**Django Framework:**

**Day88: Django Introduction:**

Introduction To Django:

=======================

Python:

X-Application

Core Python

Advanced Python

Framework

Framework:

==========

Is the collection of multiple components which we can use to build the web app/app faster.

Django:

======

==> Django is the popular framework

use with python

for web application development.

==> Django is

free and open source web application framework.

==> backend framework

==> has automatic admin interface

which supports CRUD Operations.

C ==> CREATE

R ==> READ

U ==> UPDATE

D ==> DELETE

====================================================

Content:

========

Setup:

python software

Django

App-1: (Food Menu Application)

MVT (Models, Views and Templates)

Authentication

API (Application Programming Interface)

Rest API (Django Rest Framework)

Pagination

search

filtering

App-2: (Ecommerce Web App)

Shopping cart

checkout

search

order placement etc.

Admin level customization

App-3: Web based CV Generator

web based link scraper.

**DAY89: Django Setup:**

Django Setup:

=============

To work with Django:

prerequisites:

Python software

Django Software

Install the Django Software:

============================

==> pip

==> one command:

$ pip install Django

Current Version ==> 5.1.3

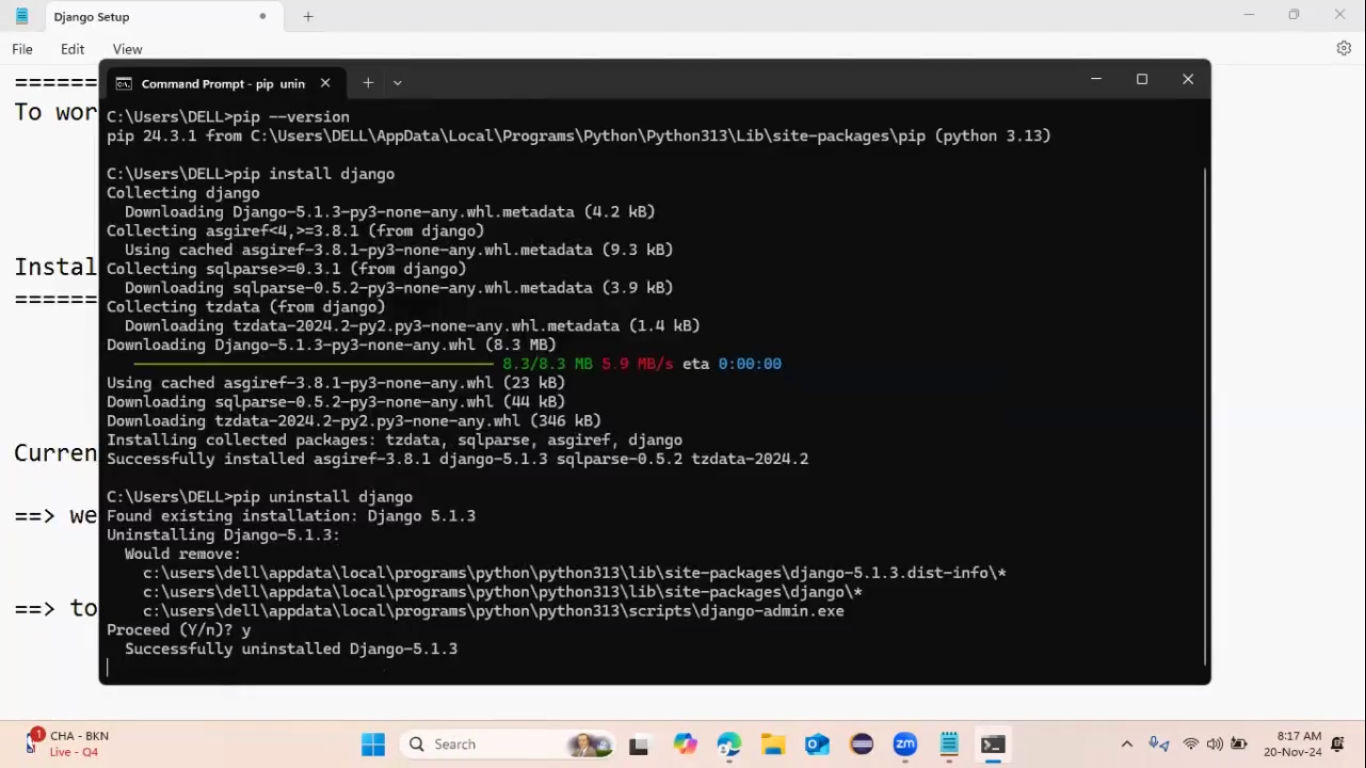
==> we can use the command with specified version also:

$ pip install Django==version



==> to uninstall the Django:

$ pip uninstall Django



============================================

PyCharm ==> Community Edition Professional

free Paid

VS Code Installation:

======================

==> Visual Studio Code

1) Download the VS Code

Chrome ==> search about: vs code download

click on first link

download the file according to your system configuration

run that downloaded file to make start installation

2) Installing the file:

select the radio button "accept the agreement"

click on next

click on next

select the option: "create the desktop shortcut"

click on next

click on install

click on finish

=============================

Project-1:

==========

Food Menu Application

=====================

1) create the food menu application page

2) add food menu items

3) description pages

4) Registration page

5) Login Page

6) Delete items

7) Logout etc.

Django ==> MVT Model

Model Views and Templates

1) we required to create the app

2) add views

3) add templates

4) modal

**Day90: Create and Launch of the Project on the server:**

HOW TO CREATE THE DJANGO PROJECT:

================================

1) Create the folder where you want with any name

2) Open the command prompt

3) Use the command for the navigation to the create folder location:

$ cd folder location

4) Use the command to create/start the project in Django:

$ django-admin startproject title-project

5) Start the VS Code

6) File ==> Open Folder ==> Select your folder

7) Open the created project structure

manage.py:

==========

==> it is the file which allows to interact with the Django project.

Ex: to start the server for existing project, manage.py can be used.

\_\_init\_\_.py:

============

==> an empty file

==> the job of this file is to tell the current directory is the python package

settings.py:

============

==> containing all the settings and configurations of current project

Ex: DB, authentication, URL Format etc.

urls.py:

=======

==> contains all the URL patterns.

=====================================================

Loading the project on the server:

===================================

Open the command prompt

navigate to the project folder

$ cd project folder location

run a command:

$ python manage.py runserver

copy the url & paste on the browser

**Day91**: **Views and URLs in Django:**

What is an app?

===============

==> the individual chunk/part/section of the website/Django project is called as a " Django app".

==> the structure of the app in Django is:

assume app name ==> "foodMenu"

foodMenu

==> migrations

\_\_init\_\_.py

admin.py

apps.py

models.py

tests.py

views.py

Create the app:

open command prompt ==> type a command: "$python manage.py startapp application-name"

Views.py:

=========

==> a python file

can process user request

view ========> 1) take the request

2) analyze the request

3) provide the appropriate response.

How to write a view to our app:

================================

Syntax:

def name-of-the-view(request):

return http-response

http ==> hyper text transfer protocol

How to implement the view on server:

====================================

view must be link with URL.

==> to link the url to your view of app, we can create "urls.py" file.

Creating the urls.py:

====================

How urlpatterns work in Django?

================================

Django ===> ROOT\_URLCONF ==>settings.py ==> urls.py ==> match the send request url ==> call the view ==> Django open in server.

**Day92**: **Database & Models**:

Django Commands:

=================

1) cd ==> navigating your project folder

$ cd your\_project\_folder\_path

2) creation of project

$ django-admin startproject project-name

3) Start VS Code and open your project in VS Code

4) Running or loading of the project on the server

$ python manage.py runserver

5) created app

$ python manage.py startapp application-name

=================================================

Database & Models:

===================

database ==> store the data in table format.

Model:

======

==> Models allow us to create database tables.

==> Models are blueprint/template which can be used to create database tables.

==> to define model, we can use "class"

==> we can add models in "models.py" file.

Ex: Student ==> database

id

name

marks

==============================================

Commands:

=========

1) To check all migrations of project:

(table entries)

$ python manage.py migrate

2) to create a table for our app:

we need to create a model.

Syntax:

open "models.py"

class ClassName(models.Model):

col1 = models.CharField(max\_length = 200)

3) go to "apps.py" file

copy the class name (default generated class by Django)

4) go to "settings.py"

INSTALLED-APPS = [

'app\_name.apps.className',

]

5) make migrations on the server:

$ python manage.py makemigrations appName

6) need to create table from the model which have created.

