Python & Flask : Web Development

* **Flask is a micro web framework written in python.**

# **Getting Started:**

Steps:

1. Create a virtual environment and activate it.

**python -m venv venv**

**.\venv\Scripts\activate**

1. Install flask

**pip install flask**

* Import Flask from flask module
* Initialize the application
* Create the route on the server and define a function
* Here we are routing some function , the function we are routing to is home function and the home function is returning “hello world”

#import Flask from flask module

from flask import Flask

#Initialize the application

app = Flask(\_\_name\_\_) #this is a general convention to initialize flask app , this line is creating the application

@app.route('/')  #here we are creating a route on our server that is /, once we create a route, we define a function

def home():

    return "Hello World"

#Here we are routing some function, and the function we are routing to is home function and the home function is returning hello world

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

    app.run()

# **Routing & Variable Rules:**

########## routing and Variable rules

#how to access different URLs and web pages

@app.route('/about')

def about():

    return "The aboue page"

@app.route("/blog")

def blog():

    return "this is the blog"

#here we are defining the actual route the server should go to and what it should return.

#variable rules

@app.route("/blog/<blog\_id>")

def blogpost(blog\_id):

    return "this is the blogpost number " + str(blog\_id)

#here we are sepcifying a rule inside the route

#when one tries to access anything /blog/something refer to that something as a variable = blog\_id

#like this we specify dfferent variable rules

#we can also write it as

# @app.route("/blog/string:blog\_id")

# def blogpost(blog\_id):

#     return "This is the blogpost number " + blog\_id  #here we do not need the string

# Understanding Templates and Jinja

@app.route("/blog")

def blog():

    return '''

    <html>

    <head></head>

    <body>

        <h2> Welcome to this blog! </h2>

        <p> I am apurva the creator of this blog </p>

    </body>

    '''

#here we have rendered the HTML code

#we can do this by using templates which are blueprints of HTML code,

Blog.html

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset="UTF-8">

    <title> Blog</title>

</head>

<body>

    <h2> Welcome to this blog! </h2>

    <p> I am adam the creator of this blog </p>

</body>

</html>

@app.route("/blog")

def blog():

    return render\_template('blog.html')



* Jinja 2 is the template engine that comes with flask framework, which allows us to input in variables and data to our templates.

{{}} this is the jinja syntax.

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset="UTF-8">

    <title> Blog</title>

</head>

<body>

    <h2> Welcome to this blog! </h2>

    <p> I am {{ author }} the creator of this blog </p>

</body>

</html>

@app.route("/blog")

def blog():

    return render\_template('blog.html', author = "Bob")

# Diving into Jinja2

* we can add conditional statements and loops in the jinja template, we define sunny = True in our route, if its sunny “ Today is sunny” if not then “Today is rainy”

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset="UTF-8">

    <title> Blog</title>

</head>

<body>

    <h2> Welcome to this blog! </h2>

    <p> I am {{author }} the creator of this blog </p>

    {% if sunny %}

    <p> Today is sunny </p>

    {% else %}

    <p> Today is rainy </p>

    {% endif %}

</body>

</html>

@app.route("/blog")

def blog():

    return render\_template('blog.html', author = "apurva", sunny = True)

A white background with black text

Description automatically generated

This is how we do conditional statement in jinja

We can also do looping here

A screenshot of a white background

Description automatically generated

@app.route("/blog")

def blog():

    posts = [{"title" : "Technology in 2019 ", "author": "Avi"},

             {"title" : "Expansion of Oil in Russia", "author" : "Bob"} ]  #this are post titles

    return render\_template('blog.html', author = "apurva", sunny = True, posts = posts)

