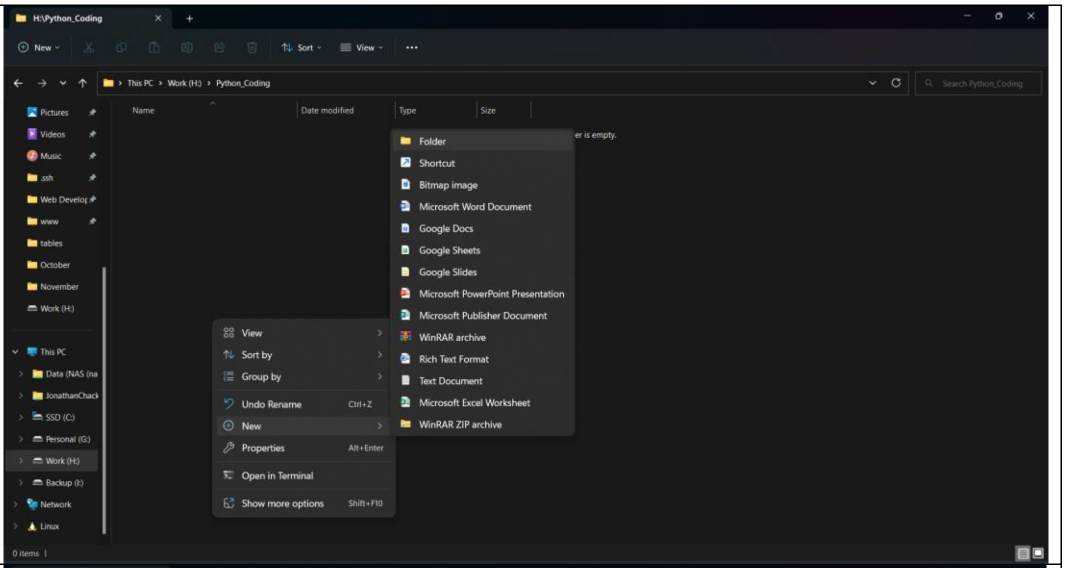
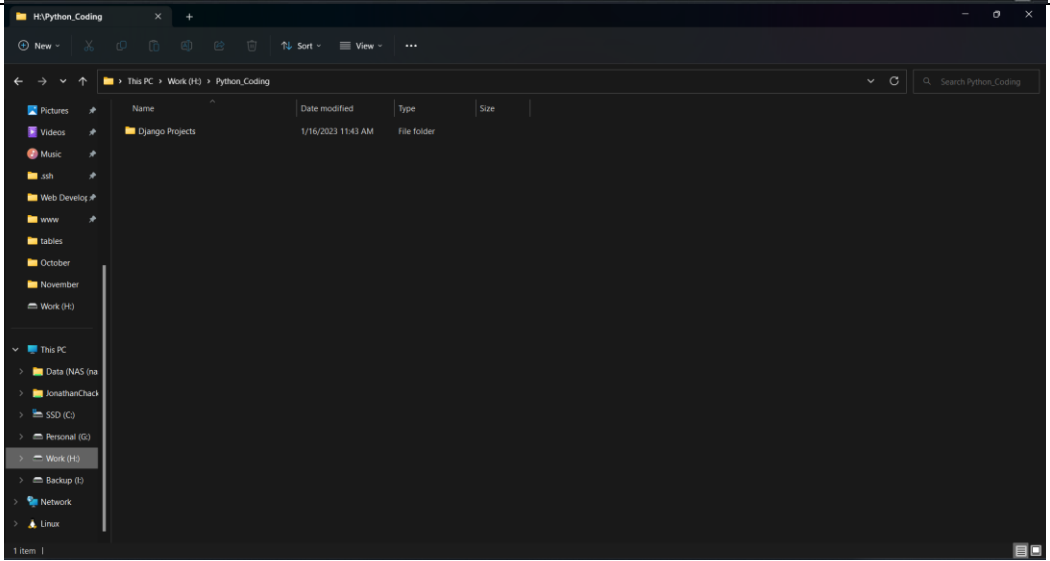
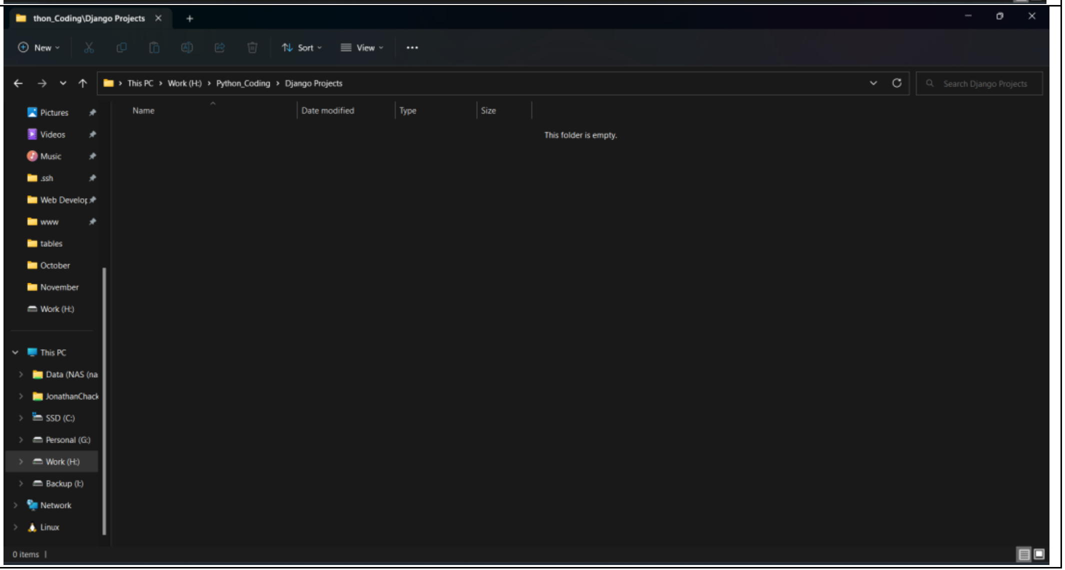
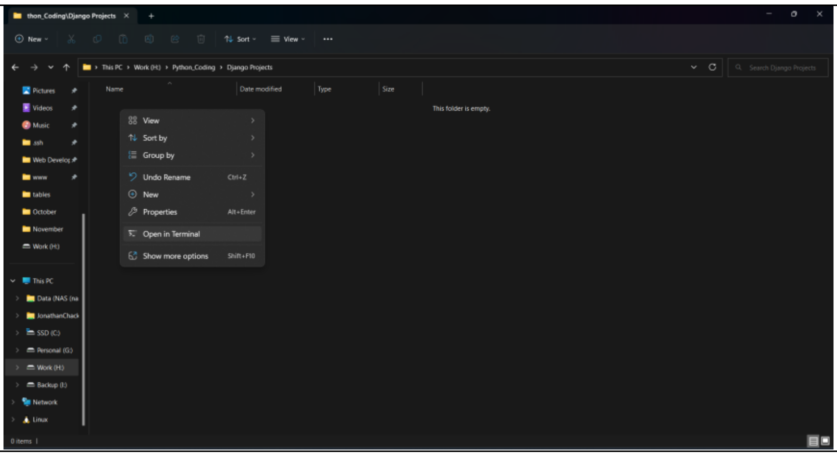
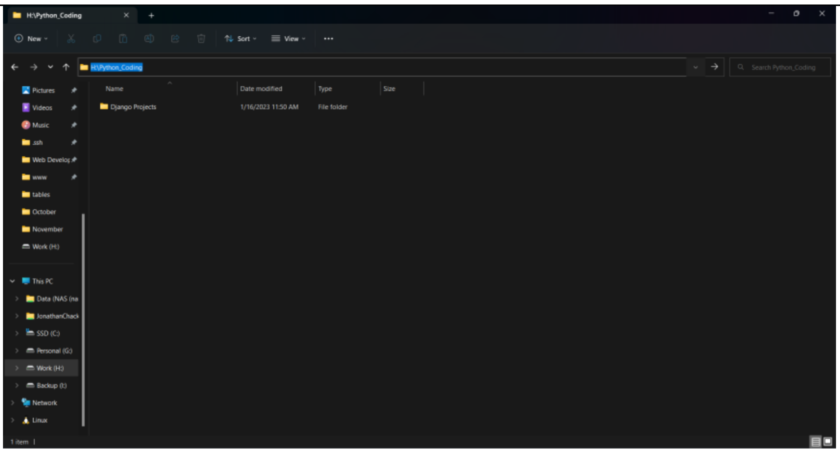
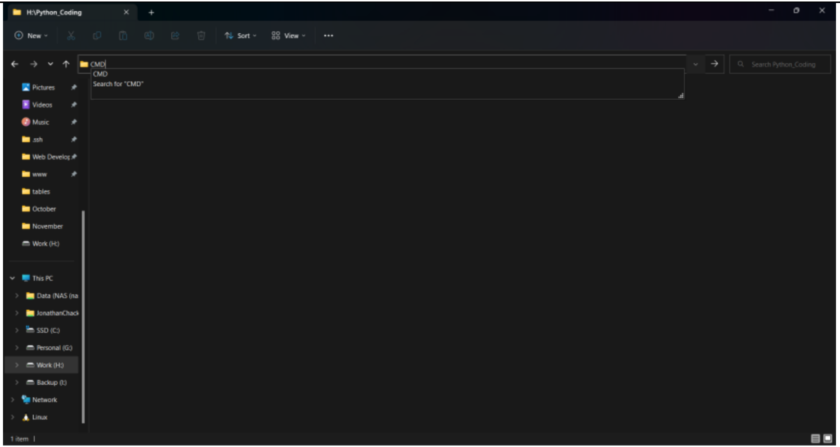
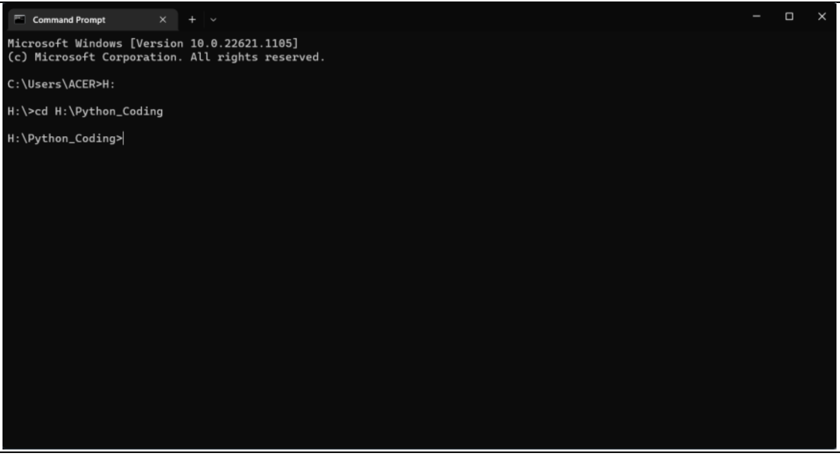
Contents & Index