$ python manage.py sqlmigrate appName AppID

7) run the migrations:

$ python manage.py migrate

**Day93: Date and Time Module:**

Date and Time Module:

=====================

module: datetime

provides different classes to deal with date and time.

Classes:

date

datetime

time

Classes ==> methods

to deal with dates, times, and time intervals.

Two objects:

Date and DateTime

Generate the date:

==================

1) we should import "date" class.

Syntax:

from datetime import date

2) create an object for the date class.

Syntax:

object-name = date(year, month, date)

Here:

year > 0

month ==> 1 to 12

date ==> 1 to 31

from datetime import date

dt = date(1995, 6, 30)

print(dt)

dt1 = date(2024,11,29)

print(dt1)

dt2 = date(2050, 6,20)

print(dt2)

============================================

Getting of the Current date:

============================

today():

=======

==> this is a static method

because it need to access/invoke with the class name.

Syntax:

object = date.today()

from datetime import date

today = date.today() # Static method can always call through the class name

print(today)

==> date class provided some attributes/properties:

year ==> can return the year

month ==> can return the month number

day ==> can return the day number

Syntax:

object-name.year

object-name.month

object-name.day

Note:

year, month and day are the instance variables/properties/attributes for the class "date"

from datetime import date

today = date.today() # Static method can always call through the class name

print(today)

print("The Year = ",today.year)

print("The Month = ",today.month)

print("The Day = ",today.day)

==================================================

Timestamp:

==========

fromtimestamp():

================

==> static method

Syntax:

datetime.fromtimestamp(number/Integer)

Here:

parameter ==> value in sec

Note:

fromtimestamp() can generate the time stamp from the year 1970 and time 5.30 am by default.

from datetime import datetime

timestamp = datetime.fromtimestamp(19876594801)

print(timestamp)

=================================================================

Date to string:

===============

two methods:

1) isoformat()

2) strftime()

isoformat()

===========

==> Static method

Syntax:

date.isoformat(date-object)

strftime()

==========

==> Instance Method

Syntax:

datetime-object.strftime("%Y-%m-%d %H : %M : %S")

from datetime import datetime

today = date.today()

now = datetime.now()

print(type(today))

s = date.isoformat(today)

print("The Date in string format = ",s)

print(type(s))

# s1 = date.strftime(today)

s1 = today.strftime("%Y-%m-%d %H:%M:%S")

print("The date in string = ",s1)

print(type(s1))

s2 = now.strftime("%Y/%m/%d %H : %M : %S")

print(s2)

print(type(s2))

===================================

Time Class:

===========

from datetime import time

myTime = time(16,55,56)

print(myTime)

t1 = time(minute = 55)

print(t1)

t2 = time()

print(t2)

t3 = time(22)

print(t3)

t4 = time(22,55)

print(t4)

t5 = time(hour = 22)

print(t5)

t6 = time(second = 55)

print(t6)

Time class properties:

======================

time-class-object.hour

time-class-object.minute

time-class-object.second

from datetime import time

Time = time(11,23,35)

print(Time)

print("Hours = ",Time.hour)

print("Minutes = ",Time.minute)

print("Seconds = ",Time.second)

**Day94: Django Templates:**

Templates:

==========

web pages/web apps ==> load/open with browser

web page ==> collection of web elements.

web elements ==> HTML elements

like:

Headings <h1>, <h2>, <h3>, <h4>, <h5> and <h6>

Paragraph <p>

<div> etc.

==> Templates can used to add all the html code instead to write in "views.py".

==> After writing, we should render that template to our app.

Creation of Template:

=====================

Select your App

create directory/folder with name "templates"

with templates folder:

we need to add sub folder with your app name.

within the sub-folder:

we need to crate the html file.

Syntax:

file-name.html

add the code into the html file

Render the html file into our Django app:

=========================================

in views.py:

===========

from Django.template import loader

def index(request):

item\_list = Item.objects.all()

template = loader.get\_template("html file location")

context = {

}

return HttpResponse(template.rende(context,request))

==> run the server

python manage.py runserver

==> copy the server url

and add your page to url navigation

ex: http://127.0.01:8000/item/