2CSDE86 Application Development Frameworks (ADF)

Lecture-4

Django Views

7th CSE

Daiwat Vyas & Ajaykumar Patel

Disclaimer

- Some of the content in the ppt is taken from various online sources and reference books and after referring it was considered for including it in the ppt slides.

Pre-requisite for this session

- **Students should have already installed Python, vscode and setup Django on their system.**

- **A file was sent for Django installation and it had all step by step procedure mentioned for installing Python, vscode and setting up Django in their system.**

Django Views

- Django views are part of the user interface — they usually render the HTML/CSS/Javascript in your Template files into what you see in your browser when you render a web page.

- (Note that if you've used other frameworks based on the MVC (Model-View-Controller), do not get confused between Django views and views in the MVC paradigm.

- Django views roughly correspond to controllers in MVC, and Django templates to views in MVC.)

Django Views

- Django Views are one of the vital participants of M**V**T Structure of Django.

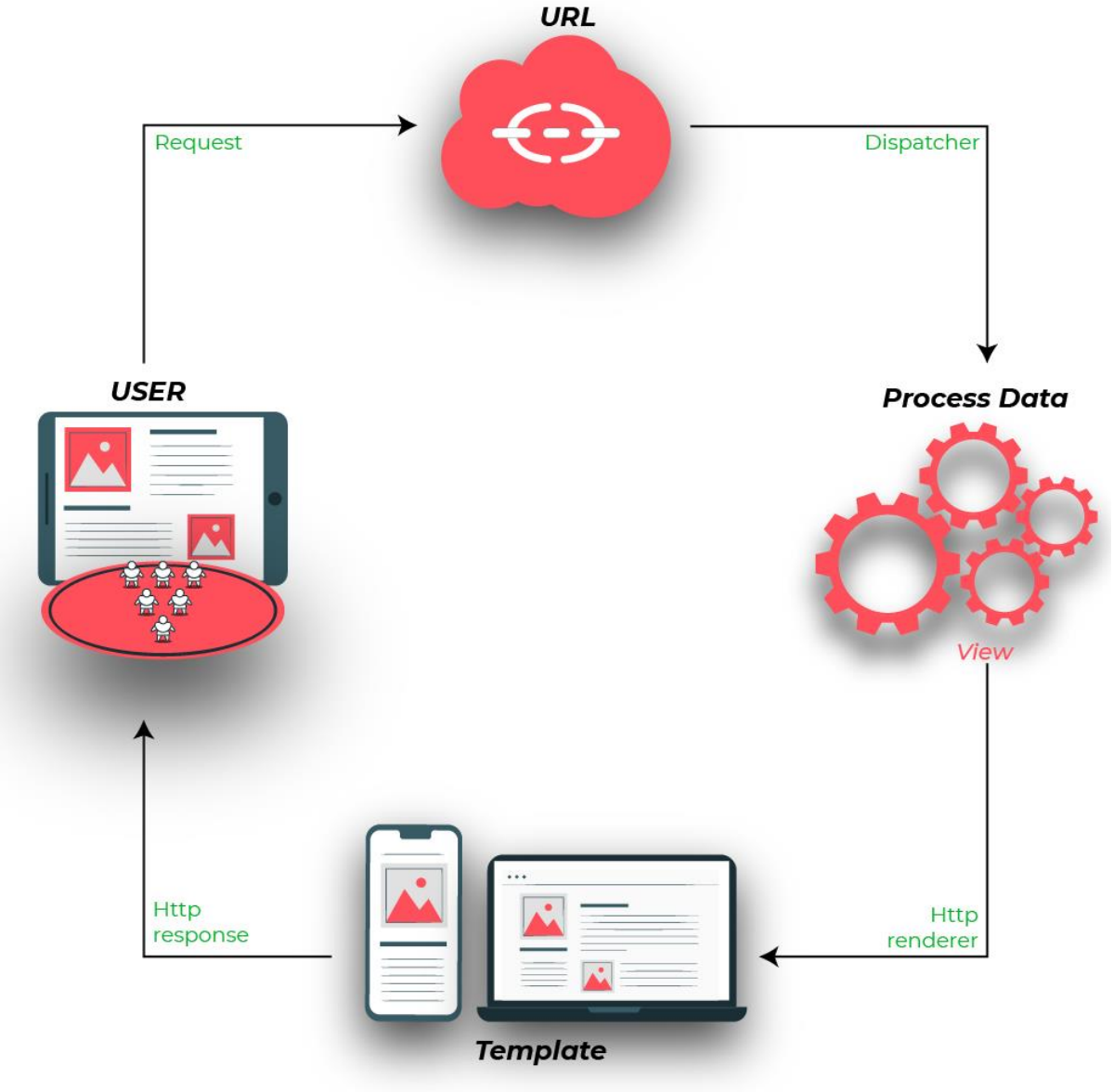
- A View is the user interface — what you see in your browser when you render a website.

- It is represented by HTML/CSS/ JavaScript files.

- As per Django Documentation, A view function is a Python function that takes a Web request and returns a Web response.

- This **response** can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, anything that a web browser displays.

