**Django Application:**

* An app in Django is a sub-module of a project, and it is used to implement some functionality(business logic).
* We can create multiple apps within a single Django project. And these apps can be independent of one another.

**Note:** The difference between a project and app is, a project is a collection of configuration files and apps whereas the app is a web application which is written to perform business logic.

**Benefits of using Django apps –**

* Django apps are reusable i.e. a Django app can be used with multiple projects.
* loosely coupled i.e. almost independent components
* Multiple developers can work on different components.
* Debugging and code organization is easy. Django has an excellent debugger tool.
* It has in-built features like admin pages, etc, which reduces the effort of building the same from scratch.

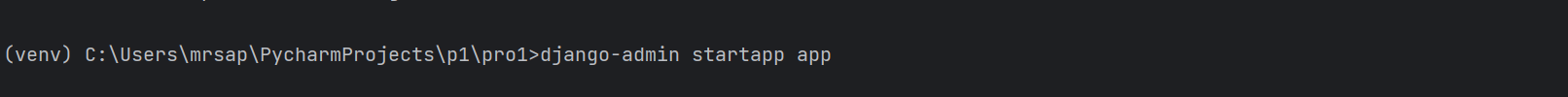
**App creation in Django :**

**Step 1:- Create django app**

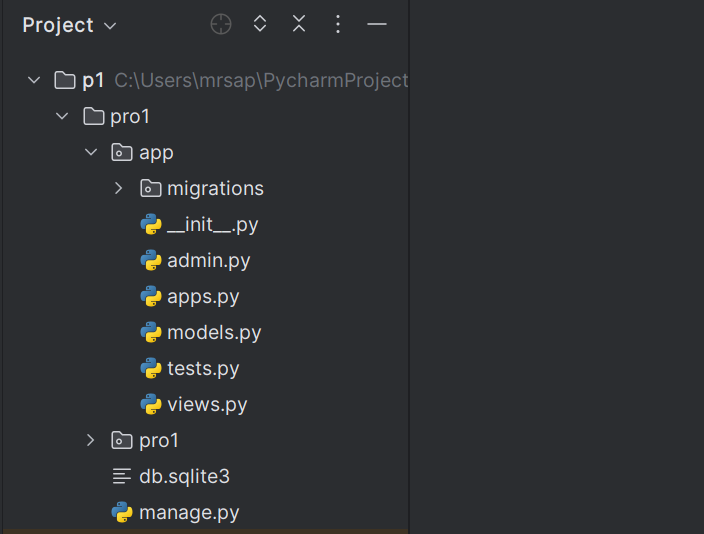
To create a basic app in your Django project you need to go to the directory containing manage.py and from there enter the command.

**python manage.py startapp projectApp**

**Django-admin startapp appname**



**Folder Structure:**



* **\_\_init\_\_**.py a special Python file that is used to indicate that the directory it is present in is a Python package.
* **admin.py** – This file is used to display your models in the Django admin panel. You can also customize your admin panel.
* **models.py** − This is where all the application models are stored.
* **tests.py** − This file is use for unit tests
* **views.py** − This file contain view.

**What is view?**

 A view is a place where we put our business logic of the application. View are Python functions or classes that receive a web request and return as a web response. The response can be a simple HTTP response, an HTML template response, or an HTTP redirect response that redirects a user to another page. Views hold the logic that is required to return information as a response in whatever form to the user.

There are 2 types of view in Django

1.Function Base View

2.Class Base View

1. **Function-Based Views**

Function-based views are good for beginners. It is very easy to understand in comparison to class-based views. function-based views are easy to understand but due to the code redundancy in a large Django project, you will find similar kinds of functions in the views.

**Pros:**

* Easy to read, understand and implement.
* Explicit code flow
* Straightforward usage of decorators.
* Good for the specialized functionality.

**Cons:**

* Code redundancy and hard to extend.
* Conditional branching will be used to handle HTTP methods.

**Step 2:-** **Define app level View**

Inside views.py --> PROJNAME-->APPNAME-->views.py

from django.http import HttpResponse  
  
# Create your views here.  
  
def view1(request):  
 return HttpResponse("This is My 1st View")

def view2(request):  
 return HttpResponse("This is My 2nd View")

* First, we import the class HttpResponse from the django.http module.
* Next, we define a function called view1. 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.