<!DOCTYPE html>

<html lang='en'>

<head>

    <meta charset="UTF-8">

    <title> Blog</title>

</head>

<body>

    <h2> Welcome to this blog! </h2>

    <p> I am {{author }} the creator of this blog </p>

    {% if sunny %}

    <p> Today is sunny </p>

    {% else %}

    <p> Today is rainy </p>

    {% endif %}

    {% for post in posts %}

        <h2>{{post.title }}</h2>

        <h3> Author : {{post.author}}</h3>

    {% endfor %}

</body>

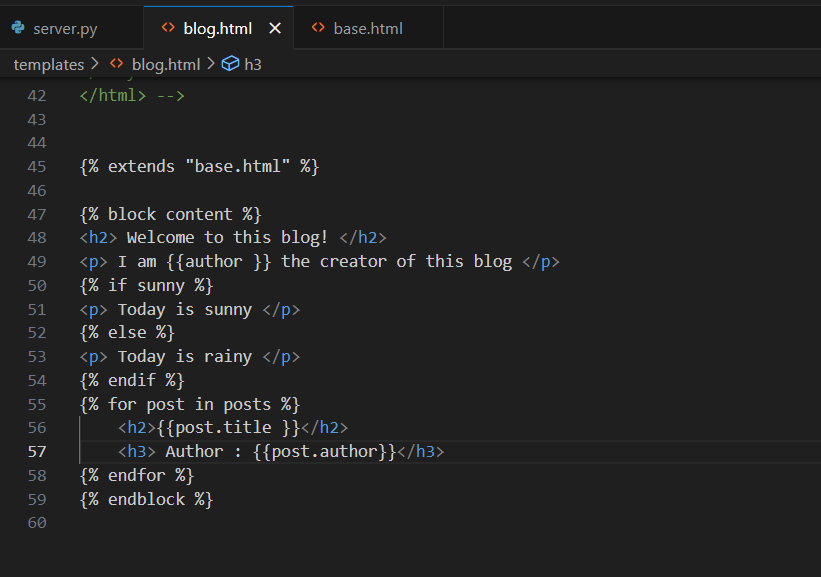
</html>

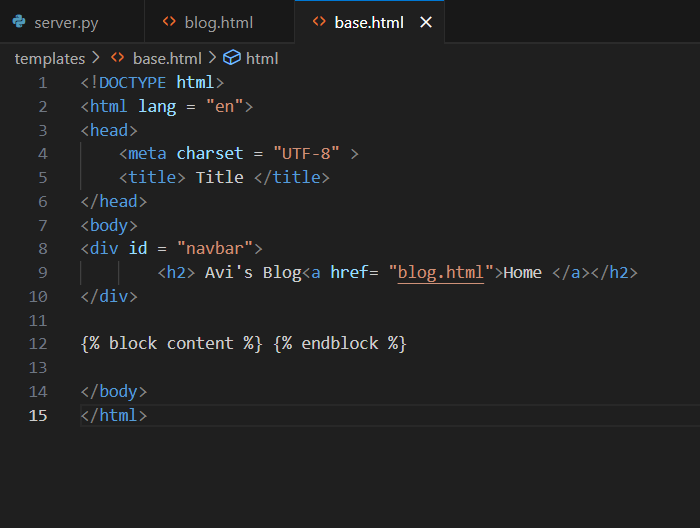
By this we can iterate through data and show the data in proper way.

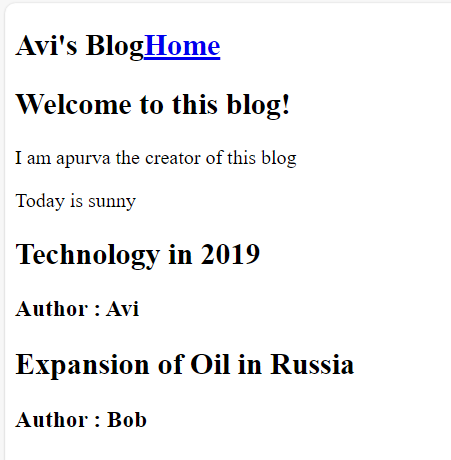
# Template Inheritance & Block content

- What is template inheritance?

* How many times we are repeating the code every time we create a website?
* We can keep the common content in base.html file. It is a sort of a template that we can inherit from to render any other webpage.
* In blog.html file keep the main code as it is and remove everything , create a new html file
* We create a simple nav bar using html.
* The base file code, which every single webpage will inherit after navbar write correct jinja type.
* Bloking content means, if anything gets extended that will get added into block content.
* Extend blog.html, define block content and end block in that.