Django Views (contd…)

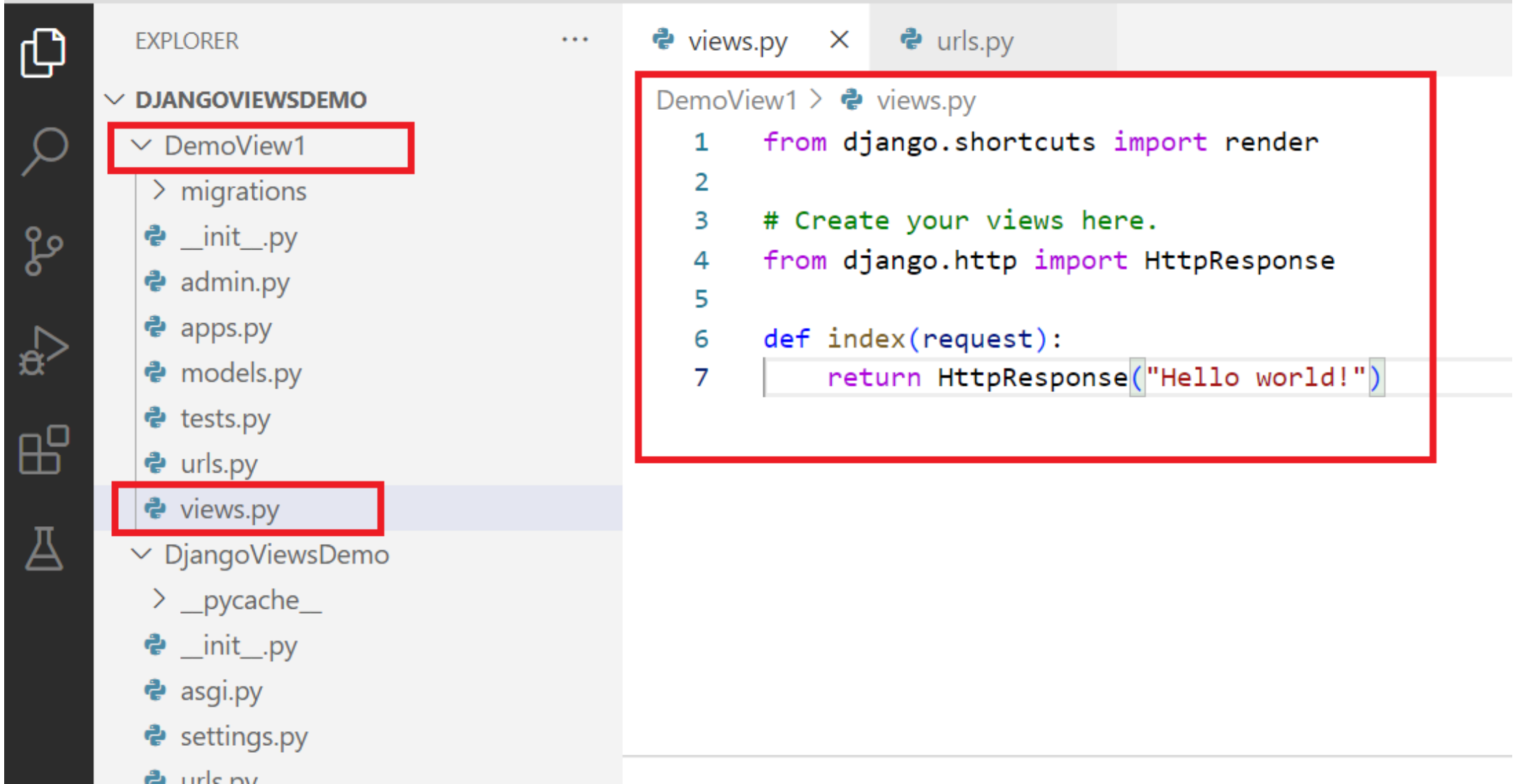


Django Views (contd…)

- **Example:**
- django-admin startproject DjangoViewsDemo (For creating a new Django Project)
- python manage.py startapp DemoView1
- Ensure that you have created a Project in Django and an App (module) in that Project. *(Recall last lab session and lecture session)*
- Go to views.py file in the Django App (Module) folder. *(We will make changes in it)*

Django Views (contd…)

- **Example:** *views.py*



Django Views (contd…)

- **Example:** *urls.py*

- This is a simple example on how to send a response back to the browser.

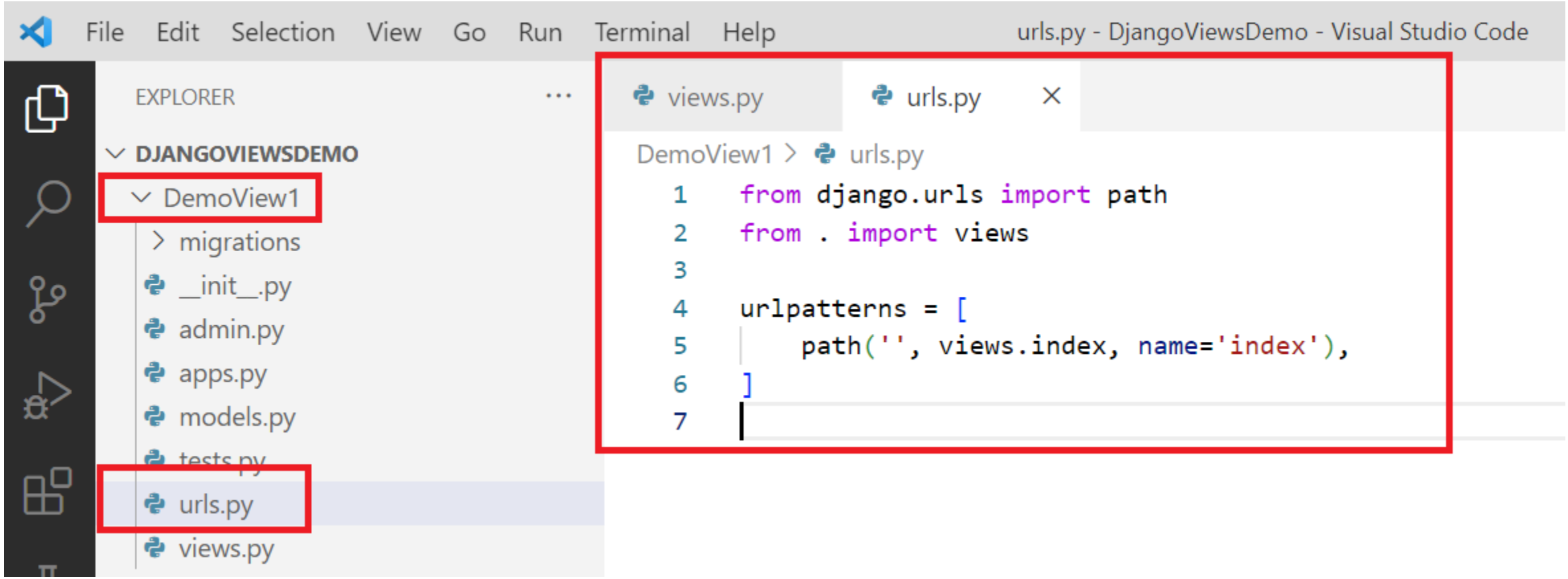
- But how can we execute the view? Well, we must call the view via a URL.

