• Introduction to embedding HTML within Python using web frameworks like Django or Flask.

* Embedding HTML within Python using web frameworks like Django or Flask enables developers to create dynamic and interactive web applications. These frameworks provide powerful tools to integrate HTML, CSS, and JavaScript with Python, facilitating the creation of modern and efficient websites.

**Part 1: Using Django**

* **Django** is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

1. **Setting Up Django**:
   * **Install Django**:

pip install django

* + **Create a New Project and Application**:

django-admin startproject myproject

cd myproject

python manage.py startapp myapp

1. **Creating a Template**:
   * **Template File**: Create an index.html file in myapp/templates/myapp/:

* html

<!DOCTYPE html>

<html>

<head>

<title>Welcome to Digital Society</title>

</head>

<body>

<h1>Welcome to Digital Society</h1>

<p>This is a dynamic web page served using Django.</p>

</body>

</html>

1. **View Function**:
   * **Definition**: Define a view function in myapp/views.py to render the template:

* python

from django.shortcuts import render

def index(request):

return render(request, 'myapp/index.html')

1. **URL Configuration**:
   * **Mapping the View**: Map the view to a URL in myproject/urls.py:

* python

from django.urls import path

from myapp import views

urlpatterns = [

path('', views.index, name='index'),

]

**Part 2: Using Flask**

* **Flask** is a lightweight WSGI web application framework designed to make getting started quick and easy, with the ability to scale up to complex applications.

1. **Setting Up Flask**:
   * **Install Flask**:

pip install flask

1. **Creating the Application**:
   * **Basic Flask App**: Create an app.py file with the following code:

* python

from flask import Flask, render\_template

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

@app.route('/')

def index():

return render\_template('index.html')

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

app.run(debug=True)

1. **Creating a Template**:
   * **Template File**: Create an index.html file in the templates directory:

* html

<!DOCTYPE html>

<html>

<head>

<title>Welcome to Digital Society</title>

</head>

<body>

<h1>Welcome to Digital Society</h1>

<p>This is a dynamic web page served using Flask.</p>

</body>

</html>

**Conclusion**

By following these steps, you can successfully embed HTML within Python using Django or Flask frameworks. This integration allows you to create dynamic and interactive web applications that combine the power of Python with the flexibility of HTML, CSS, and JavaScript.

• Generating dynamic HTML content using Django templates.

**Generating Dynamic HTML Content Using Django Templates**

Django templates are a powerful tool for generating dynamic HTML content. They allow you to embed Python code within HTML to create dynamic, data-driven web pages. Let's go through the key concepts and steps involved in generating dynamic HTML content using Django templates.

**Key Concepts**

1. **Templates**:
   * HTML files with embedded Django Template Language (DTL) to render dynamic content.
   * Located in the templates directory of your Django app.
2. **Views**:
   * Python functions that handle web requests and return web responses, typically rendering templates.
3. **Context**:
   * A dictionary of data passed from the view to the template, used to render dynamic content.

**Example: Creating a Dynamic Web Page**

1. **Creating Models**:
   * Define models in myapp/models.py to represent the data structure:

* python

from django.db import models

class Member(models.Model):

name = models.CharField(max\_length=100)

email = models.EmailField()

mobile = models.CharField(max\_length=15)

def \_\_str\_\_(self):

return self.name

* + Apply migrations to create the database schema:

python manage.py makemigrations

python manage.py migrate

1. **Creating Views**:
   * Define a view in myapp/views.py to fetch data and render the template:

* python

from django.shortcuts import render

from .models import Member

def member\_list(request):

members = Member.objects.all()

context = {

'members': members

}

return render(request, 'myapp/member\_list.html', context)

1. **Creating Templates**:
   * Create an HTML template in myapp/templates/myapp/member\_list.html:

* html

<!DOCTYPE html>

<html>

<head>

<title>Member List</title>

</head>

<body>

<h1>Member List</h1>

<table border="1">

<tr>

<th>Name</th>

<th>Email</th>

<th>Mobile</th>

</tr>

{% for member in members %}

<tr>

<td>{{ member.name }}</td>

<td>{{ member.email }}</td>

<td>{{ member.mobile }}</td>

</tr>

{% empty %}

<tr>

<td colspan="3">No members found.</td>

</tr>

{% endfor %}

</table>

</body>

</html>

1. **URL Configuration**:
   * Map the view to a URL in myapp/urls.py:

* python

from django.urls import path

from . import views

urlpatterns = [

path('members/', views.member\_list, name='member\_list'),

]

* + Include the app's URLs in the project's urls.py:
* python

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('myapp.urls')),

]

1. **Running the Server**:
   * Start the Django development server to view the dynamic HTML content:

python manage.py runserver

**Summary**

By following these steps, you can generate dynamic HTML content using Django templates. This approach allows you to create data-driven web pages that render dynamic content based on the context provided by the view. Django's templating engine and powerful model-view-controller (MVC) architecture make it an ideal choice for developing modern web applications.