**2.Class-Based View**

* Class-based views are the alternatives of function-based views.
* Class-based views don’t replace function-based views, but they do have certain advantages over function-based views.

**Pros:**

* The most significant advantage of the class-based view is inheritance. In the class-based view, you can inherit another class, and it can be modified for the different use cases.
* Code reusability is possible in class-based views.
* Another advantage of using a class-based view is code structuring. In class-based views, you can use different class instance methods (eg. get, post) (instead of conditional branching statements inside function-based views) to generate different HTTP requests.

**Cons:**

* Complex to implement and harder to read

from django.http import HttpResponse  
from django.views import View  
  
  
class MyCreateView(View):  
  
 def get(self, request):  
 return HttpResponse("My first Class based view")

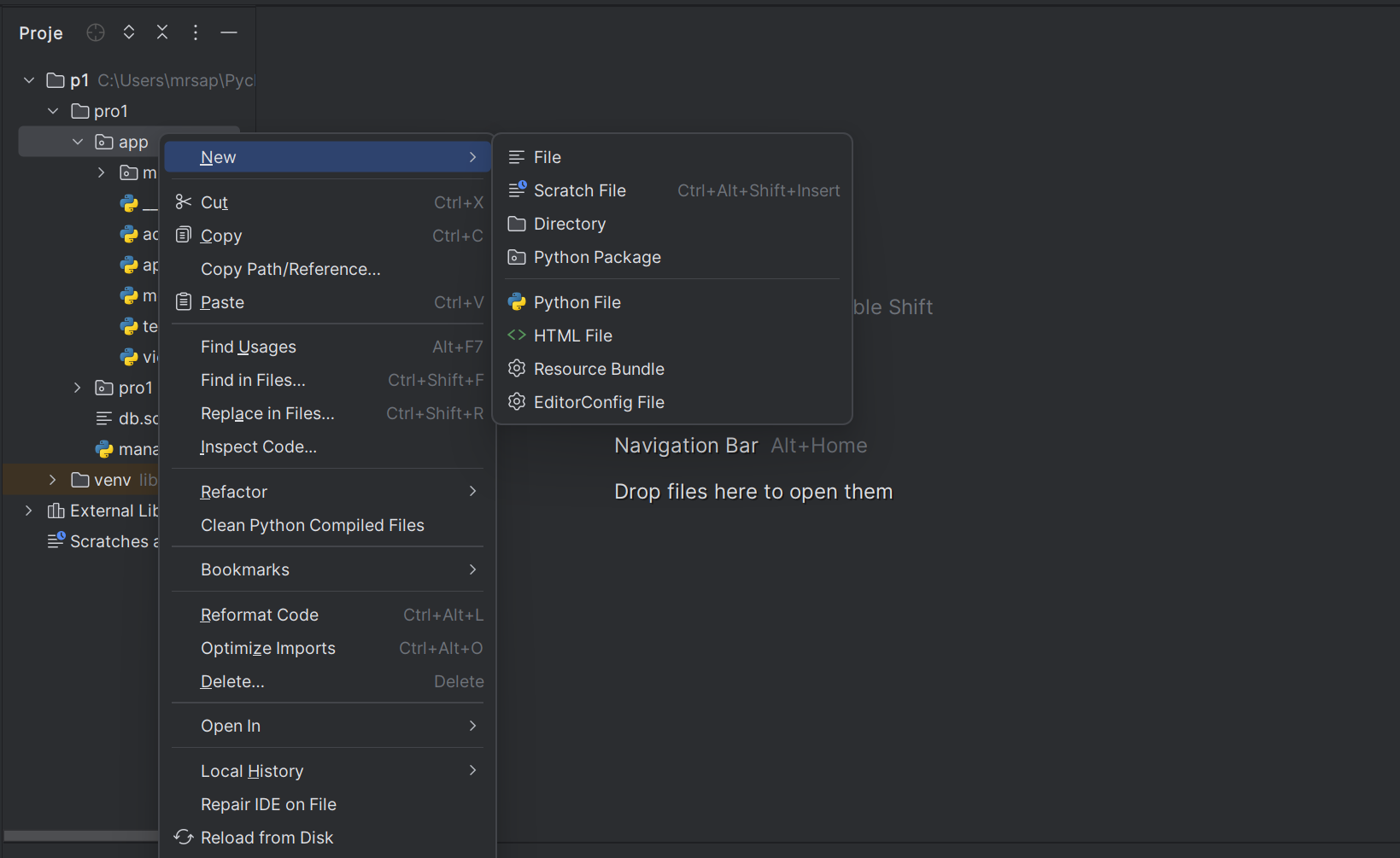
**What is url?**

* In Django, views are Python functions which take a URL request as parameter and return an HTTP response or throw an exception like 404. Each view needs to be mapped to a corresponding URL pattern.
* Django runs through each URL pattern, in order, and stops at the first one that matches the requested URL, matching against path\_info.

**Step 3:- Define App\_level url**

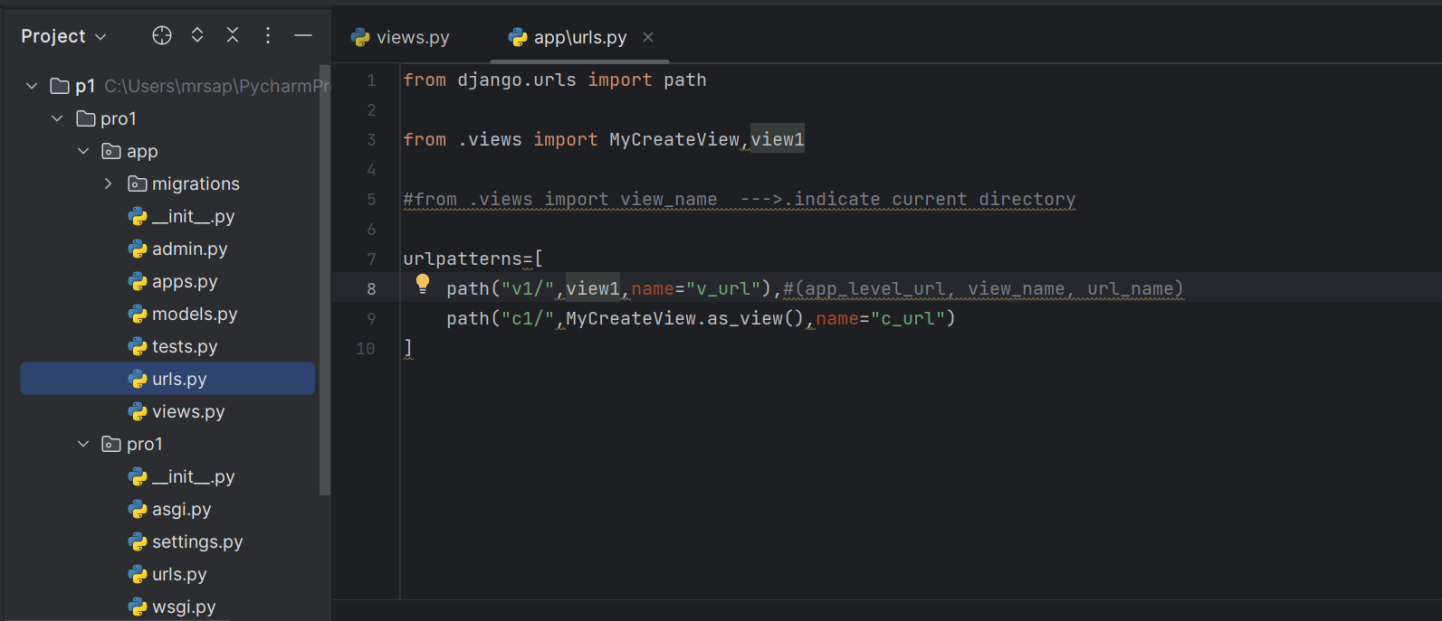
And we have to explicitly create urls.py in our app.

