***DJANGO***

Benefits of Django

It not only acts as a quick solution for web development, its ability to deliver high-quality code and transparent writing provides the right platform to customers for business and even for the developers.

* **Fast:** This has been designed in a way to help the developers make an application as fast as possible. From idea, production to release, Django helps in making it both cost effective and efficient. Thus it becomes an ideal solution for developers having a primary focus on deadlines.
* **Fully Loaded:** It works in a way that includes dozens of extras to help with user authentication, site maps, content administration, RSS feeds and much more such things. These aspects help in carrying out the web development process completely.
* **Secure:** When you are doing it in Django, it is ensured that developers don’t commit any mistakes related to security. Some of the common mistakes include SQL injection, cross-site request forgery, clickjacking and cross-site scripting. To manage effectively usernames and passwords, the user authentication system is the key.
* **Scalable:** To meet the heaviest traffic demand, the benefits of Django framework can be seen. Therefore, the busiest sites use this medium to quickly meet the traffic demands.
* **Versatile:** Content management, scientific computing platforms, and even big organizations, all these aspects are very efficiently managed by the use of Django.

Steps to make Django project

1. Install Django package through command

(python -m pip install Django)

It will install the latest version of Django

1. Now go to the location where you want to make your project and run the following command to create Django project

(Django-admin startproject projectname)

This will create the following scenario(suppose if the project name is project1)

.project1(folder)

.project1(folder)