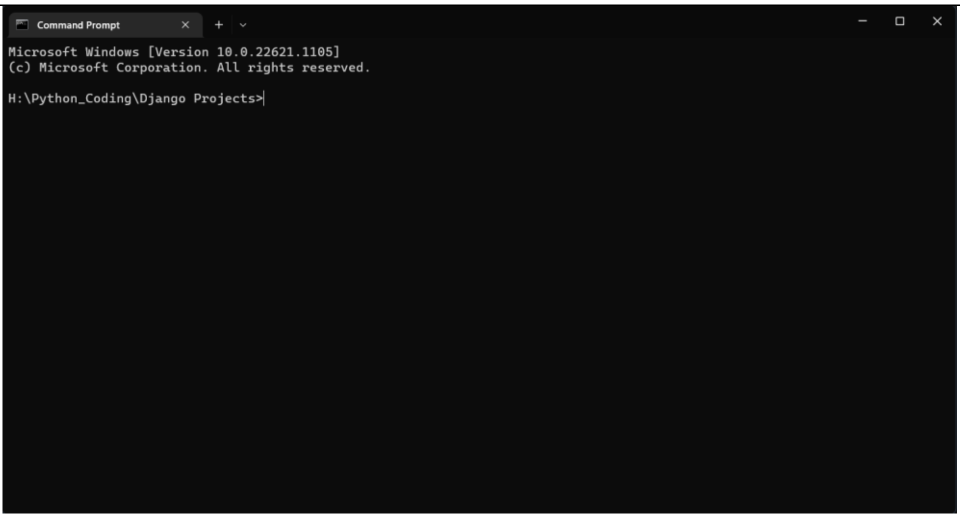
Getting Started

Create a New Folder for Django Projects:	 H:\Python_Coding New Sort View This PC › Work (H:) › Python_Coding Search Python_Coding Name Date modified Type Size er is empty. View Sort by Group by Undo Rename Ctrl+Z New Properties Alt+Enter Open in Terminal Show more options Shift+F10 Folder Shortcut Bitmap image Microsoft Word Document Google Docs Google Sheets Google Slides Microsoft PowerPoint Presentation Microsoft Publisher Document WinRAR archive Rich Text Format Text Document Microsoft Excel Worksheet WinRAR ZIP archive 0 items
Select the New Folder:	 H:\Python_Coding This PC › Work (H:) › Python_Coding Name Date modified Type Size Django Projects 1/16/2023 11:43 AM File folder 1 item
Open the New Folder:	 thon_Coding\Django Projects This PC › Work (H:) › Python_Coding › Django Projects Search Django Projects Name Date modified Type Size This folder is empty. 0 items

<table>
<tr><td>Then Right Click & Open in Terminal</td><td>
</td></tr>
<tr><td colspan="2">OR</td></tr>
<tr><td>Click on the Path</td><td>
</td></tr>
<tr><td>and Type CMD</td><td>
</td></tr>
<tr><td>OR In Case That Doesn’t Work Copy Your Path and CD to that Directory in CMD</td><td>

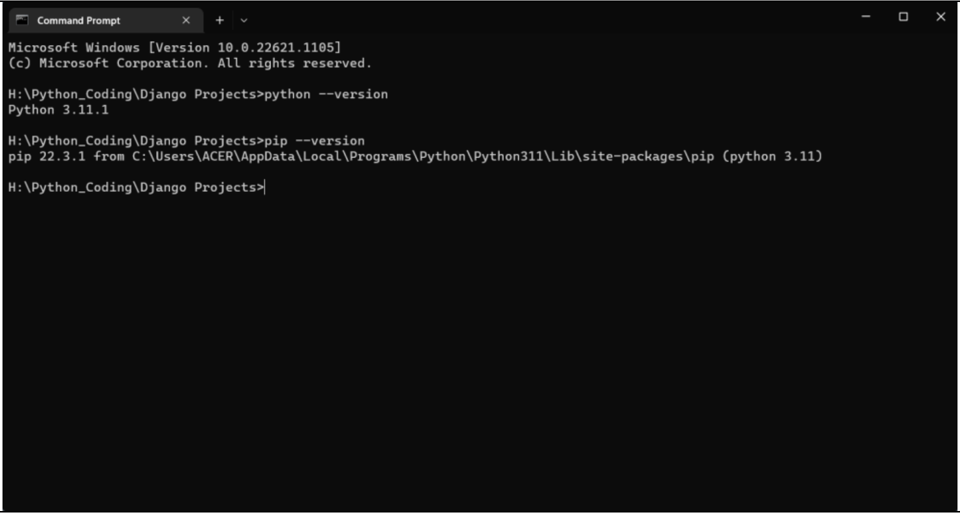
```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ACER>H:

H:\>cd H:\Python_Coding

H:\Python_Coding>
```
</td></tr>
</table>

THEN We Have the CMD Open
in the Path of the Folder we Need to Work in



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

H:\Python_Coding\Django Projects>
```

Type:
`python --version`
`pip --version`

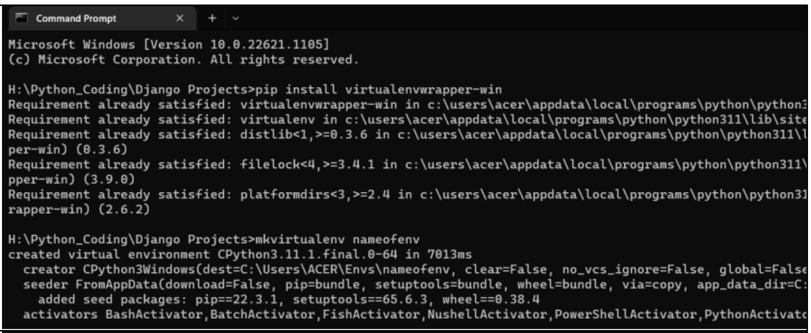


```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

H:\Python_Coding\Django Projects>python --version
Python 3.11.1

H:\Python_Coding\Django Projects>pip --version
pip 22.3.1 from C:\Users\ACER\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

H:\Python_Coding\Django Projects>
```

In Case You Don’t Get the Desired Output Go to https://www.python.org/downloads/ Download the Latest Version and During Installation Don’t Forget to **Add Python to Path** *&or* **Add Python to Environment Variables**.

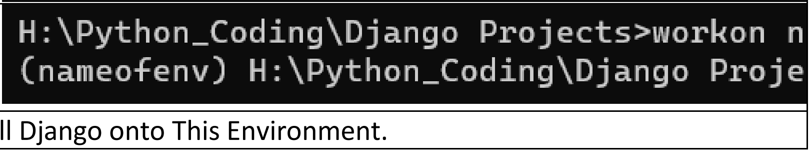
Creating an Environment

```
pip install virtualenvwrapper-win
mkvirtualenv [nameofenv]
```



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

H:\Python_Coding\Django Projects>pip install virtualenvwrapper-win
Requirement already satisfied: virtualenvwrapper-win in c:\users\acer\appdata\local\programs\python\python3
Requirement already satisfied: virtualenv in c:\users\acer\appdata\local\programs\python\python311\lib\site
Requirement already satisfied: distlib<1,>=0.3.6 in c:\users\acer\appdata\local\programs\python\python311\l
per-win) (0.3.6)
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\acer\appdata\local\programs\python\python311\
pper-win) (3.9.0)
Requirement already satisfied: platformdirs<3,>=2.4 in c:\users\acer\appdata\local\programs\python\python31
rapper-win) (2.6.2)

H:\Python_Coding\Django Projects>mkvirtualenv nameofenv
created virtual environment CPython3.11.1.final.0-64 in 7013ms
  creator CPython3Windows(dest=C:\Users\ACER\Envs\nameofenv, clear=False, no_vcs_ignore=False, global=False
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:
    added seed packages: pip==22.3.1, setuptools==65.6.3, wheel==0.38.4
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivato
```

<u>To Select the Environment:</u>
`workon [nameofenv]`

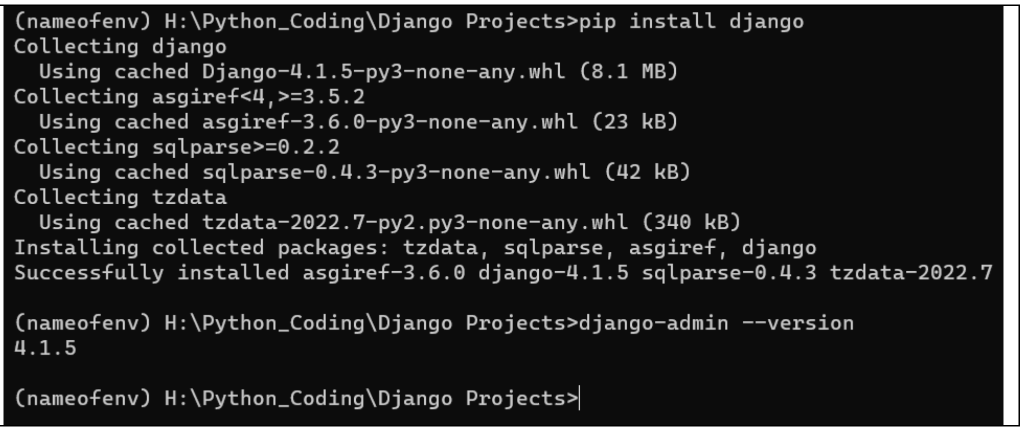


```
H:\Python_Coding\Django Projects>workon n
(nameofenv) H:\Python_Coding\Django Proje
```

Now We Install Django onto This Environment.