############## base.html ####################

<!DOCTYPE html>  
<html lang = "en">  
<head>  
<meta charset = "UTF-8" >  
<title> Title </title>  
</head>  
<body>  
<div id = "navbar">  
<h2> Avi's Blog<a href= "blog.html">Home </a></h2>  
</div>

{% block content %} {% endblock %}

</body>  
</html>

################### blog.html ###################

{% extends "base.html" %}

{% block content %}  
<h2> Welcome to this blog! </h2>  
<p> I am {{author }} the creator of this blog </p>  
{% if sunny %}  
<p> Today is sunny </p>  
{% else %}  
<p> Today is rainy </p>   
{% endif %}  
{% for post in posts %}  
<h2>{{post.title }}</h2>  
<h3> Author : {{post.author}}</h3>  
{% endfor %}  
{% endblock %}

* so the base.html code will always be there, we don’t have to repeat it, and we can use extend the code for other information using template inheritance.

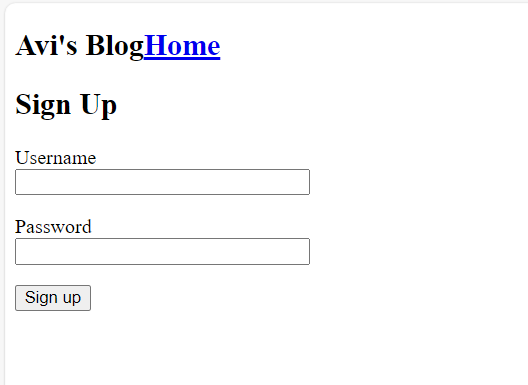
# 

# Web forms with Flask-wtf

* we need to install Flask-wtf module which allows us to integrate wt forms & web forms in general in flask.
* **pip install flask-wtf**
* create a secret key for the flask app
* app = Flask(\_\_name\_\_)

app.config[‘SECRET\_KEY’] = ‘secret\_key’ (some secret key that will be used while using the app which must be secret and should not be disclosed)

* after this we create a web form , like a signup form which will be used where users can sign up, for that create a new py file forms.py
* WTForms has different types of fields
* After creating the SignUpForm class, we must create a html page to render this form in our own site
* For that create a new file signup.html , which will extend base.html and will also use block.content
* In the form we are going to create three different fields , we will create object of the signup form, we will pass that object into this view using jinja, and once we form an object we are going to refer to form.usename, form.password, form.submit (the fields which we have created and then call them in signup.html)
* Define form action, methods, create form using html and then render this template using server.py
* Create a new route for signup and run the application.



# Getting Data from Web Forms

* Create a user.html page
* Here we are trying to see when the form gets submitted,
* Here in signup route we render the form using “GET” request and get the data using “POST” request

@app.route("/signup", methods = ['GET', 'POST'])

def signup():

    form = SignUpForm()

    if form.is\_submitted():

        result = request.form

        return render\_template("user.html", result = result)

    return render\_template ('signup.html', form = form)

* Result of this is in dictionary form, so we can iterate through it and using the key get the values and display to the user.
* We create a table in user.html, for key and value, we are rendering key and value.

**User.html**

{% extends "base.html" %}

{% block content %}

<table border = "1">

    {% for key, value in result.items() %}

    <tr>

        <th>

            {{key}}

        </th>

        <td>

            {{value}}

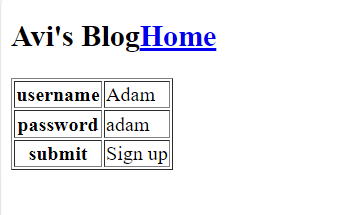
        </td>

    </tr>

    {% endfor %}

</table>

{% endblock %}



* So here we are rendering user.html in the page with our data.
* This is how we get the data from the form when it is submitted.

***This is basic Web development with Flask and Python.***