Thej

Go: Submit

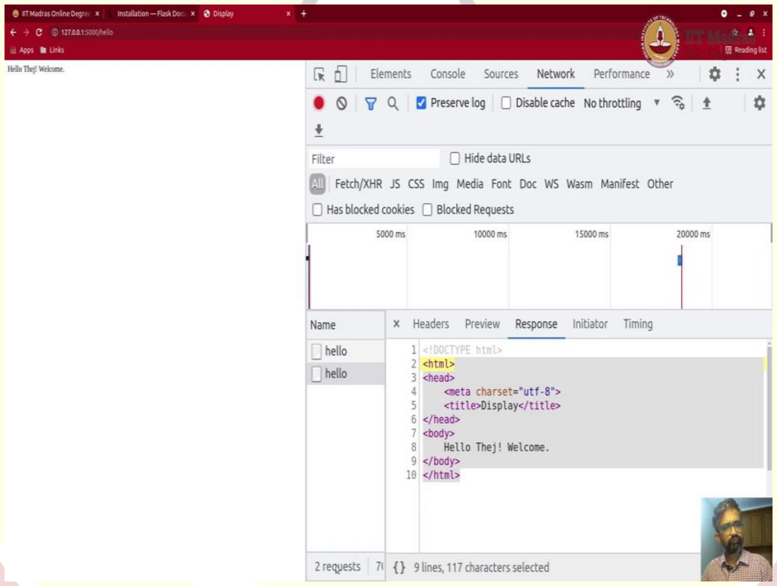


Hello Thej! Welcome.

```
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36
```

