1.to start a new project type in terminal

**django-admin startproject project\_name**

2.to add a new application

**python manage.py startapp application\_name**

3.go to ***settings.py*** in the project folder

**->in the installed apps add the 'application\_name' or 'app\_name.apps.class\_name\_present'**

4.to view what we want add the content in ***views.py*** in the folder created

5.to print on the screen create a function -- variables is added if we want to pass a parameter

also add at top in views,py **=>**

**from django.http import HttpResponse**

**def func\_name(request,variables):**

**return HttpResponse("Content to be displayed")**

//the HttpResponse,aim of the response is to provide the client with the resource it requested

6.***create a urls.py*** file in the application folder

7.also in urls.py in same folder inside th urlspatterns add all the paths required

add this at top =>

**from django.urls import path**

**from . import views**

eg: **path("",application\_name.func\_name,name="classname")**

here when the urls is http://127.0.0.1:8000/

when http://127.0.0.1:8000/some\_name here path is some\_name

*we use here path("some\_name",application\_name.func\_name,name="classname"),*

*in case of variable use path("<str:name>",application\_name.func\_name,name="classname")*

*this is for parameter added to func in application*

8.in the main file folder project\_name **open urls.py**

add at top => **from django.urls import include**

then in urlpatterns add the path to the application

-> **path('applicaion\_name/',include("application\_name.urls"))**

9**. to run** the server type in terminal => **python manage.py runserver**

10.to add html template to the page we have to render the file

create a folder name template in that create a folder name with application\_name

create the html file we require then add the following in the views.py and urls.py

eg:def func\_name(request):

return render(request,"application\_name/html\_file\_name.html")

in case of variable =>

def func\_name(request,var):

return render(request,"application\_name/html\_file\_name.html",{

"ref\_var\_name":var

})

11.to use the var value in a heading use {{ ref\_var\_name }}

12. To add CSS to HTML file we need to create a folder named static and create a sub -folder with app\_name then create CSS file in it.

In the HTML file to add CSS

**{% load static %}**

<html lang="en">

<head>

<title> </title>

<link href=**"{% static ‘app\_name/css\_file\_name.css'** %}" rel="stylesheet">

</head>

<body>

</body>

</html>

13.to use logical stmts in html

If – else stmt: {% if cond %}

Value

{% else %}

Value

{% endif %}

While stmt : {% while cond %}

Value

{% endwhile %}

For stmt : {% for x in y %}

Value

{% endfor %}

14.if the HTML code is repeating in every page we can create a common html eg layout.html

Code be like

<html lang="en">

<head>

<title> .. </title>

</head>

<body>

…

{% block body %}

{% endblock %}

…

</body>

</html>

15. In the body block we can write codes for different pages we like

In the page we want the syntax to write continutuion is

{% extends "app\_name/ layout.html" %}

{% block body %}

<!—content to be printed -->

{% endblock %}

16. to link pages use <a href="{% url 'app\_name:file\_name' %}"> .. </a>

Or

<a href="{% url 'name\_in\_urls.py' %}"> .. </a>

17.to create a superuser for admin page we need to create database by database migratitions

To create a superuser we use python manage.py createsuperuser

python manage.py makemigrations is used to check for changes in database and update it in Django

in order to migrate we need to use => python manage.py migrate then we can create a superuser

then after the details go to localhost/admin and enter details

18. Django has it’s own ORM(object relational Mapper)=>we can use different databases and allows use access database

In models.py in the application folder we write what data needs to be saved to the database

We create a class which inherites from models.Model

And to get an char input we assign a var models.CharField(max\_length=..)

To give text field we use models.TextField() and

for datatime we use DateTimeField(default=timezone.now) => from django.utils import timezone

to create a user we need to add user module => from Django.contrib.auth.models import User

we use a models.ForeignKey(User, on\_delete=models.CASCADE)

(it implies when a user is deleted the data is also deleted)

Eg: class Post(models.Model):

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

content = models.TextField()

Author = models.ForeignKey(User, on\_delete=models.CASCADE)

Then we need to makemigrations then we can see the changes in migrations folder

To create a sql database we use python manage.py sqlmigrate app\_name change\_name

Eg blog\migrations\0001\_initial.py is when you make migration then change\_name is 0001

To create database through model => we create a shell to run interactively by using

Python manage.py shell

In the shell we can get the user present trough typing

>>> from django.contrib.auth.models import User

>>> User.objects.all()

Returns a <QuerySet [<User: ….>, <User: …>]>

>>> User.objects.first() for first and .last() for last we can even filter

>>> var = User.objects.filter(username=’…’)

To get the id we may use var.id or var.pk (pk=primary key)

To get the user trough id we use

>>> user= User.objects.get(id = id\_no)

In case to get all the class objects

>>> class\_name.objects.all()

To create a new obj => obj\_name=class\_name(values)

To save it to data => obj\_name.save()

To exit the shell type exit()

To set the posts use user.post\_set

To get all the posts 🡪 user.post\_set.all()

To create a post for it use use.post\_set.create(values)

19. in the views.py we need remove the dummy data and link the data and in the variable used to save data we write

Var\_container={

“var”= class\_name.objects.all()

}

20.To get the models in admin page we need to add in admin.py

from .models import class\_name

admin.site.register(class\_name)