**A Django Project consists of one or more Django Applications.**

**First a project is created and then one more applications are created inside that project.**

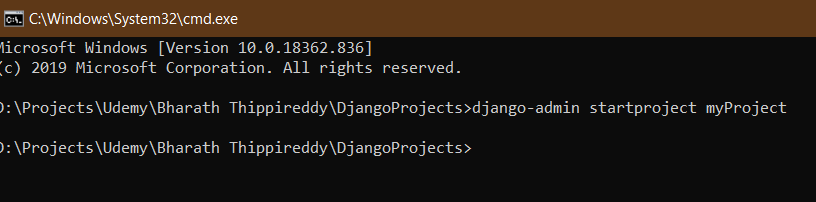
**#1. Creating our first Django Project through Command Line**

1. Create a folder any drive anywhere in the directory.

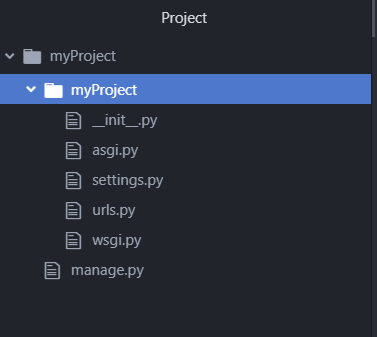
E.g. Suppose I have created a folder **DjangoProjects** in D drive.

1. Go to command line and reach inside this folder.
2. Use the command **django-admin startproject projectName**

**e.g. DjangoProjects> django-admin startproject myProject**



1. It will create a folder having name **myProject**.
2. Open this project in any code editor like **Atom or VS Code**.
3. Look at the folder structure



1. **\_\_init\_\_.py**

It tells that this folder is a python package.

1. **settings.py**

All settings for the application inside this project for the middleware & also database configuration. Later on we will create applications inside this project.

1. **urls.py**

It carries the URL’s pattern of our project as well as application.

Mapping of views created inside application to the urls is done here.

1. **wsgi.py**

Web Server Gateway Interface. It can be used to deploy our application/project on online servers or even to the cloud.

#) **manage.py**

This is very important file as it is used to run our project and to create application inside the project.