Downloading Django

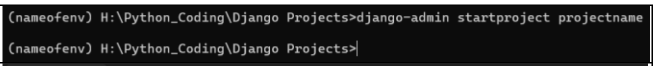
```
pip install django
django-admin -version
```



```
(nameofenv) H:\Python_Coding\Django Projects>pip install django
Collecting django
  Using cached Django-4.1.5-py3-none-any.whl (8.1 MB)
Collecting asgiref<4,>=3.5.2
  Using cached asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.3-py3-none-any.whl (42 kB)
Collecting tzdata
  Using cached tzdata-2022.7-py2.py3-none-any.whl (340 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.6.0 django-4.1.5 sqlparse-0.4.3 tzdata-2022.7

(nameofenv) H:\Python_Coding\Django Projects>django-admin --version
4.1.5

(nameofenv) H:\Python_Coding\Django Projects>
```

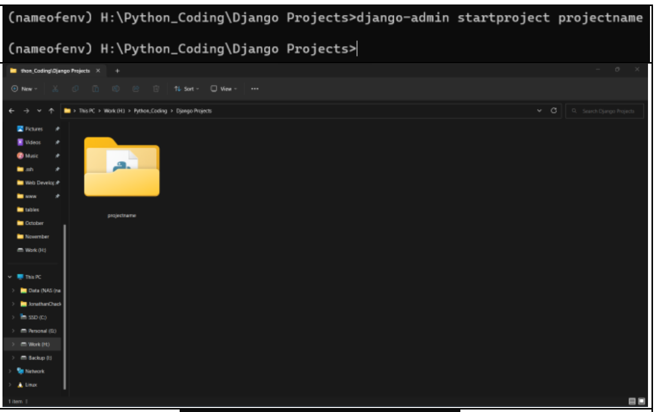
Start a Django Project in the Current Directory

```
django-admin startproject [ProjectName]
```

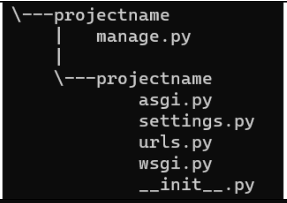


```
(nameofenv) H:\Python_Coding\Django Projects>django-admin startproject projectname

(nameofenv) H:\Python_Coding\Django Projects>
```

This Folder will be Created -



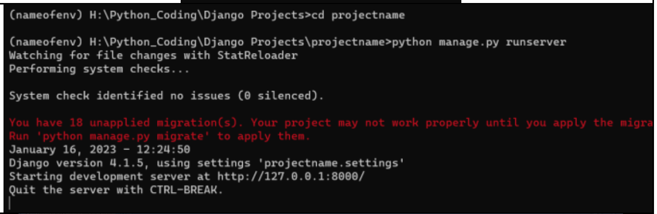
The Files Created in the Following Folder is =



```
\---projectname
    |   manage.py
    |
    \---projectname
            asgi.py
            settings.py
            urls.py
            wsgi.py
            __init__.py
```

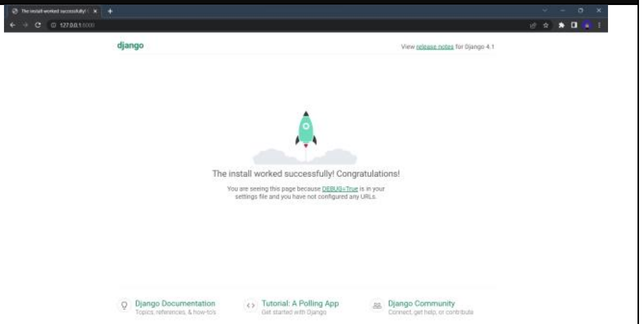
<u>Run the Server to Check if the Project Files are not Corrupted and the Django Project is Working:</u>

```
cd [ProjectName]
python manage.py runserver
```



```
(nameofenv) H:\Python_Coding\Django Projects>cd projectname

(nameofenv) H:\Python_Coding\Django Projects\projectname>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migra
Run 'python manage.py migrate' to apply them.
January 16, 2023 - 12:24:50
Django version 4.1.5, using settings 'projectname.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Go to Link http://127.0.0.1:8000/ and If the Below Output is What You Get It Means Django Project is Working.


The install worked successfully! C
127.0.0.1:8000
django
View release notes for Django 4.1

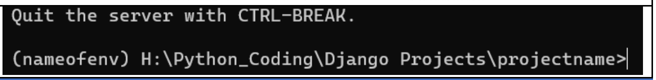
The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

to End Server

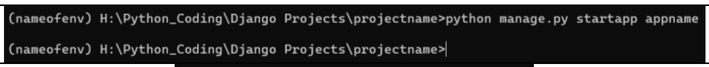
```
Ctrl + C
```



```
Quit the server with CTRL-BREAK.

(nameofenv) H:\Python_Coding\Django Projects\projectname>
```

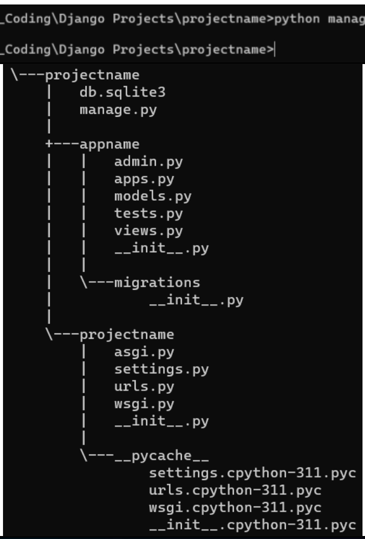
Start an Django App

`python manage.py startapp [AppName]`



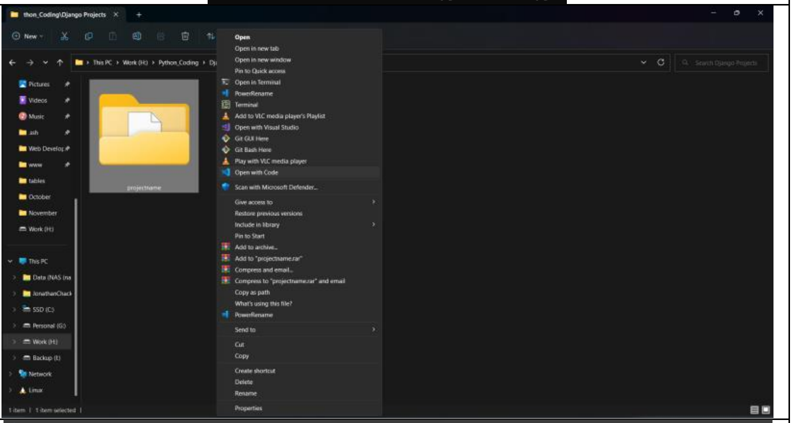
```
(nameofenv) H:\Python_Coding\Django Projects\projectname>python manage.py startapp appname

(nameofenv) H:\Python_Coding\Django Projects\projectname>
```

Then The Files in the Project File will be Updated =

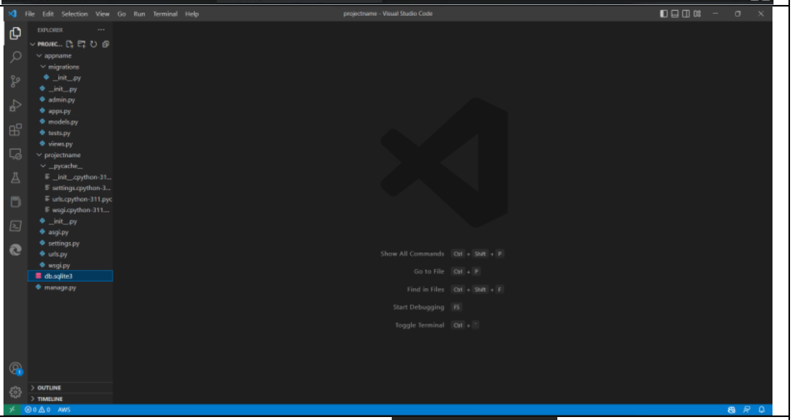


```
\---projectname
    |   db.sqlite3
    |   manage.py
    |
    +---appname
    |   |   admin.py
    |   |   apps.py
    |   |   models.py
    |   |   tests.py
    |   |   views.py
    |   |   __init__.py
    |   |
    |   \---migrations
    |           __init__.py
    |
    \---projectname
        |   asgi.py
        |   settings.py
        |   urls.py
        |   wsgi.py
        |   __init__.py
        |
        \---__pycache__
                settings.cpython-311.pyc
                urls.cpython-311.pyc
                wsgi.cpython-311.pyc
                __init__.cpython-311.pyc
```

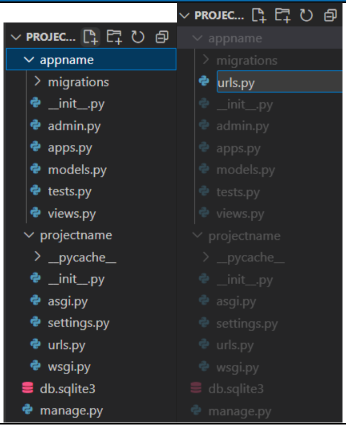
Open the Base Directory Project Folder with your Editor



in my Case I'm Using VS-Code Editor



Add a New File Called *urls.py* to the ***[AppName] Folder***



In the *projectname/appname/views.py* Create a Function with the Page ID:

FROM