- Create a file named urls.py in the same folder as the views.py i.e in App folder

- Update the urls.py file

Django Views (contd…)

- **Example:** *urls.py*



Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- This function returns an element that needs to be included in urlpatterns. That is, path acts as a link between an element (for eg Views) and its URL.

- **route:** This is the URL for a particular view. For eg: ‘<name>/’ is a route.

- So when we request this page from the URL, the server will return the view linked to it.

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- ***view:*** Here we need to write, the view name that we need to link. Or use the function “include” to add another urls.py file. (Like in the project/urls.py file)

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- *****kwargs and *args:*** args in function definition in python is a syntax used for the variable input argument list. It is used with a single asterisk.
- That is if for example, we need to input any variable number of arguments for a function, then we use *args. Eg:

```
def myFun(*args):
    for arg in args:
        print(arg)


myFun("Hello","There","Hi","There")
```

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- *****kwargs and *args:***
- So in the example we can give any number of arguments and *args will take up all of them. We can think that all the arguments are being saved by arg as some list, thereby using the line for arg in args, we are taking each element from the list.

- the * is splitting up the list into elements thus *args gives you all the elements separately and args will give the elements as a list.

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- *****kwargs and *args:*** kwargs in function definitions in Python are used for key-worded, variable arguments list.

- It is used with double asterisk. Eg:

```
def myFun(**kwargs):
    for item,price in kwargs.items():
        print(f"{item}={price}")

myFun(Book=100,Pen=10,Watch=4000)
```

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- *****kwargs and *args:*** Eg:

- As you can see in the above example, we are able to pass complete values. The variable names and the values held by those variables with the use of the assignment operator.

- We can think of the arguments being saved as a Python dictionary by the kwargs thus by using the line for the item, price in kwargs.items(), we are taking up the item(Book, Pen, etc) and their corresponding price(100,10, etc).

- The ** splits the dictionary into its elements. Thus **kwargs gives you all the key-worded elements separately while kwargs gives you the key-worded elements as a dictionary.

Django Views (contd…)

- **Example:** *urls.py*

- *path(route,view, kwargs, name)*

- ***name:*** *name is used to specify the name of the particular view that the URL is linking.*

- ***include(module,namespace=None)***
- This function takes another URL conf file that should be included by using this include function. That is to form a link with another urls.py file you should use include function. The namespaces can also be written inside but we don’t need to do that for now.

Django Views (contd…)

- **Example:** *urls.py*

- The urls.py file we just created is specific for the *DemoView1* application of the project *DjangoViewsDemo*.

- There will be a need to do some routing in the root directory i.e project root folder. as well. Just follow the instructions below:

- There is a file called urls.py in main project folder, open that file and add the include module in the import statement, and also add a path() function in the urlpatterns[] list, with arguments that will route users that comes in via 127.0.0.1:8000/DemoView1/

Django Views (contd…)

- **Example: Create a new app DemoView2 in the project DjangoViewsDemo**

- *python manage.py startapp DemoView2*

Django Views (contd…)

- **Example:** *views.py in DemoView2 app*

DemoView2 > views.py

```python

# Create your views here.
# import Http Response from django
from django.http import HttpResponse
# get datetime
import datetime

# create a function
def demo_view(request):
    # fetch date and time
    now = datetime.datetime.now()
    # convert to string
    html = "Time is {}".format(now)
    # return response
    return HttpResponse(html)
```

Django Views (contd…)

- **Example:** *views.py in DemoView2 app*

- First, we import the class HttpResponse from the django.http module, along with Python's datetime library.

- Next, we define a function called demo_view. This is the view function. Each view function takes an HttpRequest object as its first parameter, which is typically named request.

- The view returns an HttpResponse object that contains the generated response. Each view function is responsible for returning an HttpResponse object.

- In next session we will discuss in details regarding, Django's HTTP Request-Response cycle.

Django Views (contd…)

- **Example:** (*Line by line explanation*)

- **Django Request and Response cycle – HttpRequest and HttpResponse Objects will be discussed later on.**

- Let us now get the demo_view working:

- Create a urls.py file in the Django App (Module).

Django Views (contd…)

