DJANGO

while running
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
to create a superuser - python manage.py createsuperuser
to create project -   django-admin startproject project_name
to run local server-   python manage.py runserver
to create app - python manage.py startapp app_name
to migrate - python manage.py migrate
to load static - python manage.py collectstatic

FIRST - views,templates

step one

after u create the first app u have to add the app to setting.py to let the programme know that u have create an app
in setting.py
INSTALLED APP = [
'app_name'
]
 add it here
in app veiws.py add
from django.http import HttpResponse
def index(request):
    return HttpResponse('hello world')
then in url.py in project add
from django.conf.urls import url
urlpatterns = [
    url(r'^$',views.index, name = 'index'),

this going to print hello world on the web page

step two

earlier step is lengthy so there is shorter step by using include() function by creating
in project urls.py file
from django.conf.urls import include
url(r'^app_name/',include('app_name.urls'))
then create urls.py file in app_name then repeate step one in urls.py of app

step three

adding a template directory
create a dir template in project folder
to pass the file path use
import os
in setting.py
there is BASE_DIR   which is the location of base dir we have use this path as refernce for all location as while transfering project from one comp to another the actual path may change but this is python

generated path so it adjust atomatically
now to set temp location we use
  TEMPLATE_DIR = os.path.join(BASE_DIR , “template_folder_name”)
now add this location to template so django can acess it by add to
 under TEMPLATES in Setting.py there is
 templates = [
 ‘DIRS’: []
]
add temp loc to this
‘DIRS’ : [TEMPLATE_DIR]

step four

now u can create html file in template dir
and to use it
from django.shortcuts import render
def index(request):
    var = {'insert_me':"im from veiws.py"}
    return render(request,'index.html',context = var)
insert_me can be define in html file and it use as a variable to pass info

step five

adding static image file from a folder and insert it in html code
first create a folder name static
then add its directory address just like we did for template folder
STATIC_DIR = os.path.join(BASE_DIR,”static_folder_name”)
now add this loc to static
 STATICFILES_DIR = [
STATIC_DIR,
]
now u can create images folder in static folder and put ur images there
in Html code add at the top after the doctypehtml line
{% load staticfiles %}
then in html code where u want to add the image add the follwing -
<img src = {% static “images/pic_name.jpg” %} >

SECOND -models, admin

Step one

in model.py file of app_name
u can create classes
syntax as follows
class model_name(models.Model) :
        model_item_name = models.CharField()
now after that run migration command
first
python manage.py makemigrations app_name
python manage.py migrate
then in admin.py of app register your model
from app_name.models import model1,model2,blabla
admin .site.register.(model1)
after that create the first user by using command

pyhton manage.py createsuperuser

Step two

---

pip install Faker
Check later
just for demo not useful

Step Three

import models to views.py of app
from app_name.models import model1,model2,blabla
then u can pass the model item
like define a varible a
a = model_item_name.object.order_by(blabla)
note - .order_by is used to
now pass it into dictonary
var = {key : a}
in render request context=var
in html code eg

```
<div class="collegelist">
    {% if a %}
    <table>
        <thead>
        <th>Rank</th>
        <th>Name</th>
        <th>Location</th>
        </thead>
        {% for item in a %}
    <tr>
        <td> {{item.rank}} </td>
        <td> {{item.name}} </td>
        <td> {{item.location}} </td>
    </tr>
    {% endfor %}
    </table>
    {% else %}
    <p>NO ACCESS RECORD FOUND</p>
    {% endif %}
</div>
```

THIRD - User Input

Step one

creating Form
first create form.py file in app
then edit the file
from django import forms eg -

```
class FormName(forms.Form):
    name = forms.CharField()
    email = forms.EmailField()
    text = forms.CharField(widget=forms.Textarea)
```
then edit veiws.py file add a similar function
```
def form(request):
    form1 = forms.FormName()
    v = {'form' : form1}
    return render(request ,'form.html', context= v)
```
now in urls.py add a new url entry for function form
eg - url(r'^form/',views.form_name,name = 'form')
now in form.html
```
<div class = "container">
    <form method="POST">
        {{form.as_p}}
        {% csrf_token %}
        <input type="submit" class="btn btn-primary" value="submit">
    </form>
</div>
```
now u can use form but to use the data input this will print it in console
edit the view.py files form_name function to
```
def form_name(request):
    form1 = forms.FormName()
    if request.method == 'POST':
        form1 = forms.FormName(request.POST)
        if form1.is_valid():
            print("NAME : "+form1.cleaned_data['name'])
            print("EMAIL : " + form1.cleaned_data['email'])
            print("TEXT : " + form1.cleaned_data['text'])
    return render(request ,'form.html', context = {'form' : form1})
```