```python
from django.shortcuts import render

# Create your views here.
```

TO

```python
from django.shortcuts import render
from django.http import HttpResponse
from django.template import loader
# Create your views here.
def pageID(request):
    return HttpResponse("<p>Basic HTML Codes Can Go Here</p>")
```

In the *projectname/appname/urls.py*

FROM

TO

```python
from django.urls import path
from . import views
urlpatterns = [
    #This is Where You add the Path Code
    path('',views.pageID, name = "Page Name" ), #Path for HomePage would be
Empty
]
```

In the *projectname/projectname/urls.py*

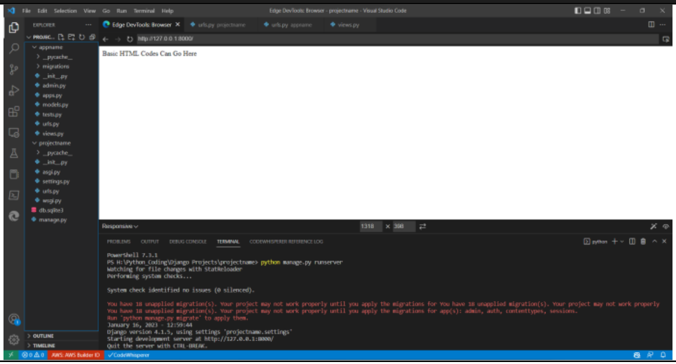
FROM

```python
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

TO

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('appname.urls'))
]
```

Then RUNSERVER



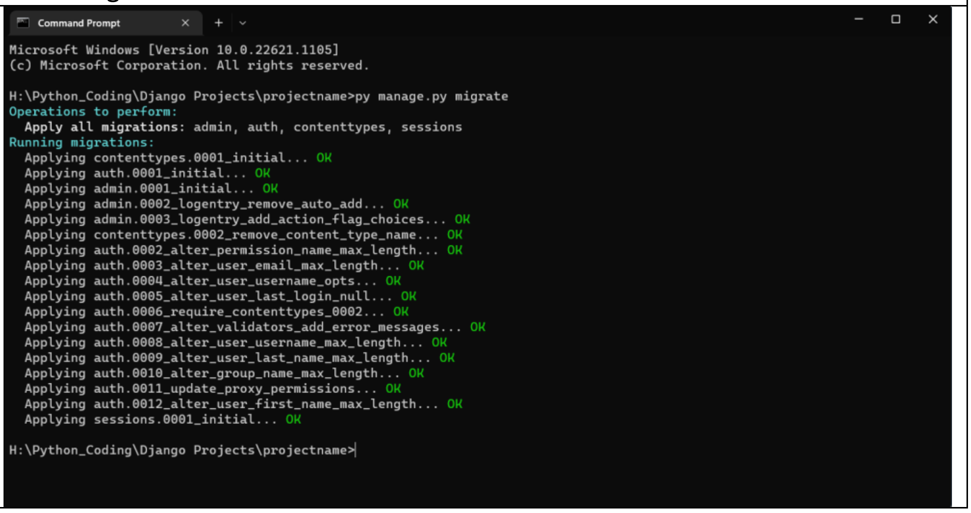
Basic HTML Codes Can Go Here

```
PowerShell 7.3.1
PS H:\Python_Coding\Django Projects\projectname> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for You have 18 unapplied migration(s). Your project may not work properly
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
January 16, 2023 - 12:59:44
Django version 4.1.5, using settings 'projectname.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Migrate

In the Project Folder Run a Command through CMD

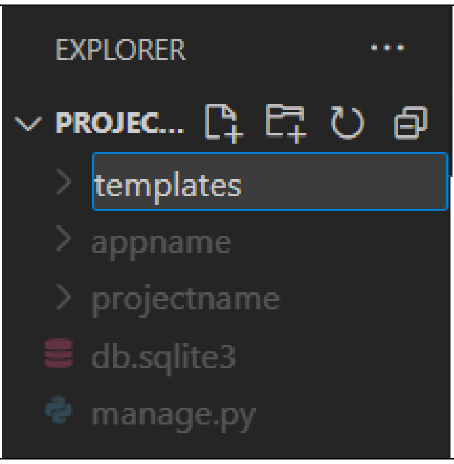
```
python manage.py migrate
```



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

H:\Python_Coding\Django Projects\projectname>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

H:\Python_Coding\Django Projects\projectname>
```

Creating Templates Folder

In the Base Directory Create a Folder Called ***templates***



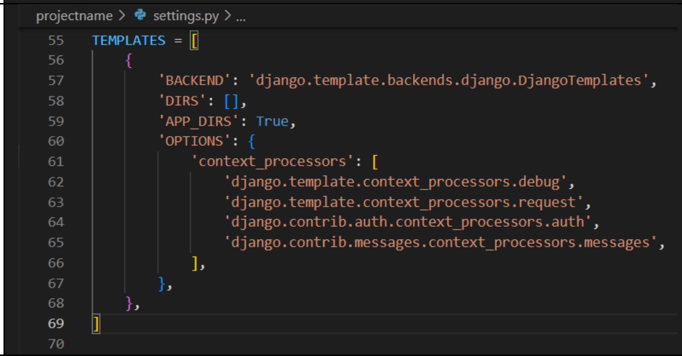
In the *projectname/projectname/settings.py* Edit the Following Code

FIRST (Import the Important Libs)

```
import os
```

Then Edit the Settings Required

FROM

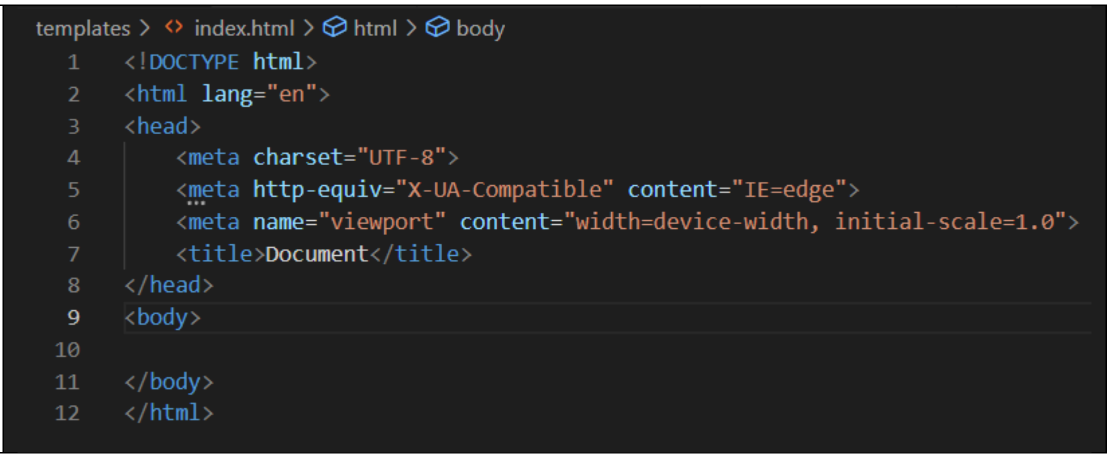


```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

TO

```
'DIRS': [os.path.join(BASE_DIR, 'templates')],
```

Then Create a File in the ***templates*** Folder Called *index.html* fill it with Default HTML Code for Now:

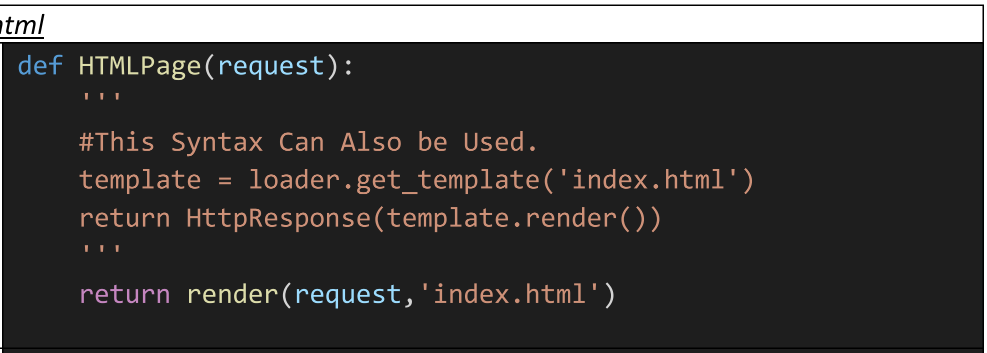


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Print the Contents of a HTML File

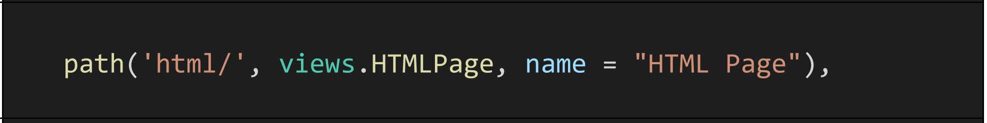
Lets Print The Contents of *index.html*

First Let's go to *projectname/appname/views.py* and Create a Function to Call the HTML File