- **Example: Create** *urls.py in DemoView2 app*

DemoView2 > urls.py

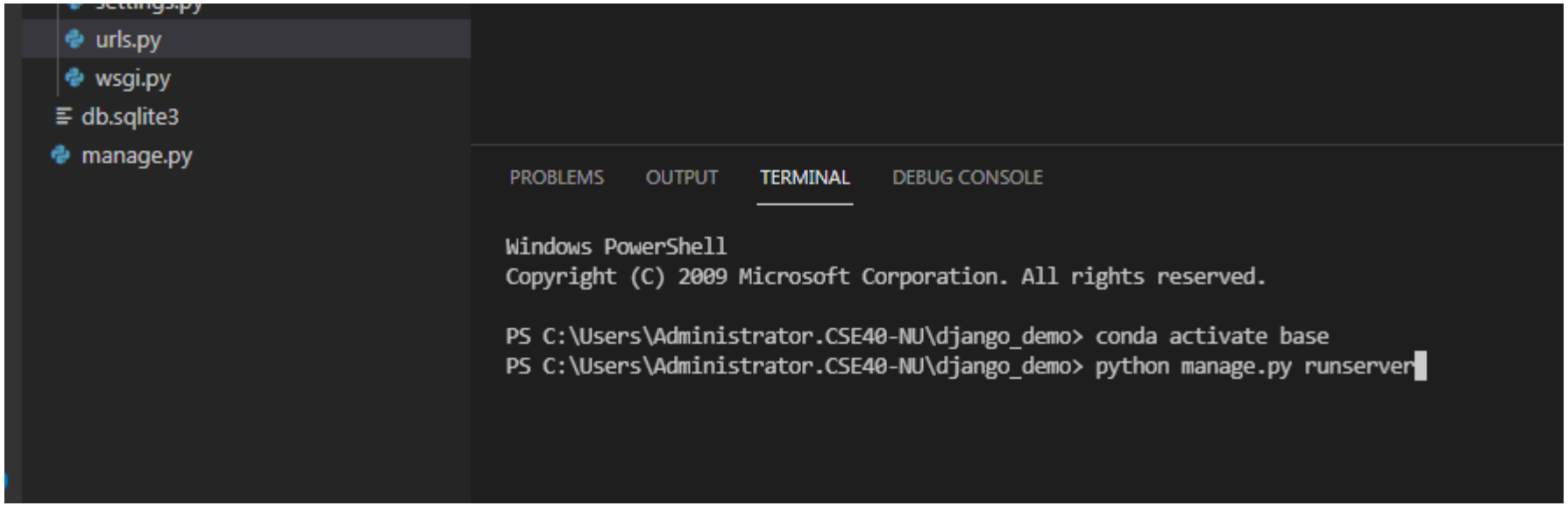
```python
from django.urls import path

# importing views from views..py
from .views import demo_view

urlpatterns = [
    path('', demo_view),
]
```

Django Views (contd…)

- **Example:**

- Now, go to in-built terminal and run the server using following command:

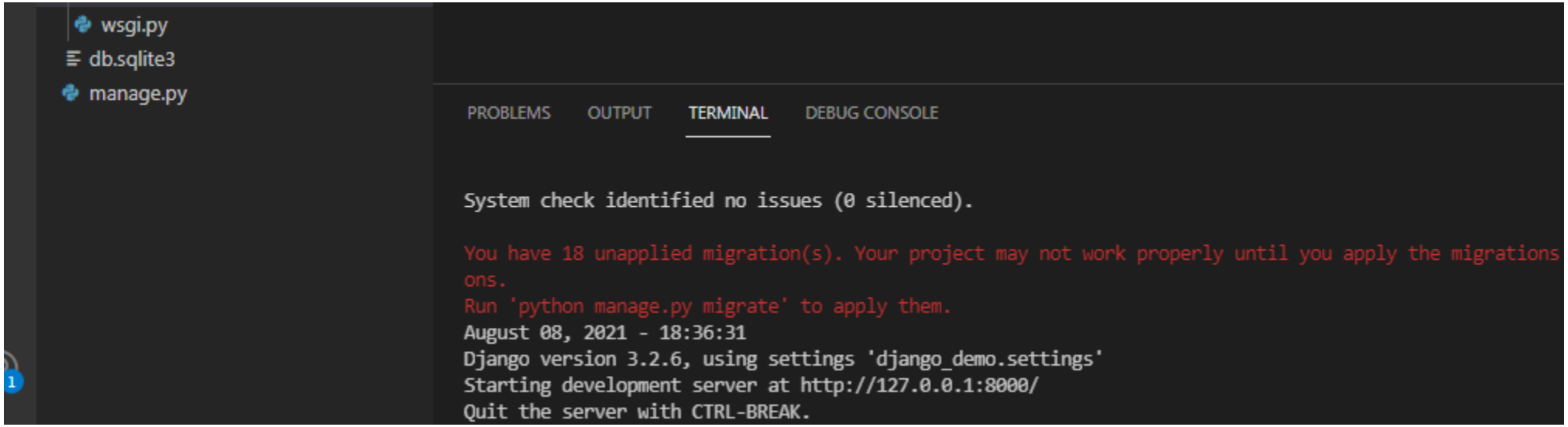
- **python manage.py runserver**



Django Views (contd…)

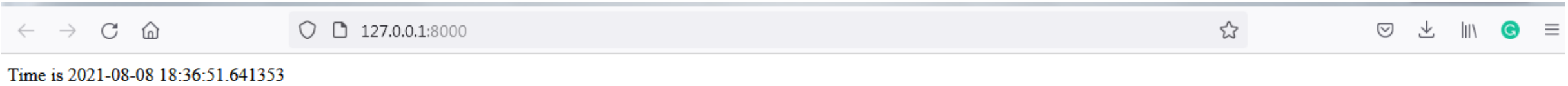
- **Example:**

- Press ctrl and click on the http://127.0.0.1:8000 in terminal window



Django Views (contd…)

- **Example:** *Output*



Time is 2021-08-08 18:36:51.641353

Django Views (contd…)

- Let us take one more example and understand the views in detail.