.settings.py

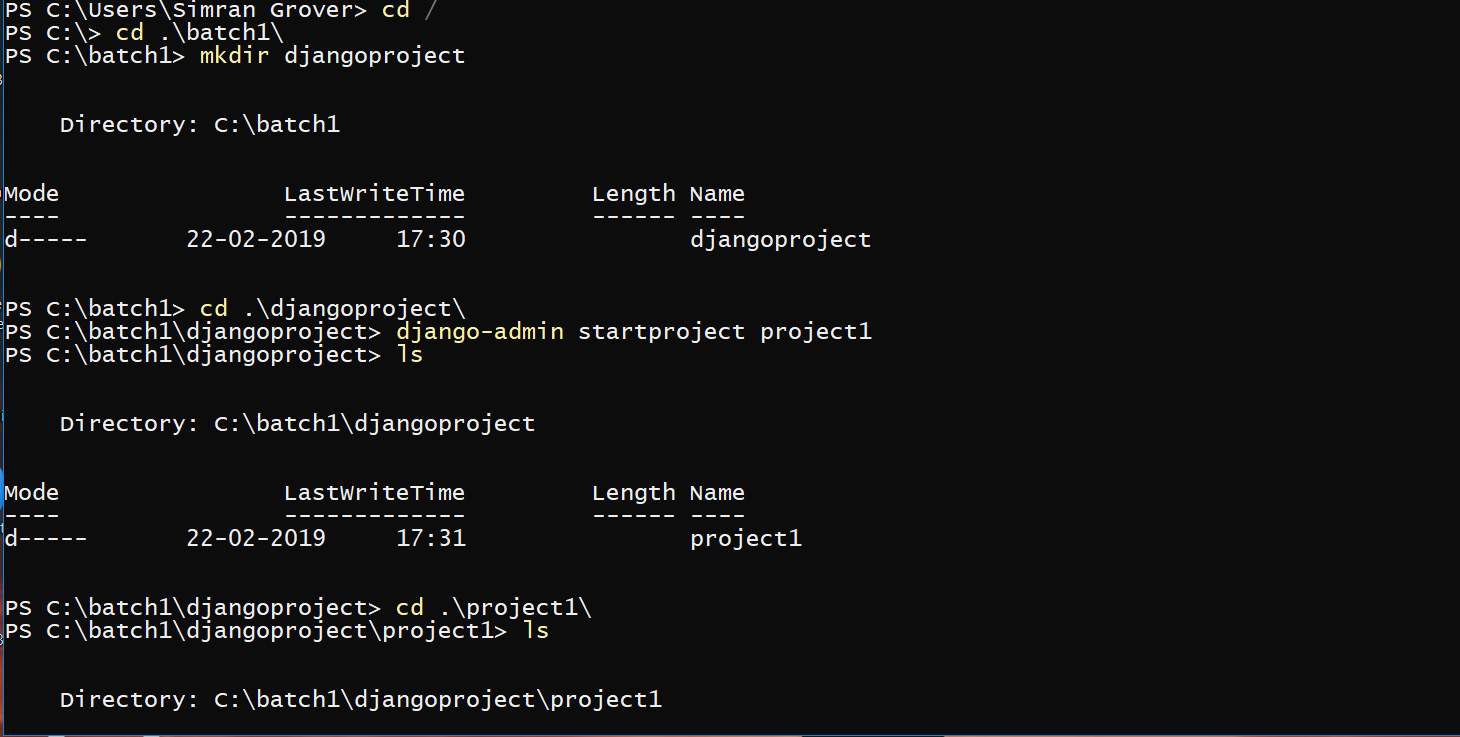
.wsgi.py

.urls.py

.\_\_init\_\_.py

.manage.py(file)

* The outer **project1/** root directory is just a container for your project. Its name doesn’t matter to Django; you can rename it to anything you like.
* **manage.py**: A command-line utility that lets you interact with this Django project in various ways.
* The inner **project1/** directory is the actual Python package for your project. Its name is the Python package name you’ll need to use to import anything inside it (e.g. **[project.urls]**).
* **Project1/\_\_init\_\_.py**: An empty file that tells Python that this directory should be considered a Python package.
* **Project1/settings.py**: Settings/configuration for this Django project. [Django settings](https://docs.djangoproject.com/en/dev/topics/settings/) will tell you all about how settings work.
* **Project1/urls.py**: The URL declarations for this Django project; a “table of contents” of your Django-powered site.
* **Project1/wsgi.py**: An entry-point for WSGI-compatible web servers to serve your project.