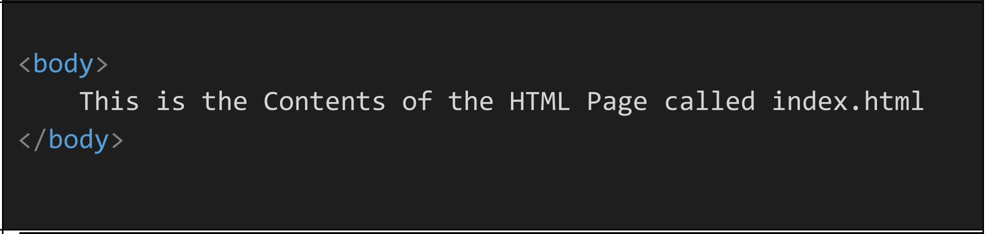
```
def HTMLPage(request):
    '''
    #This Syntax Can Also be Used.
    template = loader.get_template('index.html')
    return HttpResponse(template.render())
    '''
    return render(request,'index.html')
```

and then Call the Function to *projectname/appname/urls.py*



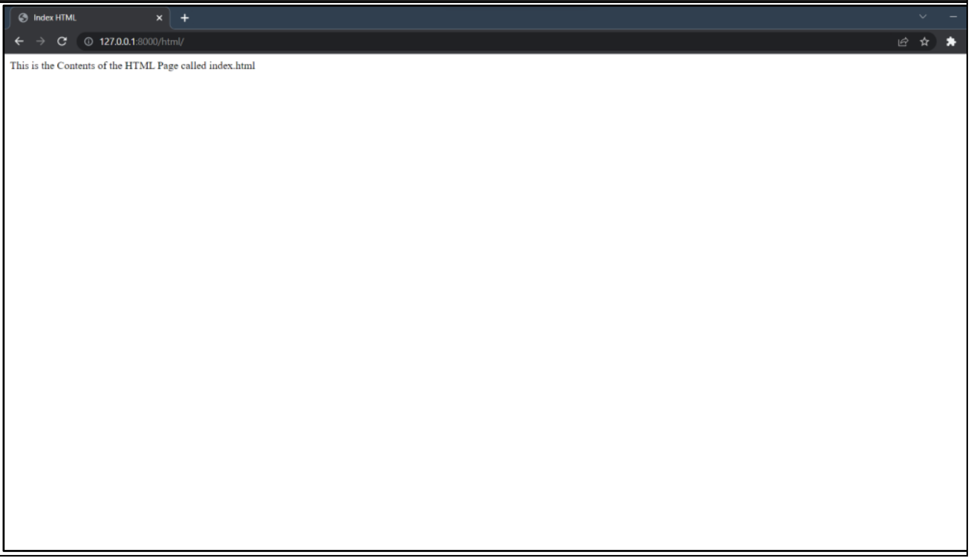
```
   path('html/', views.HTMLPage, name = "HTML Page"),
```

add Something to the HTML Page *projectname/templates/index.html*



```
<body>
    This is the Contents of the HTML Page called index.html
</body>
```

Run the Server & Go To http://127.0.0.1:8000/html/



This is the Contents of the HTML Page called index.html

Data Communication

Call Data to the Template

In the File *projectname/appname/views.py*

FROM

```
def HTMLPage(request):
    '''
    #This Syntax Can Also be Used.
    template = loader.get_template('index.html')
    return HttpResponse(template.render())
    '''
    return render(request,'index.html')
```

TO

```
def HTMLPage(request):
    return render(request,'index.html', {'KEY':'VALUE'})
```

In the File *projectname/templates/index.html*

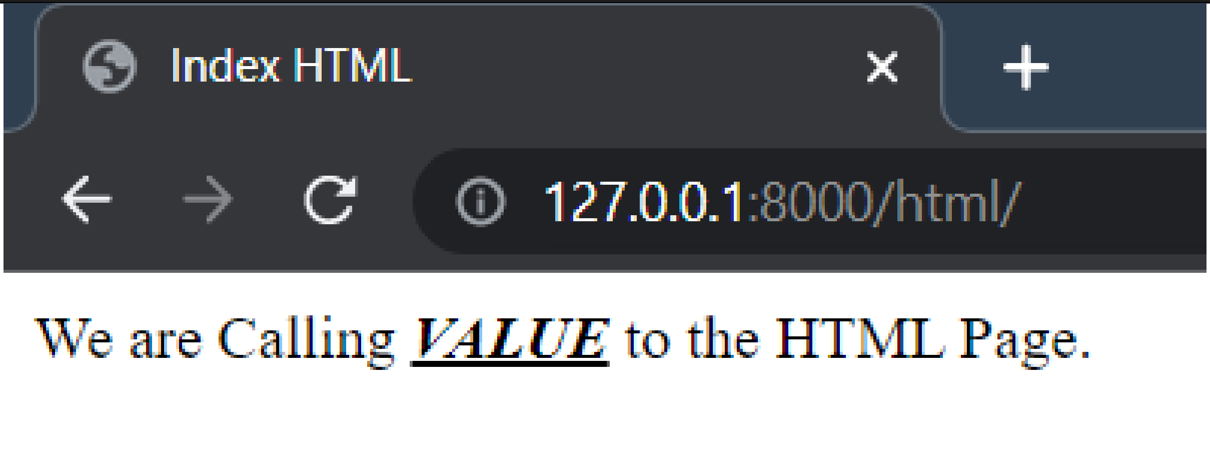
FROM

```
<body>
    This is the Contents of the HTML Page called index.html
</body>
```

TO

```
<body>
    We are Calling <b><i><u>{{KEY}}</u></i></b> to the HTML Page.
</body>
```

Result Output



We are Calling ***VALUE*** to the HTML Page.

Send List Data to the Template

In the File *projectname/appname/views.py*

FROM

```
    return render(request,'index.html', {'KEY': 'VALUE'})
```

TO

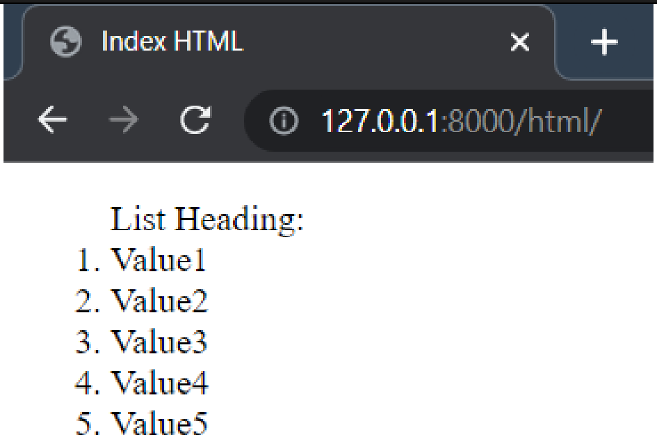
```
    List = ["Value1", "Value2", "Value3", "Value4", "Value5"]
    return render(request,'index.html', {'KEY': List})
```

In the File *projectname/templates/index.html*

FROM

```
We are Calling <b><i><u>{{KEY}}</u></i></b> to the HTML Page.
```

TO	`<ol type="1">`
	`        <lh>List Heading:</lh>`
	`              {% for x in KEY %}`
	`                    <li>{{x}}</li>`
	`                    {% endfor %}`
	`</ol>`

Result Output	 Index HTML 127.0.0.1:8000/html/ List Heading: 1. Value1 2. Value2 3. Value3 4. Value4 5. Value5

Creating Themes

In *projectname/appname/models.py* Create a Class Containing a Data Template	

FROM	`from django.db import models`
	
	`# Create your models here.`

TO	`from django.db import models`
	
	`# Create your models here.`
	
	`class theme:`
	`    dataTypeNameForStr:str`
	`    dataTypeNameForInt:int`

In the File *projectname/appname/views.py*	
FIRST Import the Class from the *models.py* File	`from appname.models import theme`

FROM	`    List = ["Value1", "Value2", "Value3", "Value4", "Value5"]`
	`    return render(request,'index.html', {'KEY': List})`

TO
(Import the Class as an Object)
	`    Object1 = theme()`
	`    Object1.dataTypeNameForStr = "Jonathan"`
	`    Object1.dataTypeNameForInt = 1805`
	`    Object2 = theme()`
	`    Object2.dataTypeNameForStr = "Chacko"`
	`    Object2.dataTypeNameForInt = 2002`
	`    MasterListObject = [Object1, Object2]`
	`    return render(request,'index.html',{'KEY':MasterListObject})`

In the File *projectname/templates/index.html*

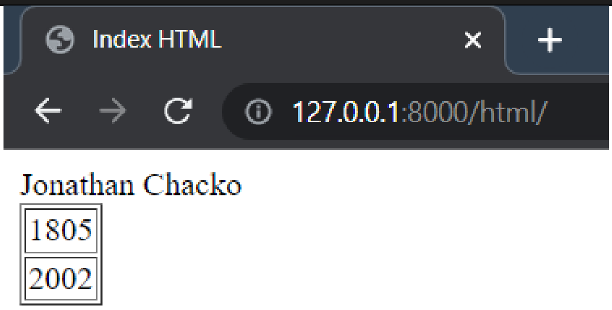
FROM

```
<ol type="1">
      <lh>List Heading:</lh>
           {% for x in KEY %}
                <li>{{x}}</li>
                {% endfor %}
</ol>
```

TO

```
<table border = '1'>
    {% for x in KEY %}
    <tr>
        {{x.dataTypeNameForStr}}
        <td>
            {{x.dataTypeNameForInt}}
        </td>
    </tr>
    {% endfor %}
</table>
```

Result Output



Jonathan Chacko

1805
2002

Form Input

In the *projectname/appname/views.py*

FROM

```
     Object1 = theme()
     Object1.dataTypeNameForStr = "Jonathan"
     Object1.dataTypeNameForInt = 1805
     Object2 = theme()
     Object2.dataTypeNameForStr = "Chacko"
     Object2.dataTypeNameForInt = 2002
     MasterListObject = [Object1, Object2]
     return render(request,'index.html', {'KEY': MasterListObject})
```

TO

```
     if request.method == 'POST':
         num1 = request.POST['num1'] # input1 = request.GET['num1']
         num2 = request.POST['num2'] # input2 = request.GET['num2']
         if 'add' in request.POST:
             result = int(num1) + int(num2)
             return render(request,'index.html',{'KEY':result})
     return render(request,'index.html')
```