- Go to terminal and run following command:

- *python manage.py startapp demo_module2*

- **This will help in creating a module/app for the Django based web application that you want to create**

Django Views (contd…)

urls.py - django_demo - Visual Studio Code [Administrator]

EXPLORER

- DJANGO_DEMO
  - demo_module1
  - demo_module2
  - django_demo
  - db.sqlite3
  - manage.py

views.py | urls.py demo_module1 | urls.py django_demo

django_demo > urls.py > ...

```python
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from django.urls import include, path


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('demo_module1.urls')),
]
```

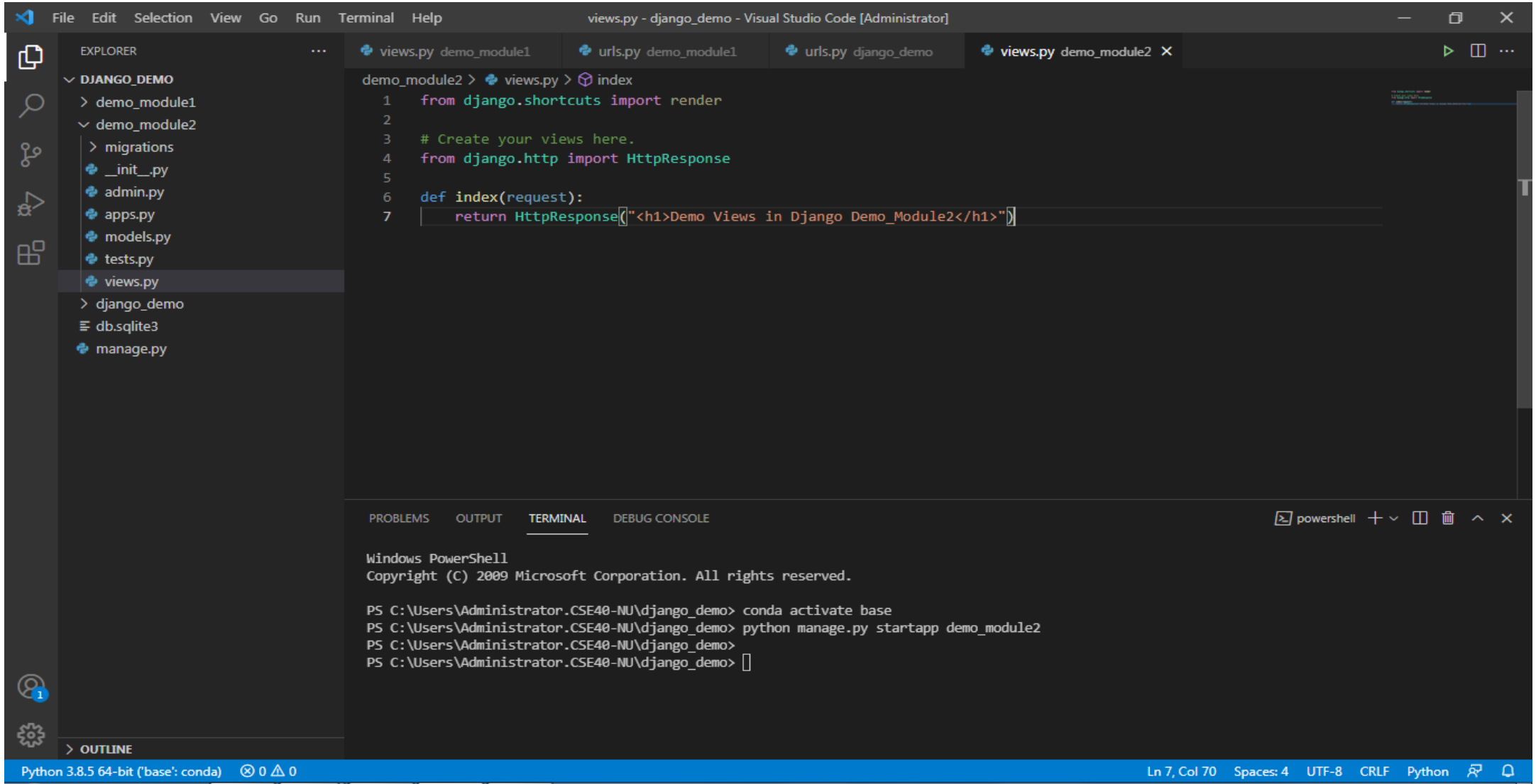
PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator.CSE40-NU\django_demo> conda activate base
PS C:\Users\Administrator.CSE40-NU\django_demo> python manage.py startapp demo_module2
PS C:\Users\Administrator.CSE40-NU\django_demo>
PS C:\Users\Administrator.CSE40-NU\django_demo>
```

Python 3.8.5 64-bit ('base': conda) ⊗ 0 ⚠ 0 | Ln 24, Col 2 | Spaces: 4 | UTF-8 | CRLF | Python

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*



Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    return HttpResponse("<h1>Demo Views in Django Demo_Module2</h1>")
```

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    return HttpResponse("<h1>Demo Views in Django Demo_Module2</h1>")
```

This is your view function. It’s an example of a function-based view. It takes a request from your web browser and returns a response. In this simple case, it’s just a line of text formatted as an HTML heading. (Types of views will be discussed in next slides)

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

- ***Configuring urls***

- If you started the development server now, you will notice it would still display the welcome page.

- For Django to use your new view, you need to tell Django the index view is the view you want to display when someone navigates to the site root (home page). ***We do this by configuring our URLs.***

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

- ***Configuring urls***

- In Django, the *path()* function is used to configure URLs.

- In its basic form, the *path()* function has a very simple syntax:

```
path(route, view)
```

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

- ***Configuring urls***

- A practical example of the basic path() function would be:

- path('mypage/', views.myview)

- In this example, a request to *http://example.com/mypage* would route to the myview function in the application's views.py file.

- The path() function statements live in a special file called urls.py.

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

- ***Configuring urls***

- When startproject created our website i.e django_demo project, it created a urls.py file in our site folder (\django_demo\urls.py).

- This is the correct place for site-wide navigation, but is rarely a good place to put URLs relating to individual applications.

- Not only is having all our URLs in the one file more complex and less portable, but it can lead to strange behavior if two applications use a view with the same name.

Django Views (contd…) views example in detail

- **Example 2:** *views.py django_module2*

- ***Configuring urls***

- To solve this problem, we create a new urls.py file for each module/application.

- If you are wondering why startapp didn’t create the file for us, not all apps have public views accessible via URL.

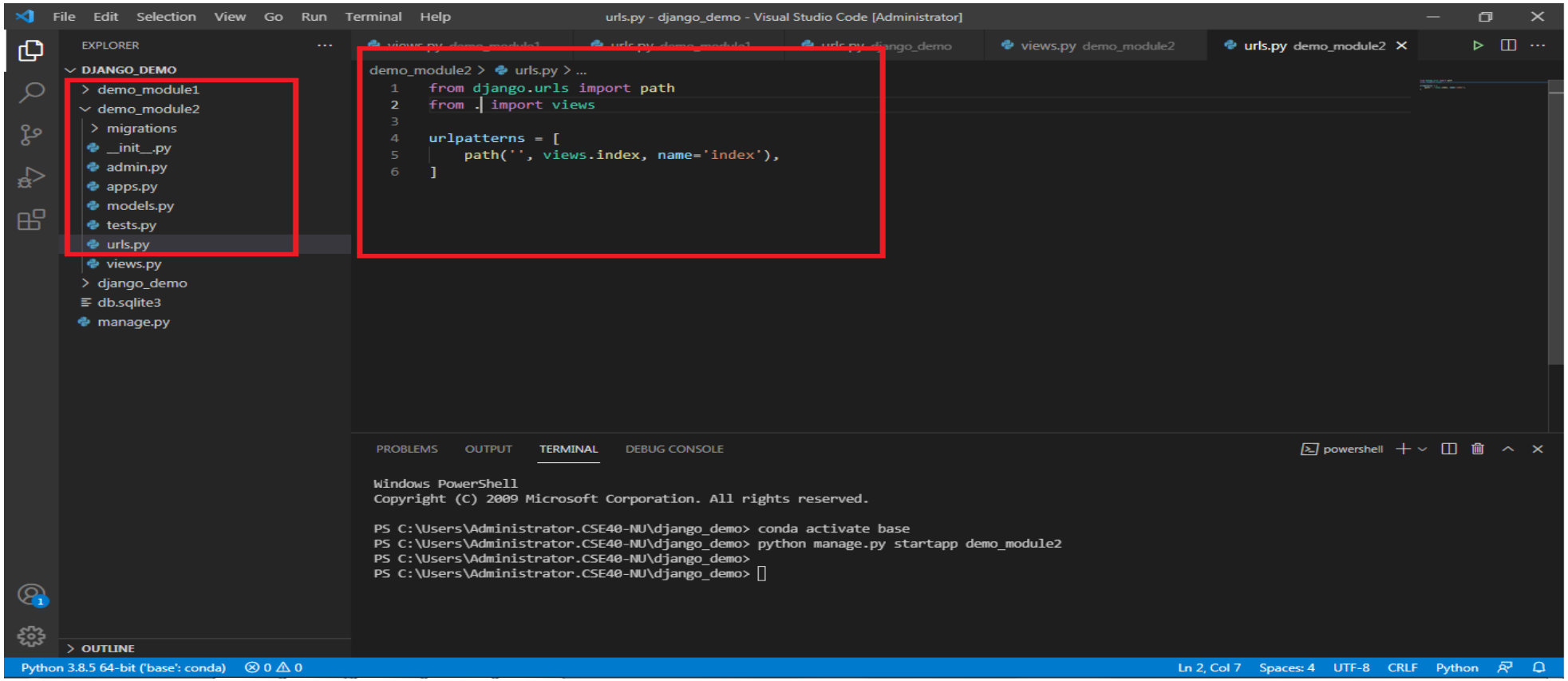
- For example, a utility program that performs background tasks would not need a urls.py file.

- For this reason, Django lets you decide whether your app needs its own urls.py file.

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_module2*

- *Create a urls.py file in the django_module2 app/module*



Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_module2*

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_module2 (Line by Line explanation of code)*

- **Line 1** imports the path() function. This import is necessary for the URL dispatcher to work and is common to all urls.py files.

- **Line 2** imports the local views.py file. The dot operator (“.”) in this case is shorthand for the current package, so this is saying “import all views from the current package (events)”.

- **Line 4** lists the URL patterns registered for this app. For readability, the list is broken into multiple lines, with one URL pattern per line.

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_module2 (Line by Line explanation of code)*

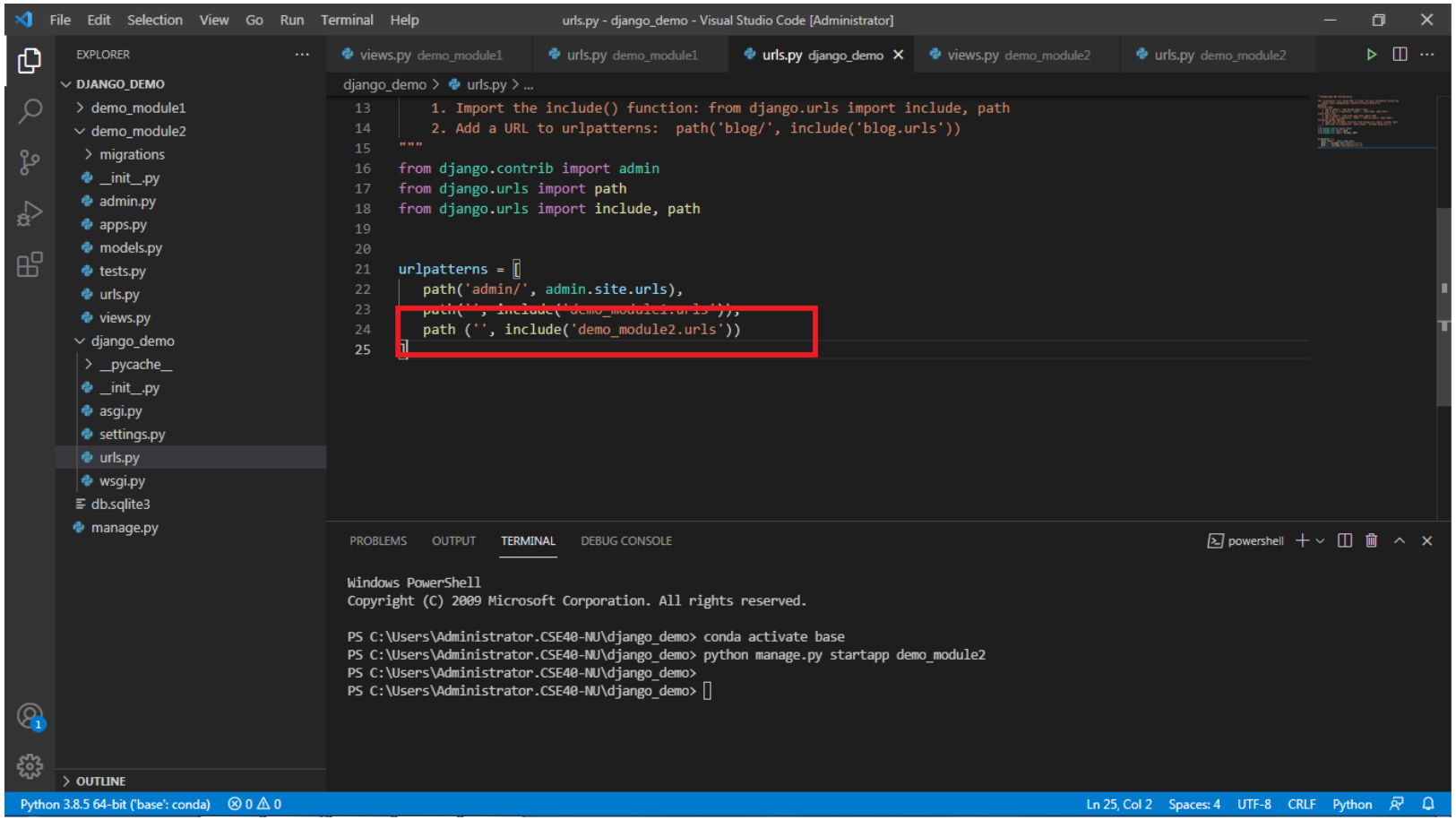
- **Line 5** is the actual URL dispatcher: ' ' matches an empty string. It will also match the "/" as Django automatically removes the slash.

- In other words, this matches both http://example.com and http://example.com/. views.index points to our index view.
- i.e., the dot operator is pointing to the index view inside the views.py file that we imported in line 2. name='index'.
- While it's optional, you should always name your URLs. We name URLs so they can be referred to in code (reverse lookup). URL reversing is common in both templates and views, so you will see several examples as we see more examples.

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_demo*

- Go to urls.py file in the django_demo project



Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_demo*

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


urlpatterns = [
  path('admin/', admin.site.urls),
  path('', include('demo_module1.urls')),
  path ('', include('demo_module2.urls'))
]
```