Form Data

```
user_name: Thej
Go: Submit
```



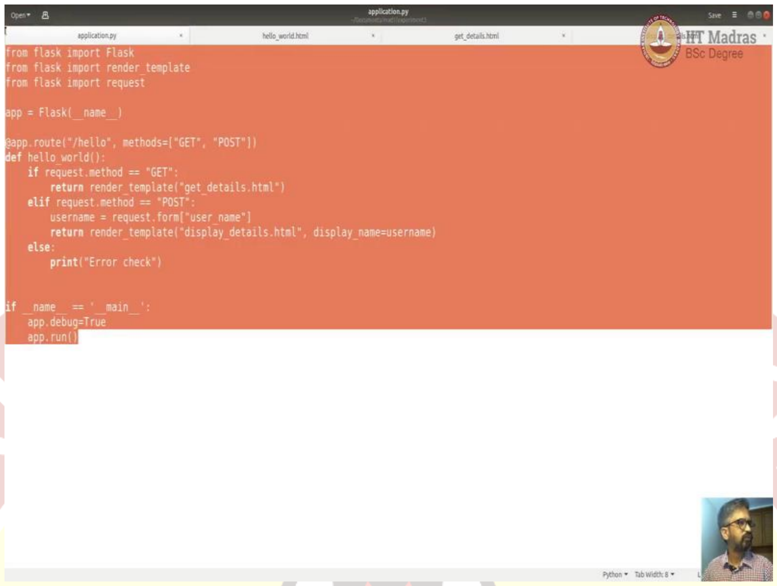
Hello Thej! Welcome.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Display</title>
</head>
<body>
    Hello Thej! Welcome.
</body>
</html>
```

So, that is done by getting the form from the request. So you can do the same way as we did for method, you can do request dot form should return the form that was submitted and since it was user name call it username and will assign it to a variable, local variable called username, and then we will pass this username to our jinja template right.

Now let us see, let us go back, clear, refresh, we got a get and get comes here and it gets get details; get details prints this form which has a method post and it has two form elements and then when someone fills the name and clicks and submit it does post and post sends the username as Thej and go as data then there is a response, the response is under here which is hello Thej.

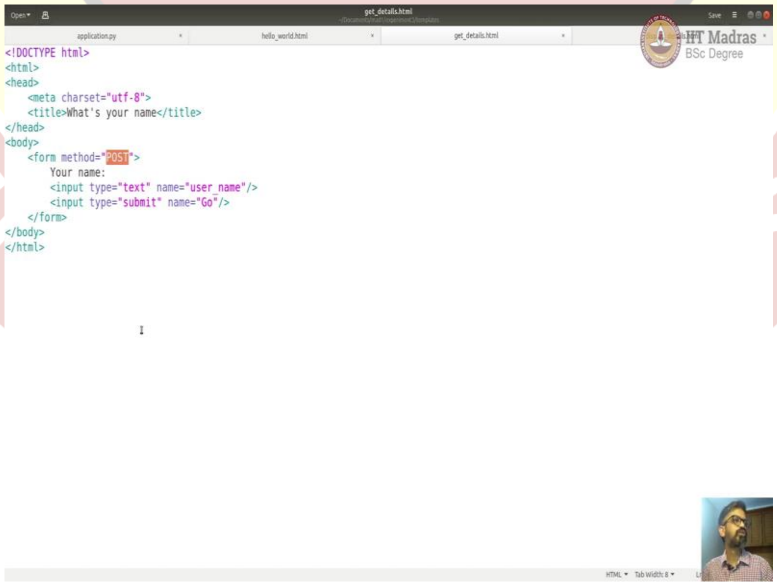
(Refer Slide Time: 27:55)



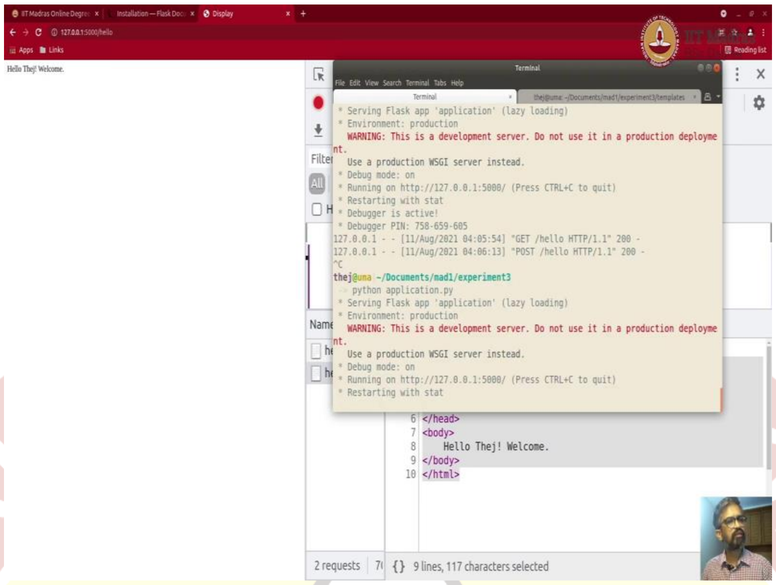
```python
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        username = request.form["user_name"]
        return render_template("display_details.html", display_name=username)
    else:
        print("Error check")


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

So, now we have done one whole round trip in this simple flask py, so what we have done to recap is to create a URL endpoint where you can when somebody does a get access we will send them a form to fill. Once they fill up the form and send a post request we will read the form elements and display the message to the user. Now there are a couple of interesting things happening here.

(Refer Slide Time: 28:30)



```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>What's your name</title>
</head>
<body>
    <form method="POST">
        Your name:
        <input type="text" name="user_name"/>
        <input type="submit" name="Go"/>
    </form>
</body>
</html>
```