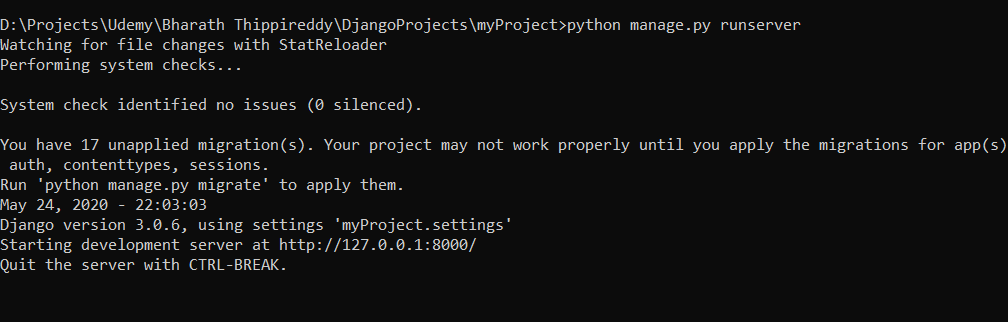
**#2. Running the Project**

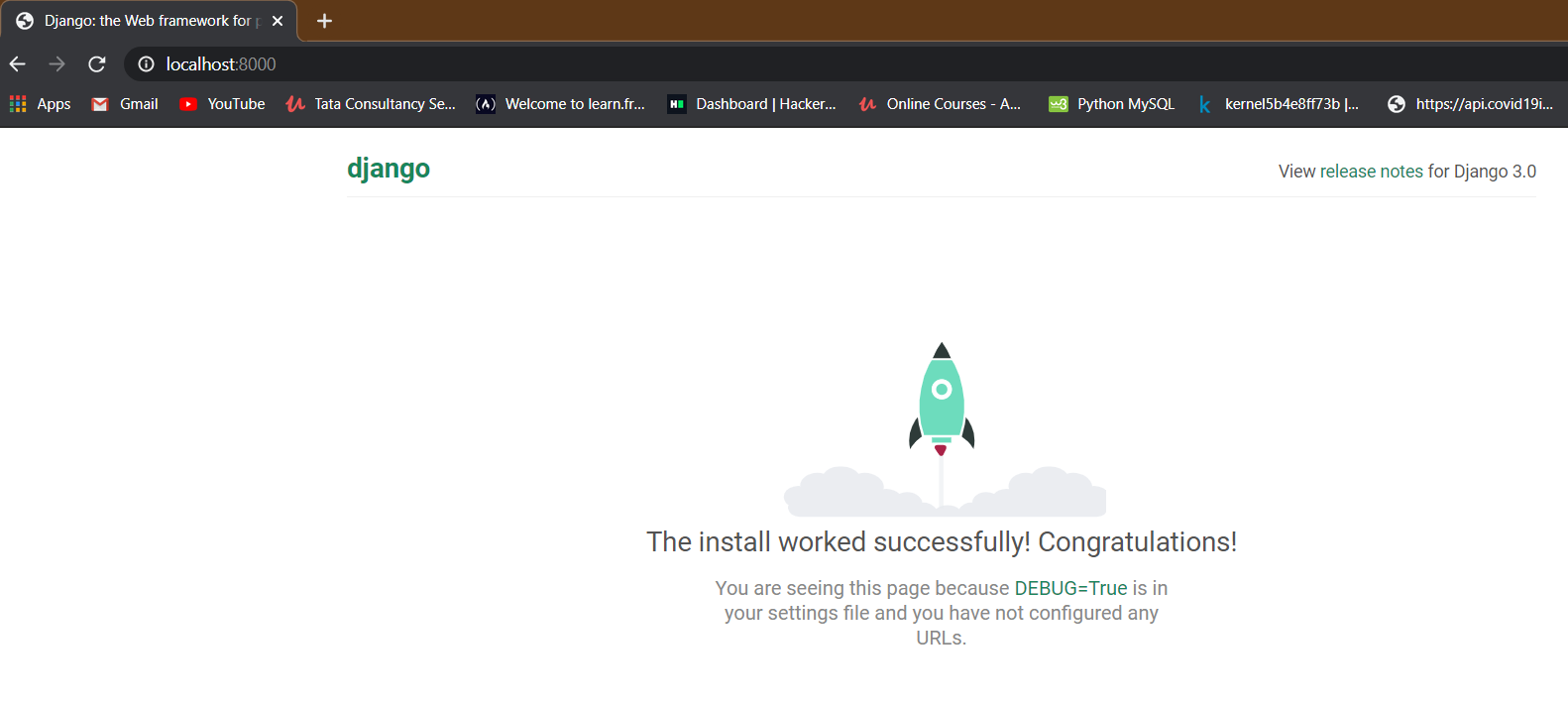
However, we have not written any codes or configured anything, still the project is ready to be run but with default output.