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_demo (Line by Line explanation)*
- **Line 18** We have added the include() function to our imports.

- **Line 24** We have added a new URL dispatcher. In this file, the dispatcher is including the urls.py file from the demo_module2 app.

- The empty string (' ') will match everything after the domain name.

- This pattern must be the last entry in the urlpatterns list, otherwise Django's shortcut logic will switch to the events app before trying to match any of the other site URLs.

Django Views (contd…) views example in detail

- **Example 2:** *urls.py django_demo (Line by Line explanation)*
- **Line 18** We have added the include() function to our imports.

- **Line 24** We have added a new URL dispatcher. In this file, the dispatcher is including the urls.py file from the demo_module2 app.

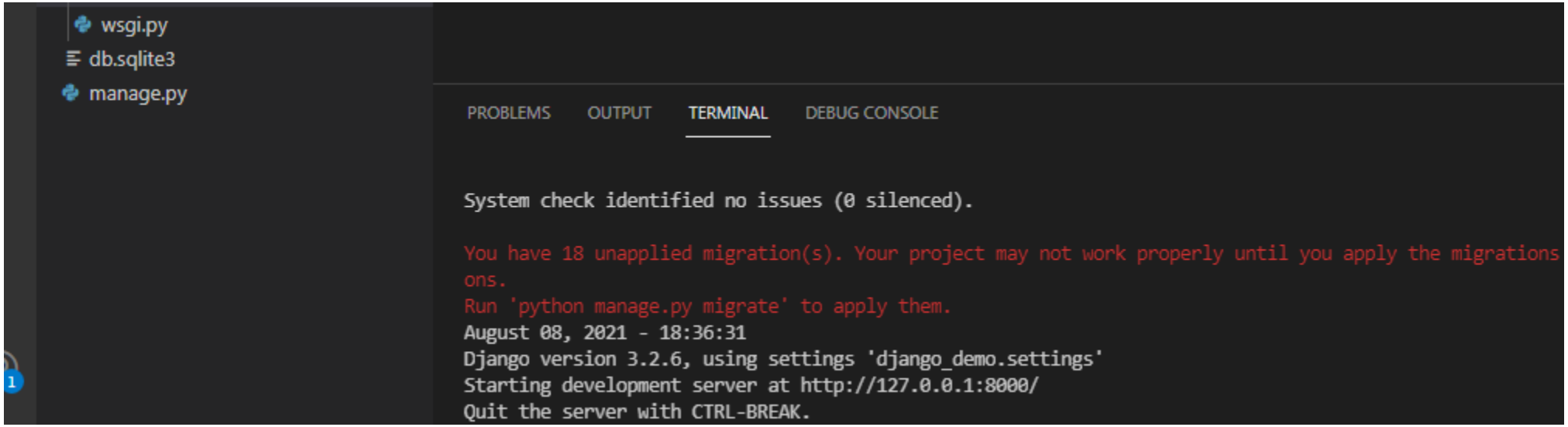
- The empty string (' ') will match everything after the domain name.

- This pattern must be the last entry in the urlpatterns list, otherwise Django's shortcut logic will switch to the events app before trying to match any of the other site URLs.