1. Now create static and templates folder inside the main project folder(in the above example it is ‘project1’)

.mkdir static

.mkdir templates

To have the following scenario

.project1

.project1

.settings.py

.\_\_init\_\_.py

.urls.py

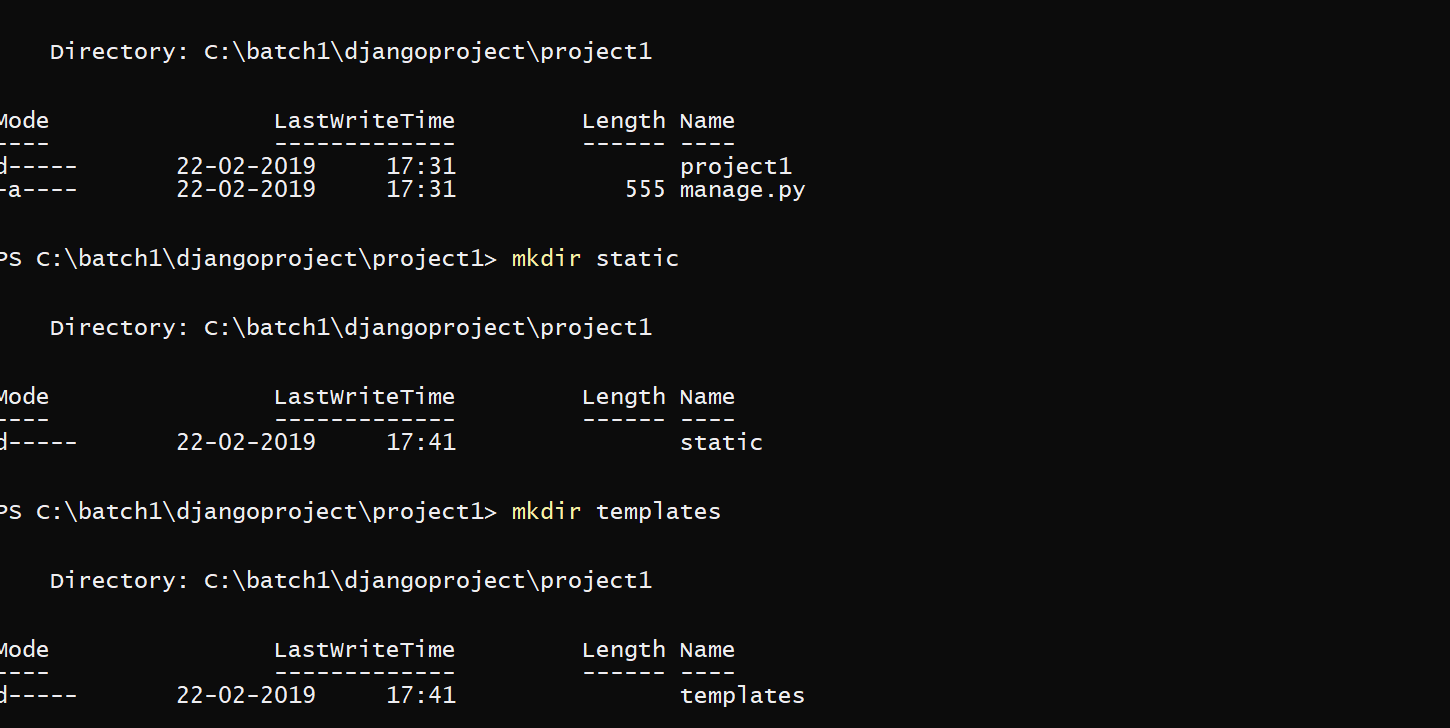
.wsgi.py

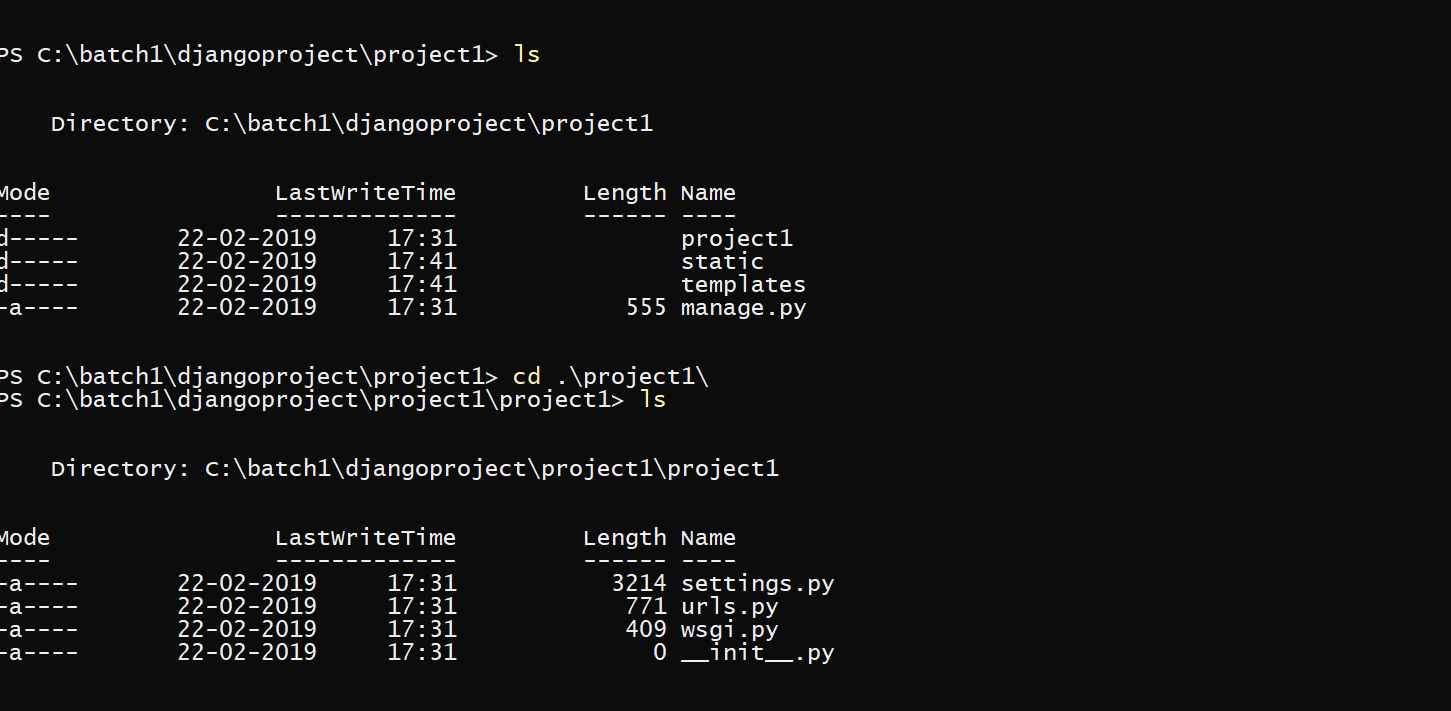
.manage.py

.static

.templates

* **Project1/static**: To store static files like css , javascripts, images etc.
* **Project1/templates**: To store templates that is html files.