In the File *projectname/templates/index.html*

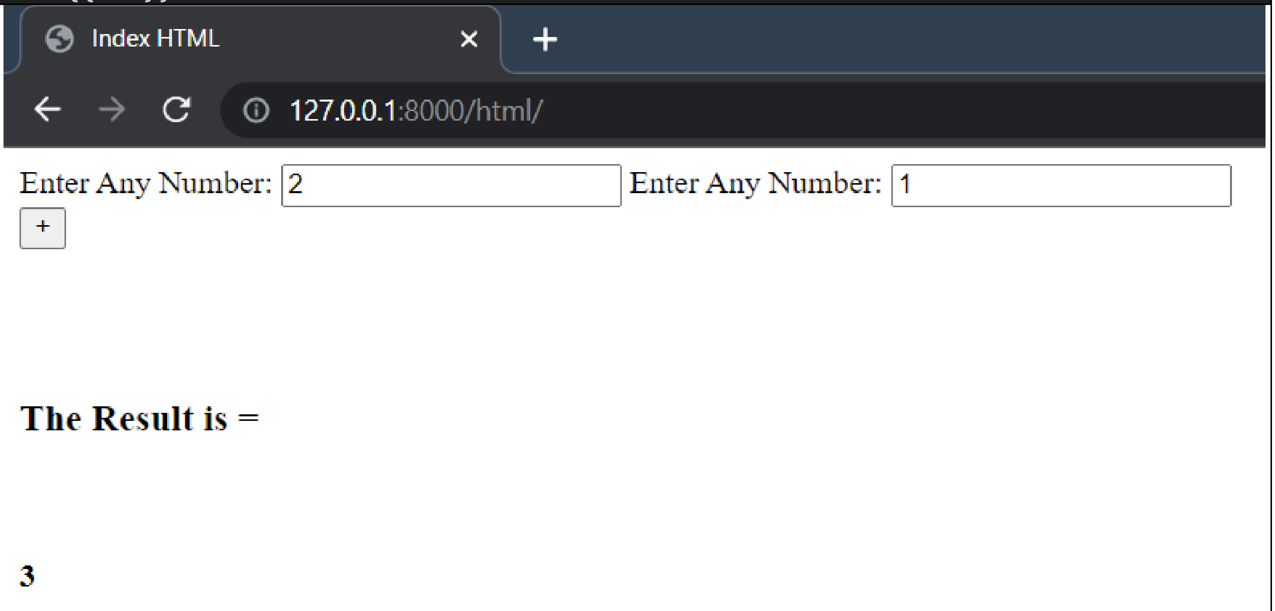
FROM

```
<table border = '1'>
    {% for x in KEY %}
    <tr>
        {{x.dataTypeNameForStr}}
        <td>
            {{x.dataTypeNameForInt}}
        </td>
    </tr>
    {% endfor %}
</table>
```

TO

```
<form method="POST">
    {% csrf_token %}
    <label for="num1">Enter Any Number: </label>
        <input type="text" name="num1"/>
    <label for="num2">Enter Any Number: </label>
        <input type="text" name="num2"/>
    <br/>
    <button type="submit" name="add">+</button>
</form>
<br><br><br>
<h3>The Result is = </h3><br>
<h4>{{KEY}}</h4>
```

Result Output



Enter Any Number: 2 Enter Any Number: 1

+

The Result is =

3

Django's login form is returned using the POST method, in which the browser bundles up the form data, encodes it for transmission, sends it to the server, and then receives back its response.

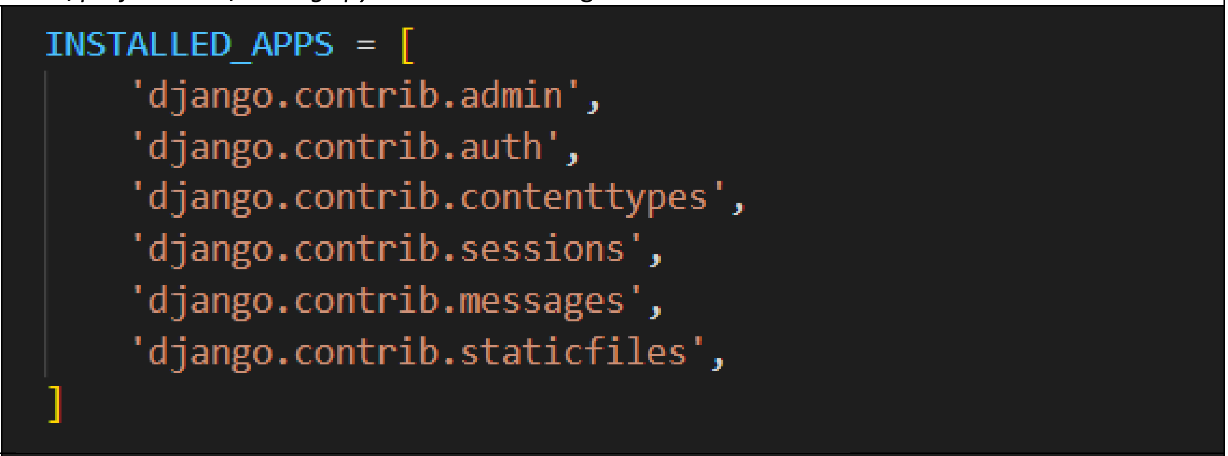
GET, by contrast, bundles the submitted data into a string, and uses this to compose a URL.

SQL

Initialization

In the *projectname/projectname/settings.py* Edit the Following Code:

FROM



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

TO

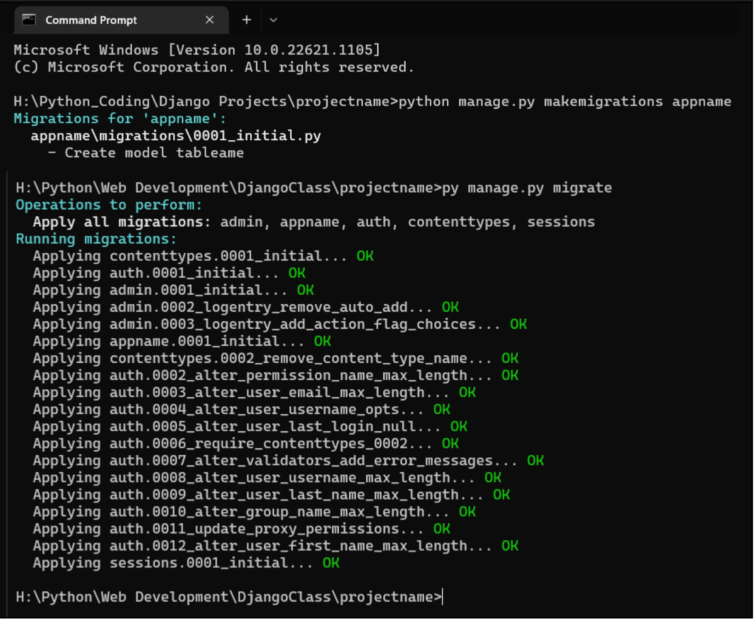
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'appname',
]
```

Migrations of Models

Create a Class Containing a Model in In *projectname/appname/models.py*

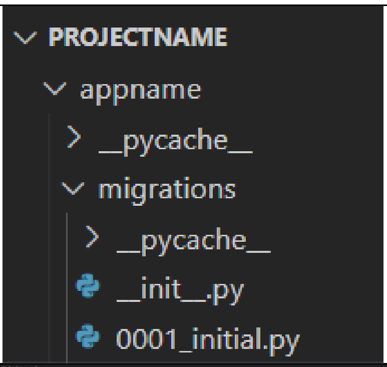
```
class tablename(models.Model):
    dtstr1 = models.CharField(max_length=255)
    dtstr2 = models.CharField(max_length=255)
    #This is to Sync SQLite3
```

```
py manage.py makemigrations appname
py manage.py migrate
```



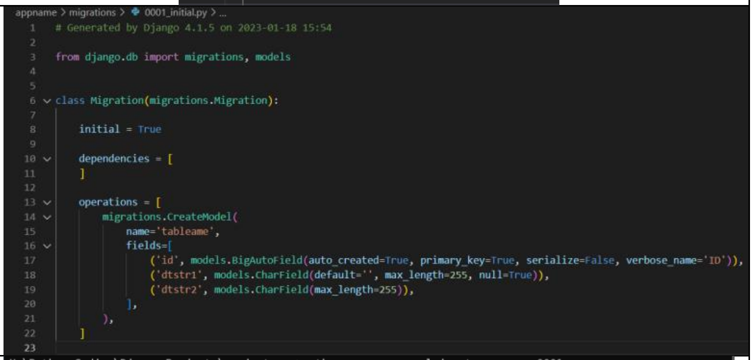
```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

H:\Python_Coding\Django Projects\projectname>python manage.py makemigrations appname
Migrations for 'appname':
  appname\migrations\0001_initial.py
    - Create model tableame

H:\Python\Web Development\DjangoClass\projectname>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, appname, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying appname.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

H:\Python\Web Development\DjangoClass\projectname>
```

These are the Files Created:



```
PROJECTNAME
  appname
    __pycache__
    migrations
      __pycache__
      __init__.py
      0001_initial.py
```

This Should be the Content of the *projectname/appname/migrations/0001_initial.py*

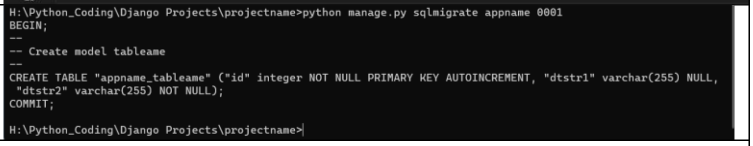


appname > migrations > 0001_initial.py > ...