Django Views (contd…)

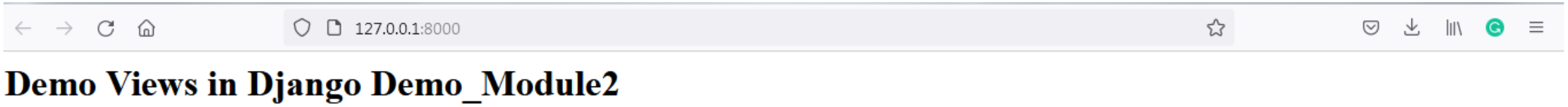
- **Example 2:**

- Press ctrl and click on the http://127.0.0.1:8000 in terminal window



Django Views (contd…)

- **Example 2: output**



Django Views (contd…)

- **Summary: How it worked?**

- **Think and answer…**

Django Views (contd…)

- **Summary: How it worked?**

- Our browser sent a message to the Django development server requesting it return content located at the root URL (http://127.0.0.1:8000/).

- Django then looked for a URL pattern matching the request, by first searching the site level urls.py, and then each of the apps for a urls.py file containing a pattern that matches.

- Django checks the first pattern (admin/) in our site level urls.py which doesn’t match and moves on to the second line in which the empty string (root URL) matches.

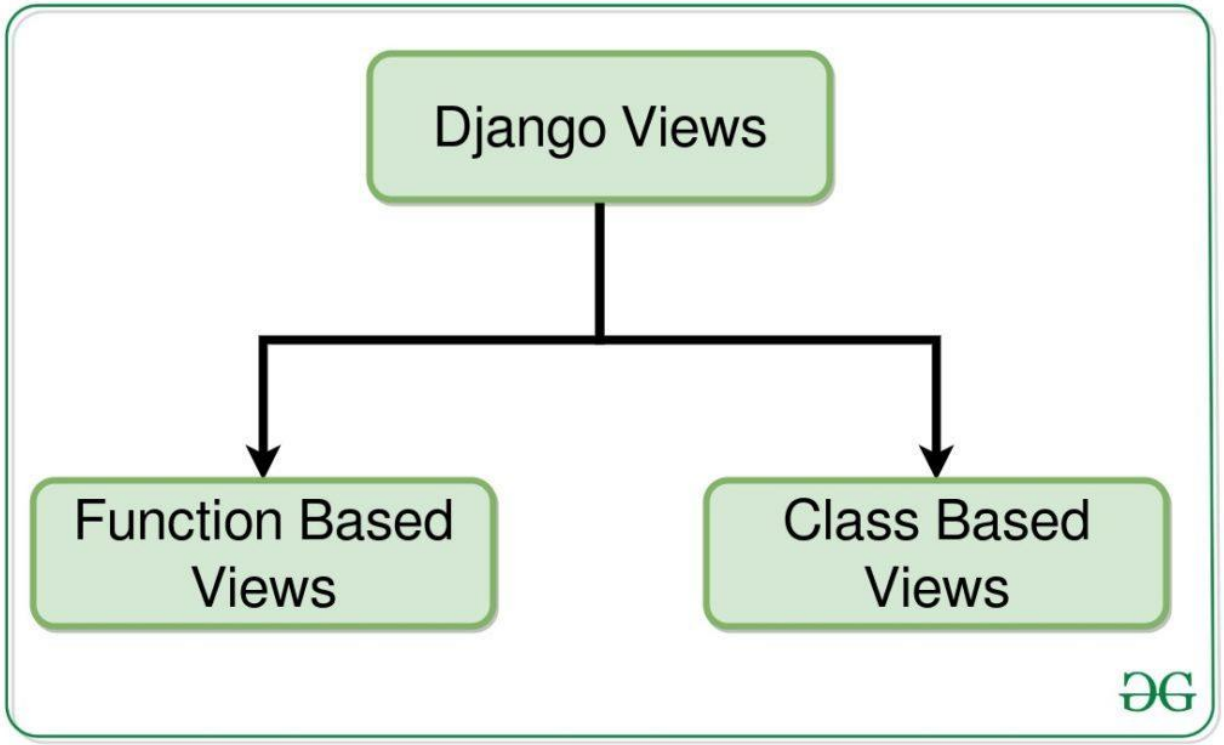
Django Views (contd…)

- **Summary: How it worked?**
- The matching pattern includes the urls.py from the events app. Basically, this include says “go look in the events app for a pattern that matches”.
- Once in the app-level urls.py, the empty string matches again. But this time, the request is sent to the index view.
- The index view then renders our simple HTML message to a HttpResponse and sends it to the browser.
- The browser renders the response and we see our page heading.
- Every Django application follows this same basic process each time it receives a request from the browser.

Types of Views

- Django views are divided into two major categories :-

  - Function Based Views
  - Class Based Views



Types of Views (contd…)

- **Function Based Views**
- Function based views are writer using a function in python which receives as an argument HttpRequest object and returns an HttpResponse Object.

- Function based views are generally divided into 4 basic strategies, i.e., CRUD (Create, Retrieve, Update, Delete).

- CRUD is the base of any framework one is using for development.

Types of Views (contd…)

- **Class Based Views**
- Class-based views provide an alternative way to implement views as Python objects instead of functions.

- They do not replace function-based views, but have certain differences and advantages when compared to function-based views:

  - Organization of code related to specific HTTP methods (GET, POST, etc.) can be addressed by separate methods instead of conditional branching.

  - Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

Types of Views (contd…)

- **Class Based Views**

- Class-based views are simpler and efficient to manage than function-based views.

- A function-based view with tons of lines of code can be converted into a class-based views with few lines only.

- This is where Object-Oriented Programming comes into impact.

Next Lecture Agenda

- **Django Request and Response cycle – HttpRequest and HttpResponse Objects will be discussed later on.**

- **Types of Views with example**