1. Go inside the project i.e. **myProject**.
2. Execute the command

**myProject**> **python** **manage**.**py** **runserver**

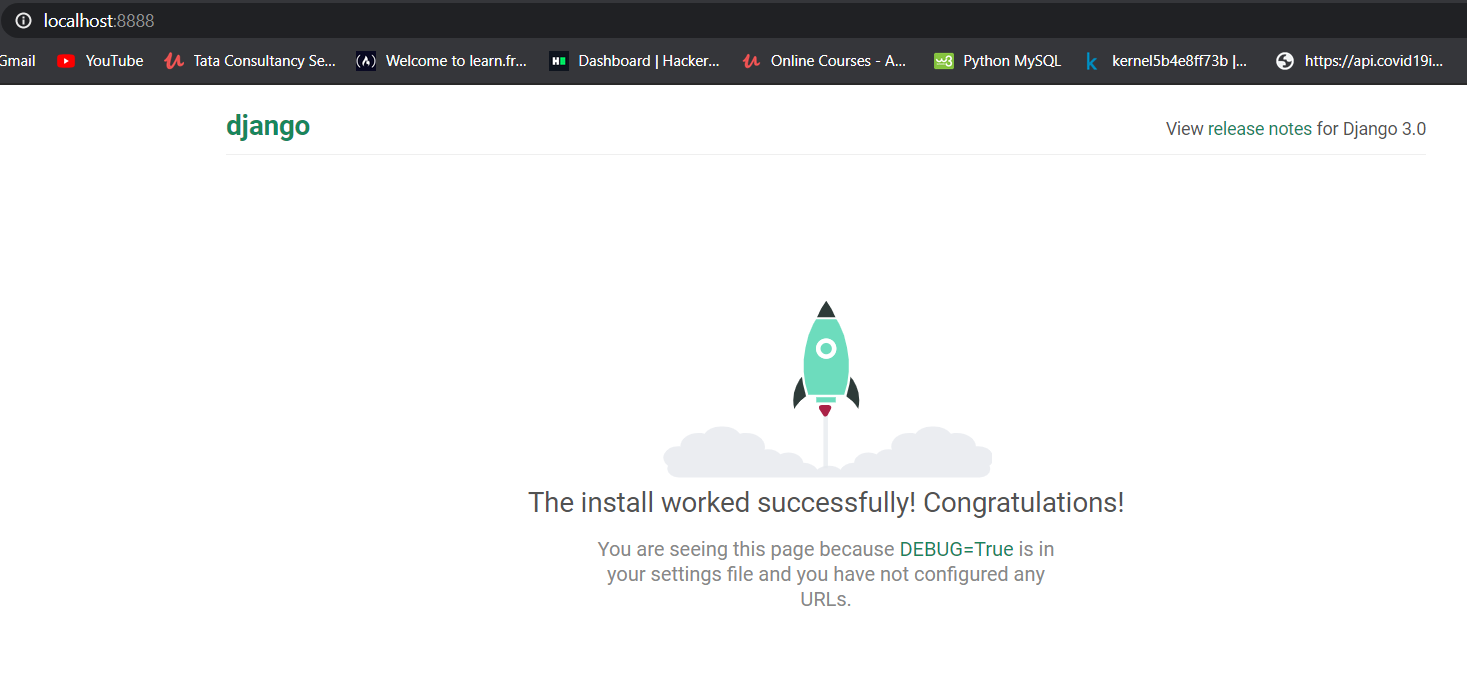
1. As python or django internally provides a default web server for us and it will launch the project on the server (**localhost**:**8000**).





1. By default, it will be launched on **port** **8000**, but it can be **changed**.
2. Changing port number
3. Stop the server (ctrl + c)
4. Use the as usual command to run the project but just put the required port number at the last.

E.g. **myProject**>**python manage.py runserver 8888.**



**In Brief so far**

**Creating Project.**

**DjangoProjects> django-admin startproject demoProject**

**Running the Project.**

1. **Get inside the project and run the command**

**demoProject>python manage.py runserver**

1. **To run the project on demanded port**

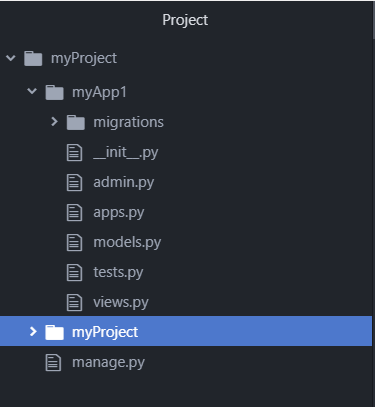
**demoProject>python manage.py runserver 7777**

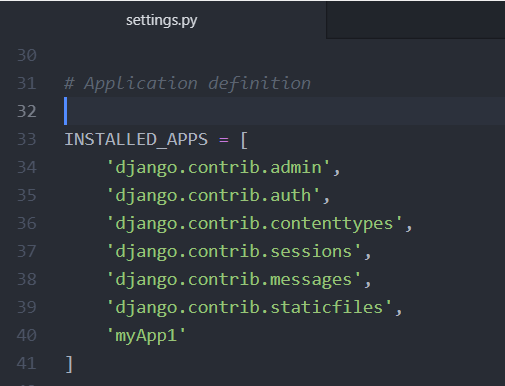
**#3. Creating application(s) inside the Project**