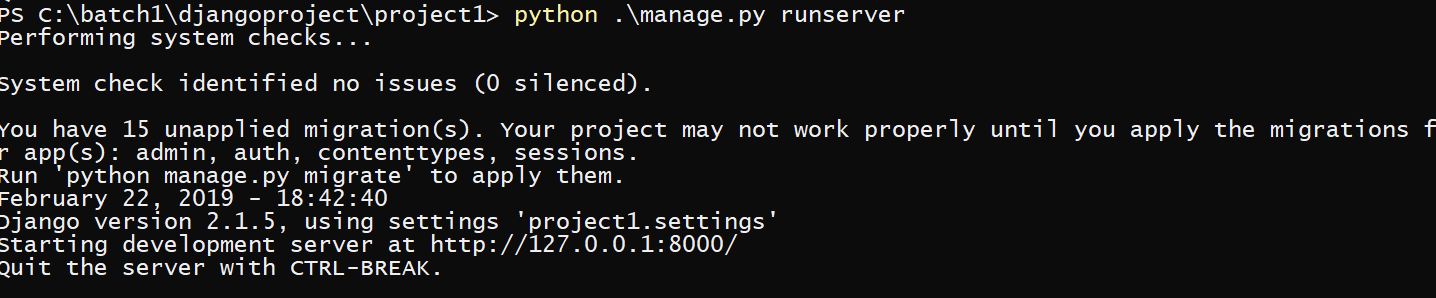
1. If your want to run your project following command is executed

(python manage.py runserver)

You’ve started the Django development server, a lightweight Web server written purely in Python. We’ve included this with Django so you can develop things rapidly, without having to deal with configuring a production server – such as Apache – until you’re ready for production.

Now’s a good time to note: **don’t** use this server in anything resembling a production environment. It’s intended only for use while developing. (We’re in the business of making Web frameworks, not Web servers.)

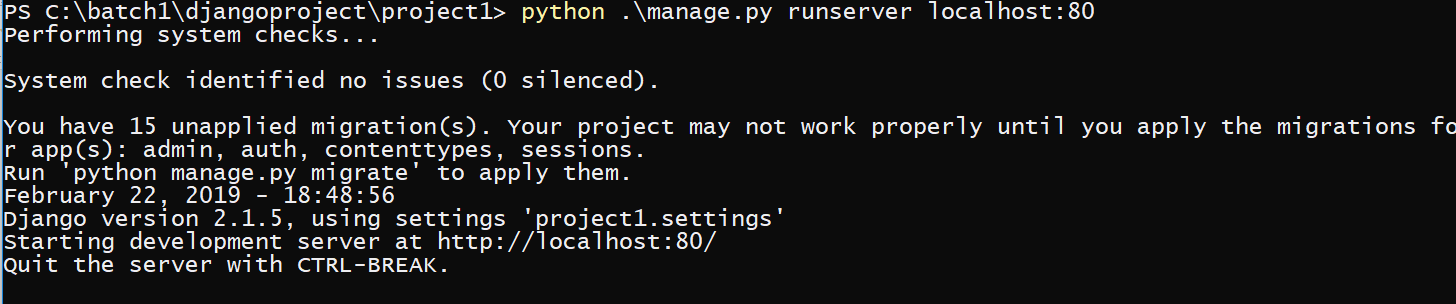
Now that the server’s running, visit <http://127.0.0.1:8000/> with your Web browser. You’ll see a “Congratulations!” page, with a rocket taking off. It worked!



(python manage.py runserver host:port)

If host is your localhost and port is 80 then following command is executed

(python manage.py runserver localhost:80)



1. Django can handle multiple apps…as our whole project cannot be handled in a single directory so we can create as much apps as needed

Difference between main project and apps is- A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects.