```python
# Generated by Django 4.1.5 on 2023-01-18 15:54

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='tableame',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('dtstr1', models.CharField(default='', max_length=255, null=True)),
                ('dtstr2', models.CharField(max_length=255)),
            ],
        ),
    ]

```

Another Way to Check what the Migration Executed is:

`py manage.py sqlmigrate appname 0001`



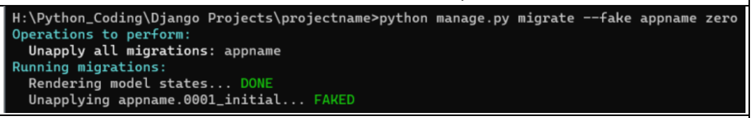
```
H:\Python_Coding\Django Projects\projectname>python manage.py sqlmigrate appname 0001
BEGIN;
--
-- Create model tableame
--
CREATE TABLE "appname_tableame" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "dtstr1" varchar(255) NULL,
 "dtstr2" varchar(255) NOT NULL);
COMMIT;

H:\Python_Coding\Django Projects\projectname>
```

NOTE:

https://sqliteonline.com/ ***USE THIS SITE TO CHECK THE SQLITE DATABASE.***

Incase of Error in Creation of SQL Table Run

`python manage.py migrate --fake appname zero`



```
H:\Python_Coding\Django Projects\projectname>python manage.py migrate --fake appname zero
Operations to perform:
  Unapply all migrations: appname
Running migrations:
  Rendering model states... DONE
  Unapplying appname.0001_initial... FAKED
```

Then Rerun all the Above CMD Commands for Migrations

Read &or Write Data

In the *projectname/appname/views.py*

FROM

```python
    if request.method == 'POST':
        num1 = request.POST['num1'] # input1 = request.GET['num1']
        num2 = request.POST['num2'] # input2 = request.GET['num2']
        if 'add' in request.POST:
            result = int(num1) + int(num2)
            return render(request,'index.html',{'KEY':result})
    return render(request,'index.html')
```

TO

```python
    if request.method == 'POST':
        if request.POST.get('data1') and request.POST.get('data2'):
            post = tablename()
            post.dtstr1 = request.POST.get('data1')
            post.dtstr2 = request.POST.get('data2')
            post.save()
            return render(request, 'index.html')
    else:
        objectName = tablename.objects.all()
        return render(request, 'index.html', {'KEY': objectName})
```

In the File *projectname/templates/index.html*

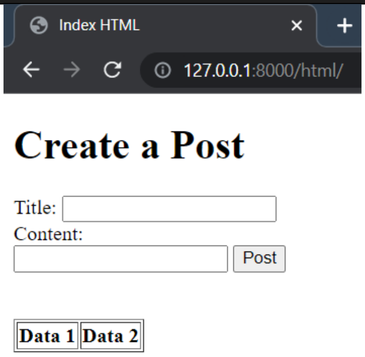
FROM

```
<form method="POST">
    {% csrf_token %}
    <label for="num1">Enter Any Number: </label>
        <input type="text" name="num1"/>
    <label for="num2">Enter Any Number: </label>
        <input type="text" name="num2"/>
    <br/>
    <button type="submit" name="add">+</button>
</form>
<br><br><br>
<h3>The Result is = </h3><br>
<h4>{{KEY}}</h4>
```

TO

```
<h1>Create a Post </h1>
<form action="" method="POST">
    {% csrf_token %}
    Title: <input type="text" name="data1"/><br/>
    Content: <br/>
    <input type="text" name="data2"/>
    <input type="submit" value="Post"/>
</form>
<br><br>
<table border="1">
    <tr>
        <th>Data 1</th>
        <th>Data 2</th>
    </tr>
    {% for x in KEY %}
    <tr>
        <td>{{ x.dtstr1 }}</td>
        <td>{{ x.dtstr2 }}</td>
    </tr>
    {% endfor %}
</table>
```

Result Output



Create a Post

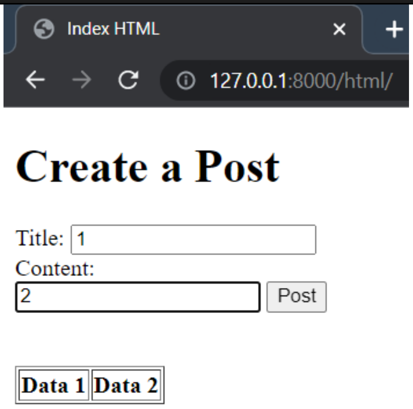
Title:

Content:

Post

Data 1	Data 2

1



Create a Post

Title: 1

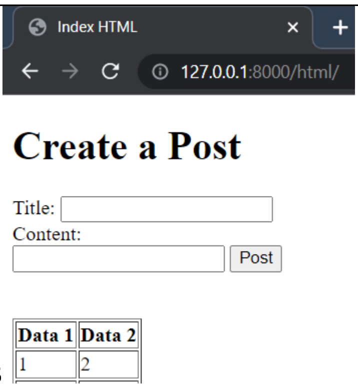
Content: 2

Post

Data 1	Data 2

2

---

3 

Create a Post

Title:

Content:

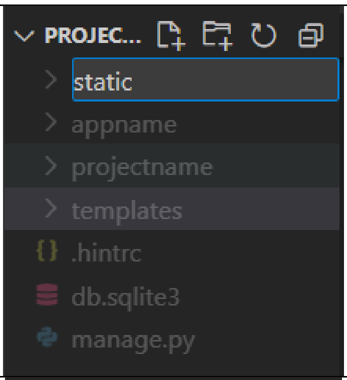
Post

Data 1	Data 2
1	2

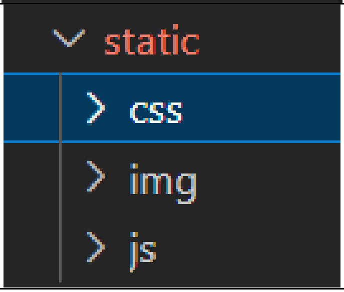
Static Folder

Initialization

Create a ***static*** Folder in the Base Directory

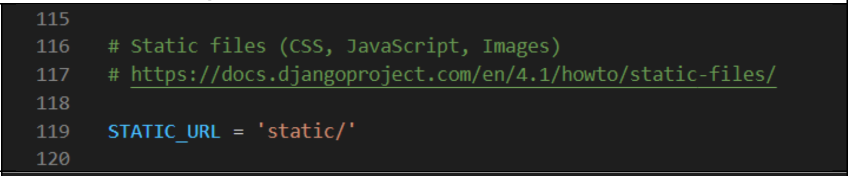


& Add the Resources Folders in it.



In the *projectname/projectname/settings.py* Edit the Following Code

FROM



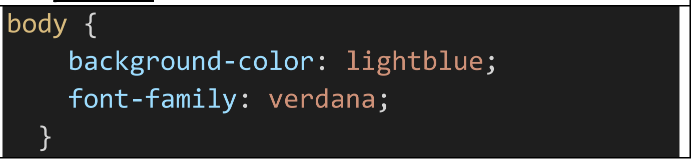
```python

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = 'static/'

```

TO

```python
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
STATIC_ROOT = os.path.join(BASE_DIR, 'assets')
```

Create Resources Folders

CSS

Create a CSS File called ***first.css*** then	
open *projectname\static\css\first.css* & Add Script to It	

```
body {
    background-color: lightblue;
    font-family: verdana;
  }
```

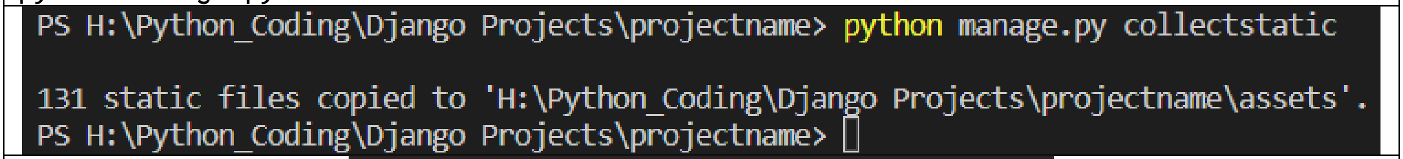
JS

IMG

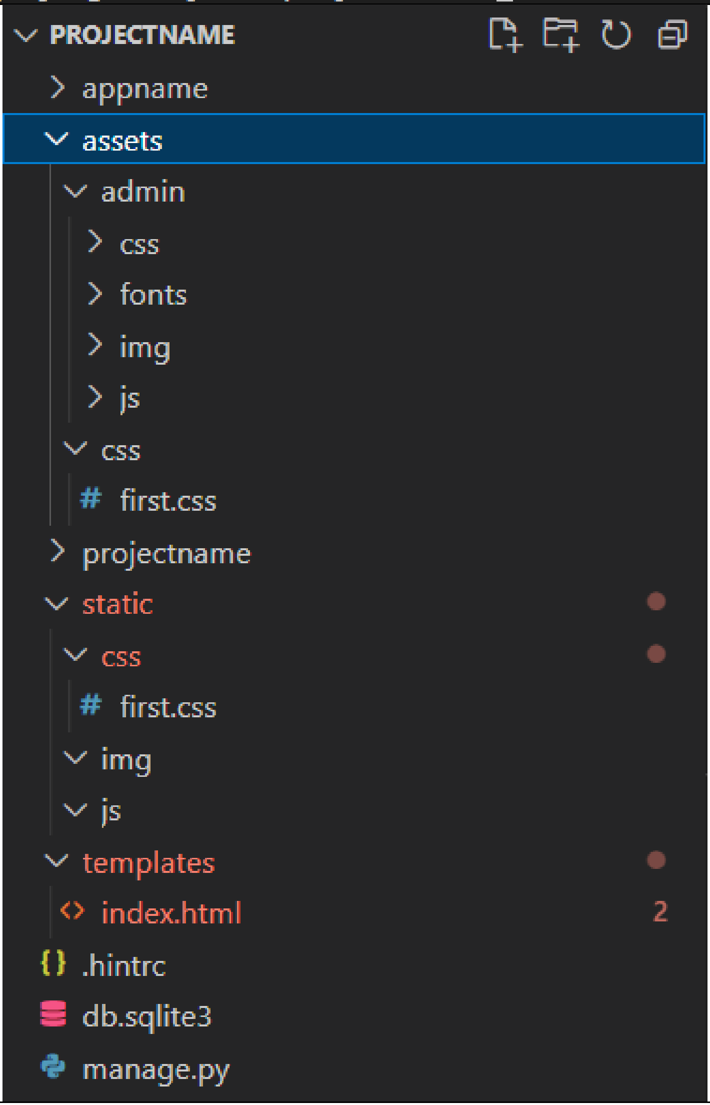
Fonts

Collect Static