1. Go inside the project and run the command

**myProject**>**python** manage.py **startapp** **myApp1**

1. A folder myApp1 will be created inside the myProject
2. Look at the folder structure.



1. As we have created a new app so it needs to mentioned in the **settings**.py of the **myProject** file.
2. **Go to settings.py**
3. **Go to INSTALLED\_APPS**
4. Add the name of the newly created app i.e. **myApp1** to the list at last.
5. We can create as many app as we need, just make sure to mention it in settings.py **Installed\_Apps** list here.

**#4. Creating Views inside the apps**

Views mean the output or response we should get on hitting a url from the browser.

**There are two ways in which we can create views.**

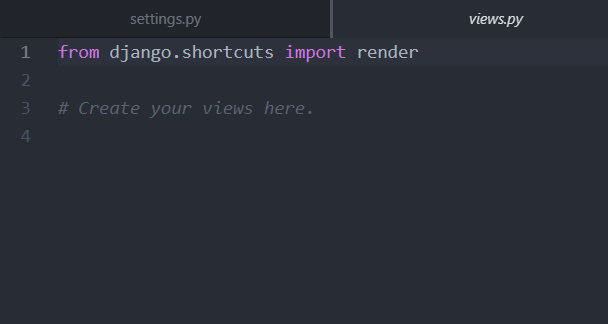
1. Function Based Views
2. Class Based Views

**# Function Based Views**

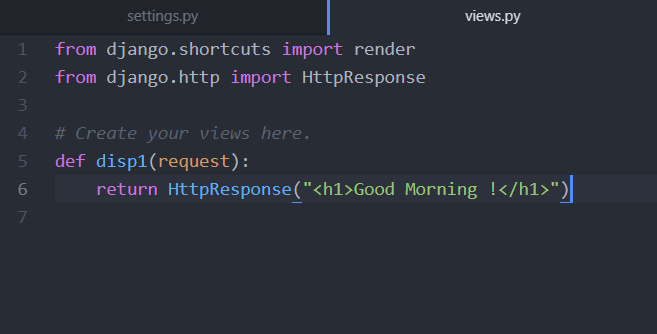
**Step1 : Creating the view**

A view is responsible for taking a request from the browser and sending a response back to the browser.

We have only an app i.e. myApp1, so view will be created inside that app in views.py file.



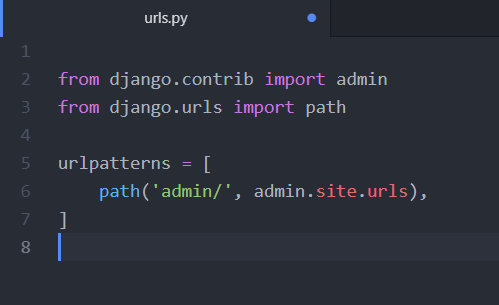
Here’s the code,



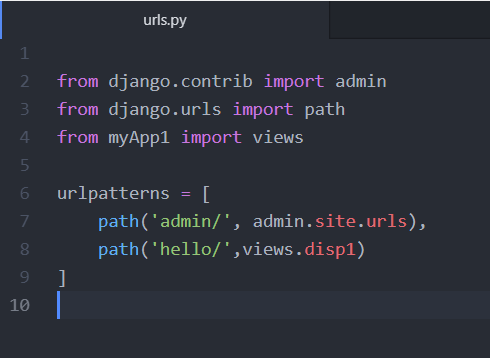
**Step 2 : Mapping a url to this views’s method(s)**

We have to map this view to a unique url, so that using that url this view(method) can be invoked from the browser.

It will be done in the **urls**.**py** file of the myProject.



So mapping code,



* First import that **views** in that file.
* Then add the **urlpatterns** as you can see.
* So, when a user hits the URL **localhost:8000/hello**, then due to mapping **disp1**() will be invoked and whatever it returns will be displayed on the browser.