Django comes with a utility that automatically generates the basic directory structure of an app

To create your app, make sure you’re in the same directory as **manage.py** and type this command:

(python manage.py startapp ‘app\_name’)

If app\_name is app then following command is executed

(python manage.py startapp app1)

After executing this command the following scenario is obtained

.project1

.app1

.manage.py

.project1

.static

.templates

There will be some files generated in your app(in above example it is app1)

The files in app1 are

.app1

.\_\_init\_\_.py

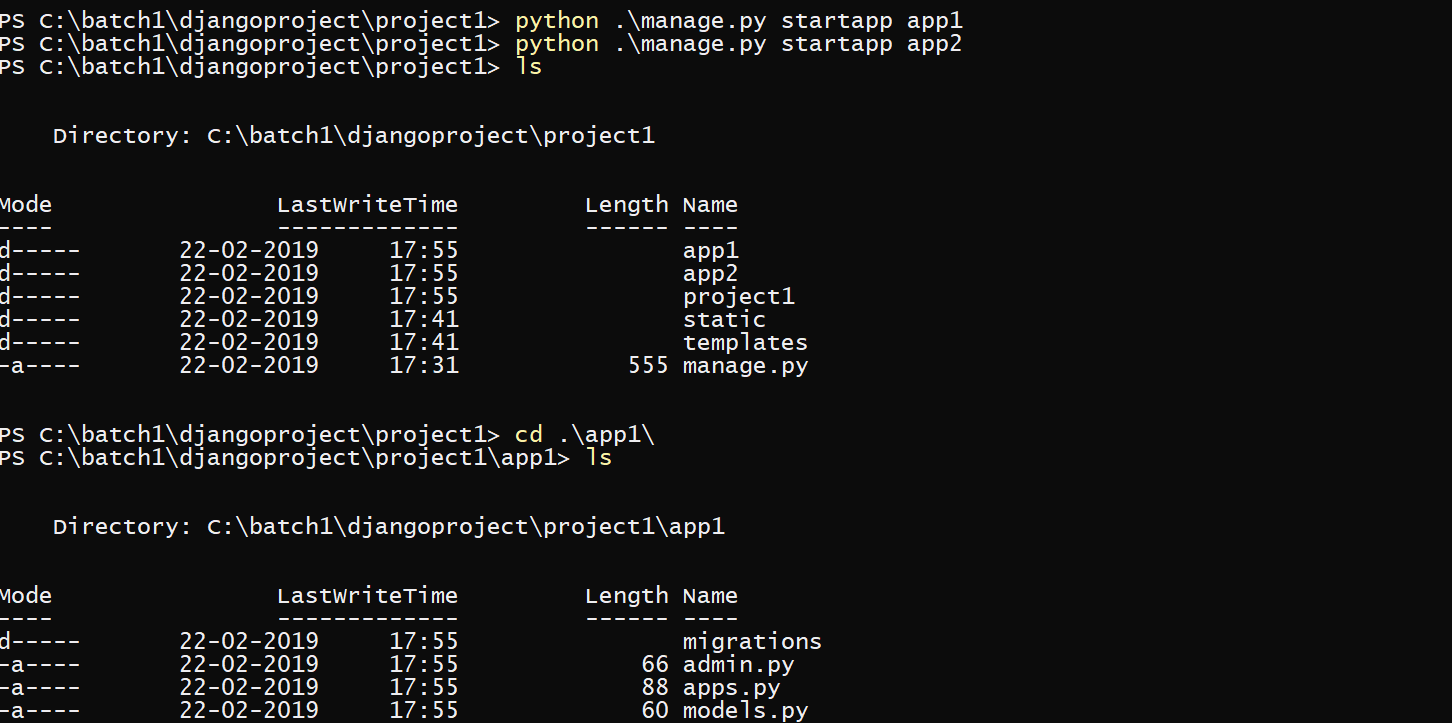
.admin.py

.apps.py