Step two
validation
to detect bot we add a hidden field human cant see them but will fill it so we can detect the bot
from django.core import validators
in form.py file in class formname add a new line
botcatcher = forms.CharField(required=False , widget=forms.HiddenInput , validators = [validators.MaxLengthValidator(0)])
if we want user enter there email twice and want to make sure that they are both same them use this funtion
```
verify_email = form.EmailField(label='Enter your email again')
text = forms.Chafield(widget = form.Textarea)
def clean(self):
        all_clean_data = super().clean()
        email = all_clean_data['email']
        vmail = all_clean_data['verify_email']
        if email != vmail:
                raise forms.ValidationError("Make Sure Emails Match")
```

Step three
saving input data to model
first create a new model in models.py
```
class User(models.Model) :
    first_name = models.CharField(max_length=20)
```

    last_name = models.CharField(max_length=20)
    email = models.EmailField(max_length=50,unique=True)
then register the model in admin.py

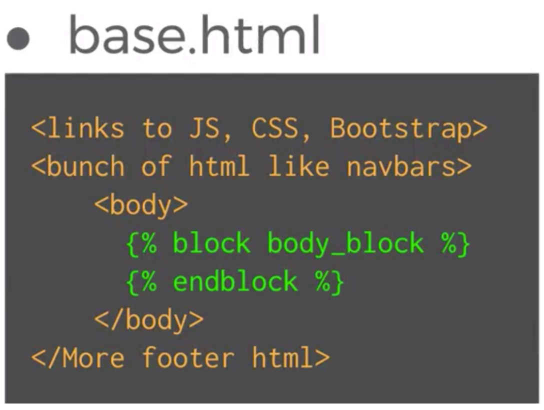
FORTH - URL Mapping

Step one - relative url

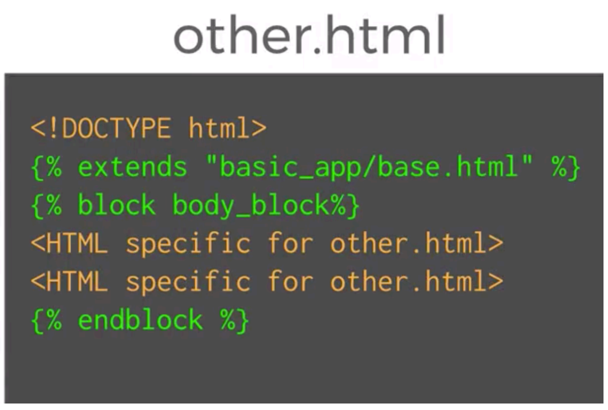
it means we can call other webpage from one webpage
first we have to include template tagging in url.py file of app
app_name = 'basic_app'
then we have to add url link in the html as
<a herf="{% url 'basic_app:other' %}">To go to other page click here </a>
eg for index {% url 'index'%}. —— for index page we only have type index
eg for admin {% url 'admin:index'%}

Step two - template inheritance

we create a basic template which contain all the things we want other pages to show also like navigation bar
it all HTML code by the way but with django power to do so we use two tags
{%block body_block%}
{%endblock%}
in template html file all the code outside these tags are inherited to the other page and we leave these tags empty
in inherited html we first inherit the template using the tag
{%extend "location of template html in template dir"%}
then include body block and end block tag and add all things in side these tags
eg if we include code of nav bar outside these code in template then some text in inherited html then inherited html will have both nav bar code from template and text from its own code


- base.html

```
<links to JS, CSS, Bootstrap>
<bunch of html like navbars>
    <body>
      {% block body_block %}
      {% endblock %}
    </body>
</More footer html>
```


other.html

```
<!DOCTYPE html>
{% extends "basic_app/base.html" %}
{% block body_block%}
<HTML specific for other.html>
<HTML specific for other.html>
{% endblock %}
```

Step three - template filters

general form -   {{value | filter:"parameter"}}
to include a dictonary from data base in to the html we first include the dict in html by including it in render functions parameter in django them in html just use {{key}} to include the dict and it will show the value. Now we can apply filters on it too eg to convert the text to upper we {{text | upper}} or to add {{number | add:24}} this will add number there are many filter on django doc u have explore how they work but i have shown two of them and we create custom too.

FIVE - FORMS

step one - password

we just going to encrypt the saved password instead of saving them as plain text
pip install bcrypt
pip install django[argon2]
these libraries are need for password encryption
now in setting.py at the end add the following -
```
PASSWORD_HASHERS = [
                    'django.contrib.auth.hashers.Argon2PasswordHasher',
                    'django.contrib.auth.hashers.BCryptSHA256PasswordHasher',
                    'django.contrib.auth.hashers.BCryptPasswordHasher',
                    'django.contrib.auth.hashers.PBKDF2PasswordHasher',
                    'django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher'
                    ]
```
then in password validators we can set value for like minimun length of password and all
now we can segrigate the user media and root media by creating media folder in which all the media uploaded by user will be saved
then create MEDIA_DIR = os.path.join(BASE_DIR,'media')
then at the bottom add ——
```
MEDIA_ROOT = MEDIA_DIR
   MEDIA_URL = "/media/"
```