```
python manage.py collectstatic
```



```
PS H:\Python_Coding\Django Projects\projectname> python manage.py collectstatic

131 static files copied to 'H:\Python_Coding\Django Projects\projectname\assets'.
PS H:\Python_Coding\Django Projects\projectname> 
```



```
PROJECTNAME
  > appname
  v assets
    v admin
      > css
      > fonts
      > img
      > js
    v css
      # first.css
  > projectname
  v static
    v css
      # first.css
    v img
    v js
  v templates
    <> index.html                2
  {} .hintrc
  db.sqlite3
  manage.py
```

Connect to HTML Page

Open *projectname\template\index.html* & Edit It

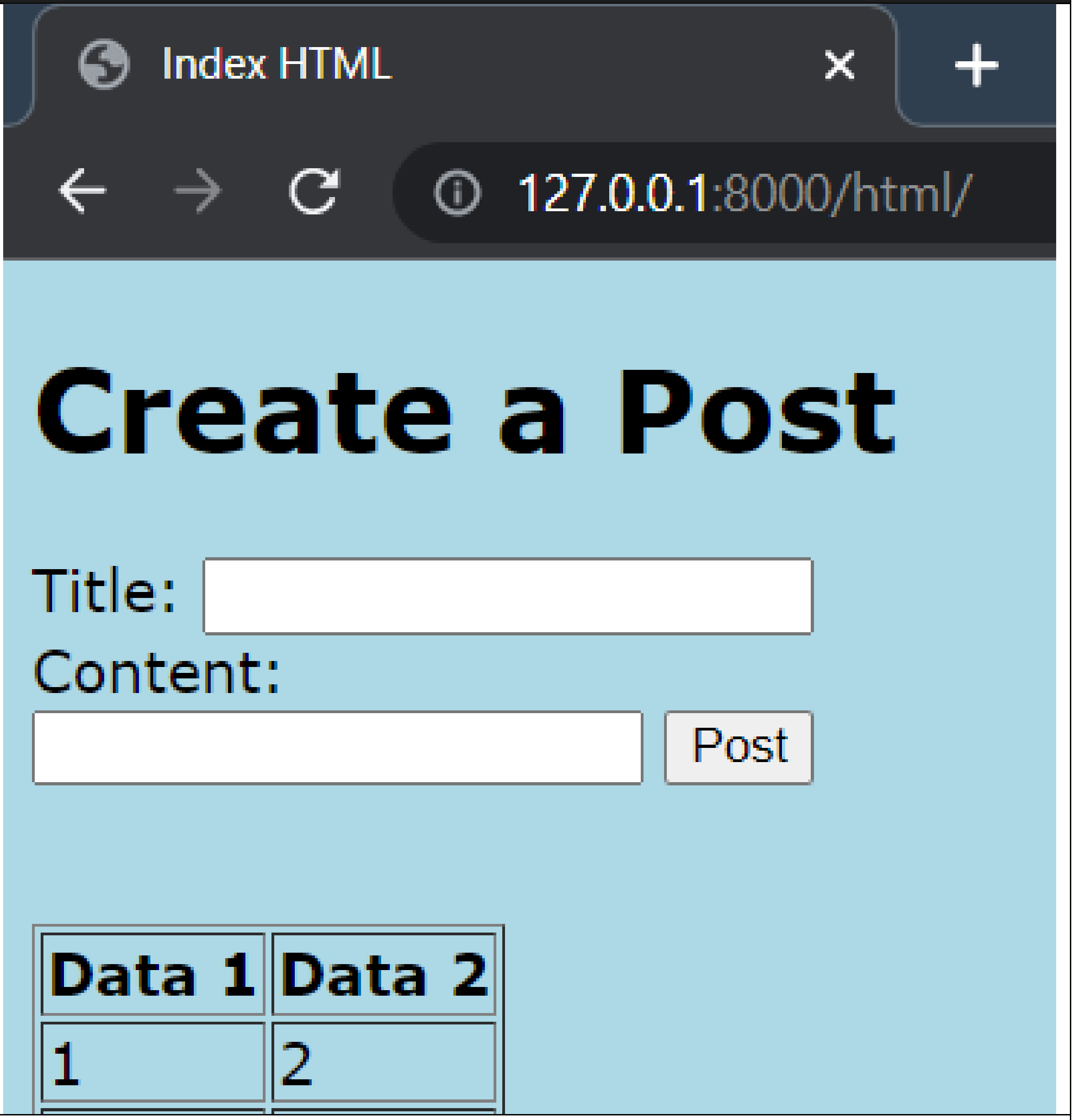
FROM

```
<!DOCTYPE html>
<html lang="en">
<head>
```

TO

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
    <link rel="stylesheet" href="{% static '\css\first.css' %}">
<head>
```

Result Output

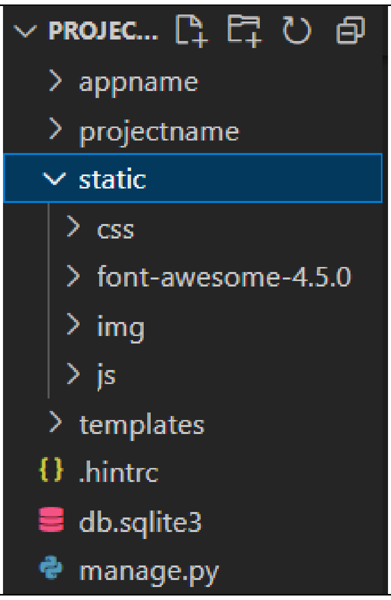
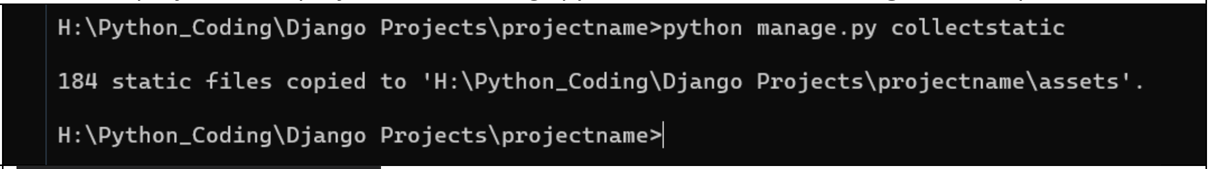
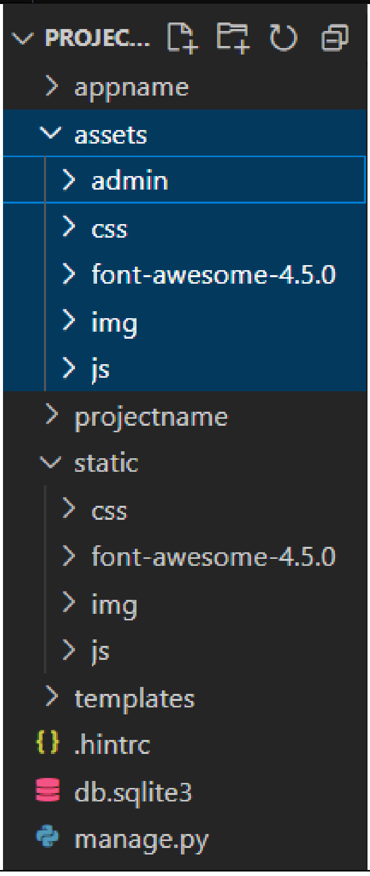


Create a Post

Data 1	Data 2
1	2

Note: We Can Do the Same for The Other Resources Not Only CSS Files.

Project

First Lets Gets a Free Template-	https://templatemo.com/tag/one-page
Add All the Resources to the ***static*** Folder	 PROJEC... appname projectname static css font-awesome-4.5.0 img js templates .hintrc db.sqlite3 manage.py
Make Sure the Content in *projectname/projectname/settings.py* Make Sure All the Settings are Completed	
python manage.py collectstatic	 H:\Python_Coding\Django Projects\projectname>python manage.py collectstatic 184 static files copied to 'H:\Python_Coding\Django Projects\projectname\assets'. H:\Python_Coding\Django Projects\projectname>
These Files will be Created in the Assets Folder.	 PROJEC... appname assets admin css font-awesome-4.5.0 img js projectname static css font-awesome-4.5.0 img js templates .hintrc db.sqlite3 manage.py

THEN FIRST	Copy All the HTML Files into the ***templates*** Folder
Add the Line in the First Line of all the ***html files***	`{% load static %}`
Change All The Links of the Resources Folder to route through ***static*** in all the ***html files***	
EXAMPLE CHANGE FROM	`<link rel="stylesheet" href="projectname/static/css/bootstrap.min.css">`
TO	`<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">`
DO THIS TO ALL THE LINKS WHICH ARE INTERNAL (NOT EXTERNAL .com'S)	
FINALLY	Create the Loops in the ***HTML Files*** and Load all the Data Through Templates Created Through models.py (Do this to Reduce the Number of Lines in the HTML Files).

Refer

HTML

https://www.w3schools.com/html/default.asp

https://www.w3schools.com/css/default.asp

https://www.w3schools.com/js/default.asp

https://www.w3schools.com/howto/default.asp

https://www.w3schools.com/bootstrap/bootstrap_ver.asp

SQL

https://www.w3schools.com/sql/default.asp