.models.py

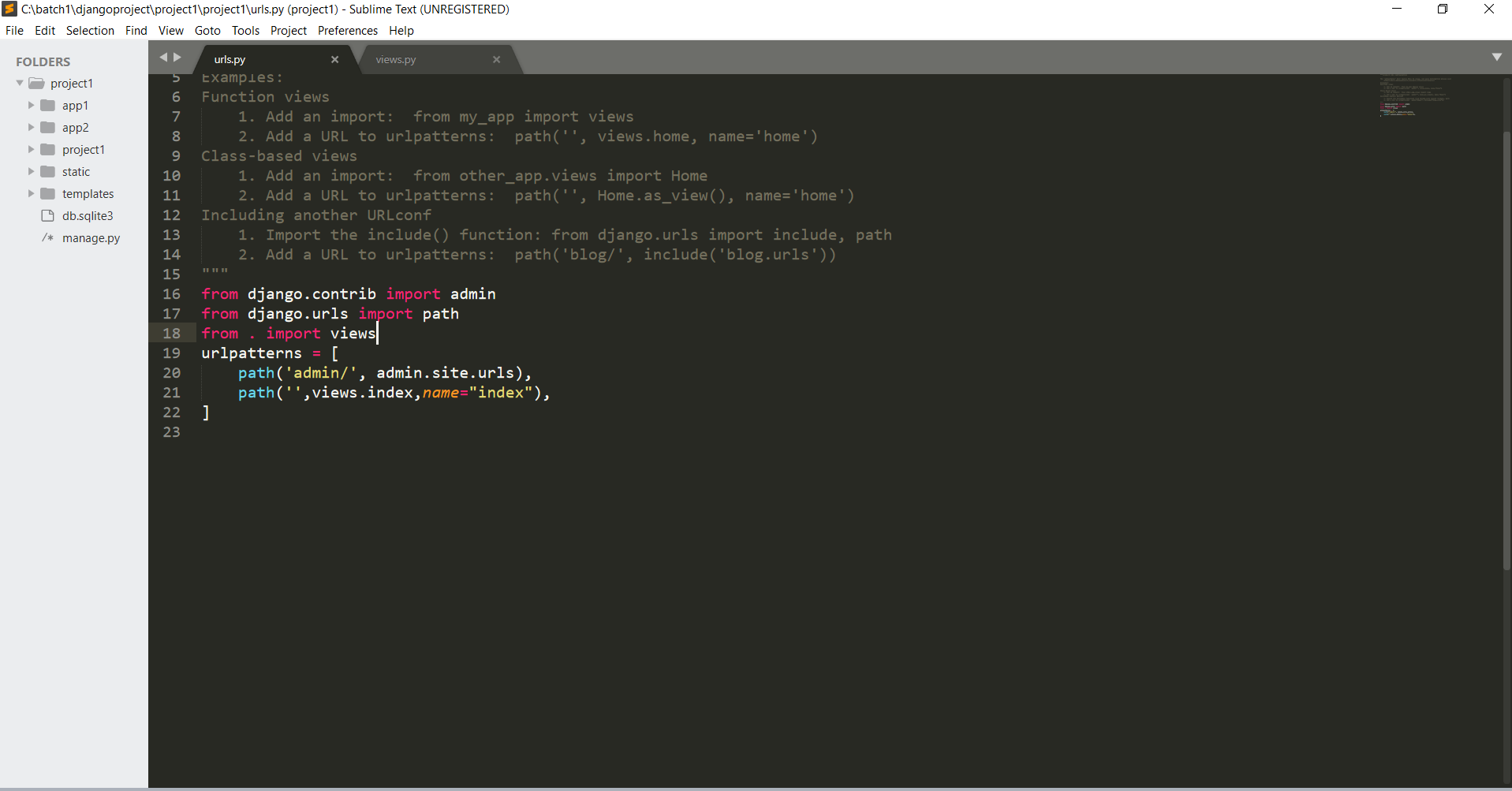
.tests.py

.views.py



1. Go to the project1 then urls.py file (project1.urls)

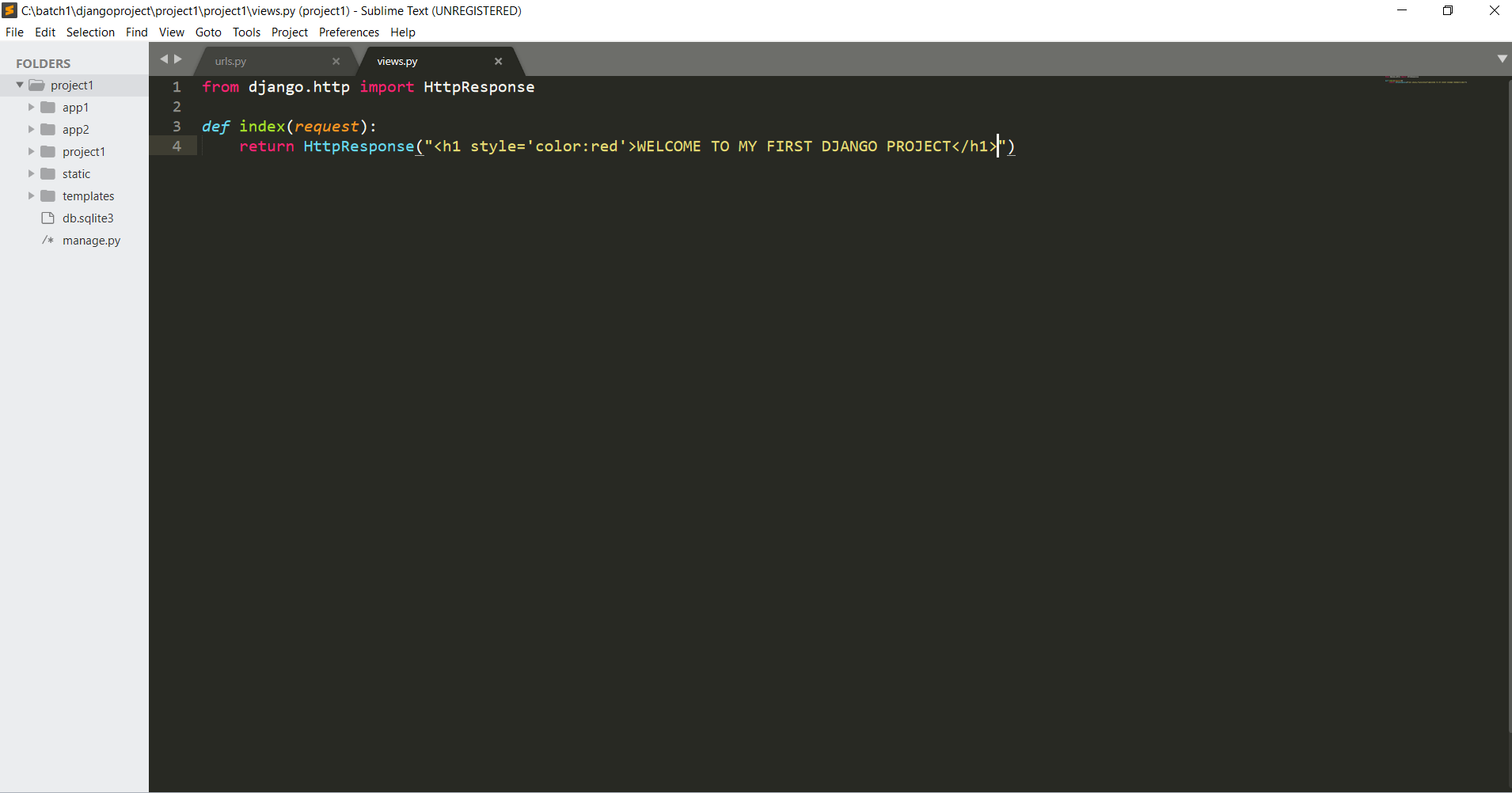
And write the url for localhost i.e (‘ ’)



### [path()](https://docs.djangoproject.com/en/dev/ref/urls/#django.urls.path) argument: route

**route** is a string that contains a URL pattern. When processing a request, Django starts at the first pattern in **urlpatterns** and makes its way down the list, comparing the requested URL against each pattern until it finds one that matches.

1. Then define the index function in project1.views file



Django uses request and response objects to pass state through the system.

When a page is requested, Django creates an **[HttpRequest](https://docs.djangoproject.com/en/2.1/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** object that contains metadata about the request. Then Django loads the appropriate view, passing the **[HttpRequest](https://docs.djangoproject.com/en/2.1/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** as the first argument to the view function. Each view is responsible for returning an **[HttpResponse](https://docs.djangoproject.com/en/2.1/ref/request-response/" \l "django.http.HttpResponse" \o "django.http.HttpResponse)** object.