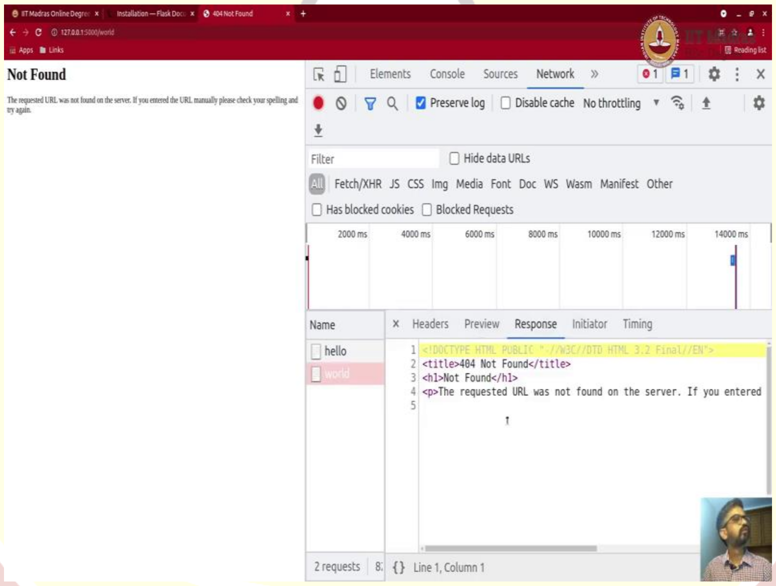


Hello Thej! Welcome.

```
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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: 758-659-605
127.0.0.1 - - [11/Aug/2021 04:05:54] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2021 04:06:13] "POST /hello HTTP/1.1" 200 -
^C
thej@uma ~/Documents/mad1/experiment3
 -> python application.py
 * Serving Flask app 'application' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   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
```

```
</head>
<body>
    Hello Thej! Welcome.
</body>
</html>
```

2 requests | 9 lines, 117 characters selected

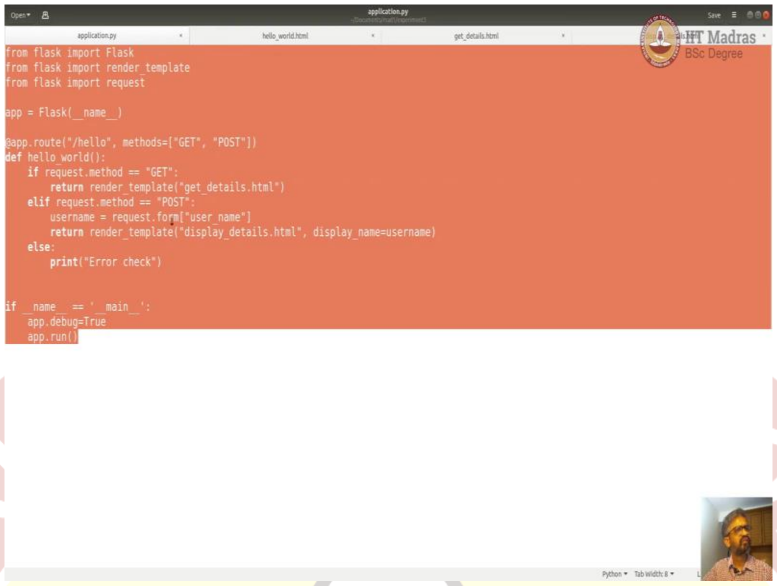


127.0.0.1:5000/world

Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered
```



```python
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

@app.route("/hello", methods=["GET", "POST"])
def hello_world():
    if request.method == "GET":
        return render_template("get_details.html")
    elif request.method == "POST":
        username = request.form["user_name"]
        return render_template("display_details.html", display_name=username)
    else:
        print("Error check")


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

In the form here we are using a method called post you could have also used get but post is the best and it has more, it is meant to submit things and then get is usually used to get the documents or the information from the web server, post is usually used to send it I mean you could use get to send as well but it is not usually advised.

Then it is actually submitting to the same URL point for example you can see the submission is also to the same URL end point getting also from the same thing, submitting also same thing. But you can change it, you can define an action let us say we can call it world, if I do this let us see what happens once I submit.

Let me just double check the server has restarted let me just do it manually, let me just clear it, get request has happened you can see inside the browser the action is world and post, method is post. Now, if I enter a name click a submit it actually submits to a URL endpoint world which is not there, which we saw here, which there is only hello there is no world.

But you could write if you want it to be handled separately you can write another function and add a route to route to that part. This is it actually this is a simple web application where user can submit things and get information back all in one single file with some template files that is it for today and that is it from this screencast, thank you so much. Bye.