Right click on app create new python file name it **urls.py** (in lowercase)



**Create new urls.py 🡪 PROJNAME 🡪 APPNAME 🡪 urls.py (AppLevel urls)**

**Urls.py**



**Note:** First import View class from Django.views module.

Name of list must be **urlpatterns** (in lowercase).

we are passing 3 arguments here (app\_level\_url, view\_name, url\_name)

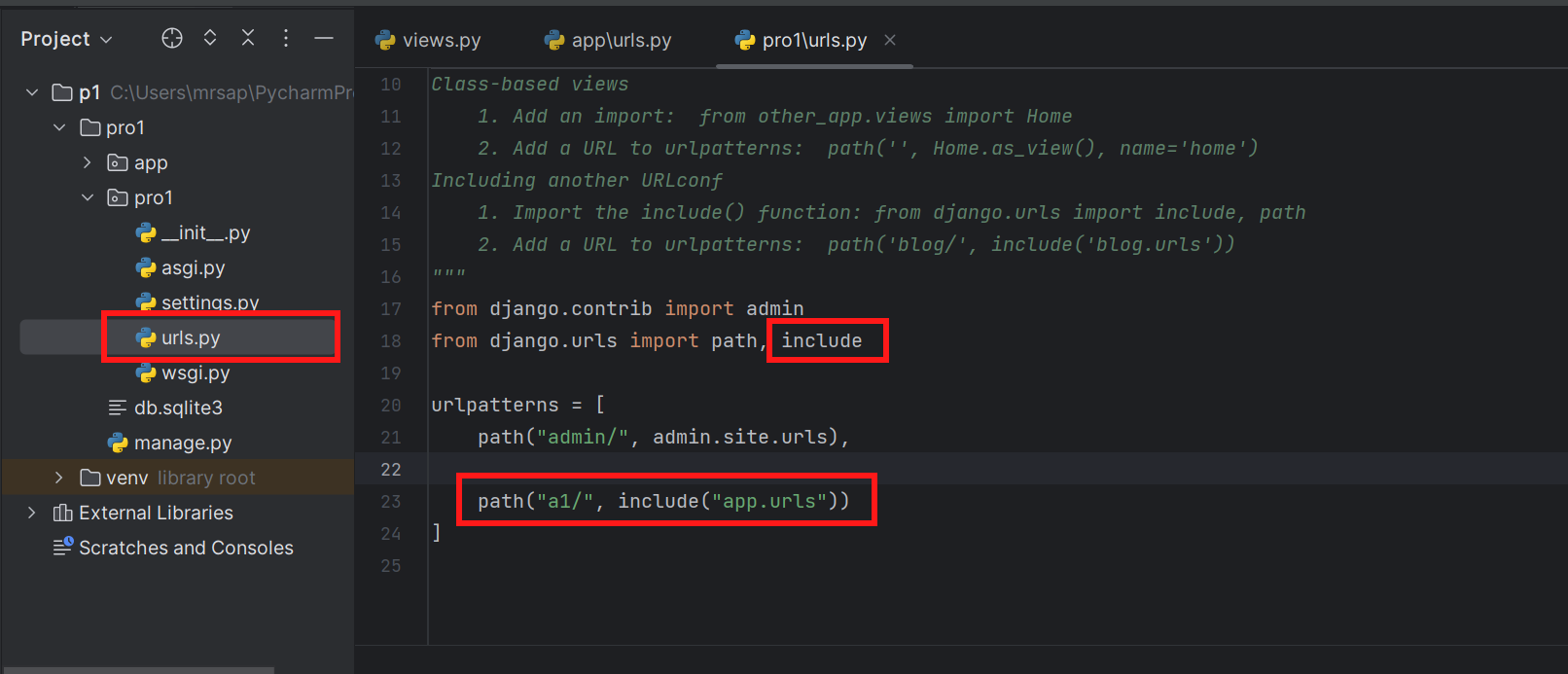
in case of class based views we have to add suffix(**.as\_view()**) className.as\_view().

**Step 4:- Register app\_url into project\_url**

**Define an urlpattern(path) --> PROJNAME-->PROJNAME-->urls.py**

Steps:-

* Let's open the file **urls.py** of the project. There is default file named urls.py in project.
* we import the **include** function from the **django.urls** and map the path of the **a1** to the **app.urls.**



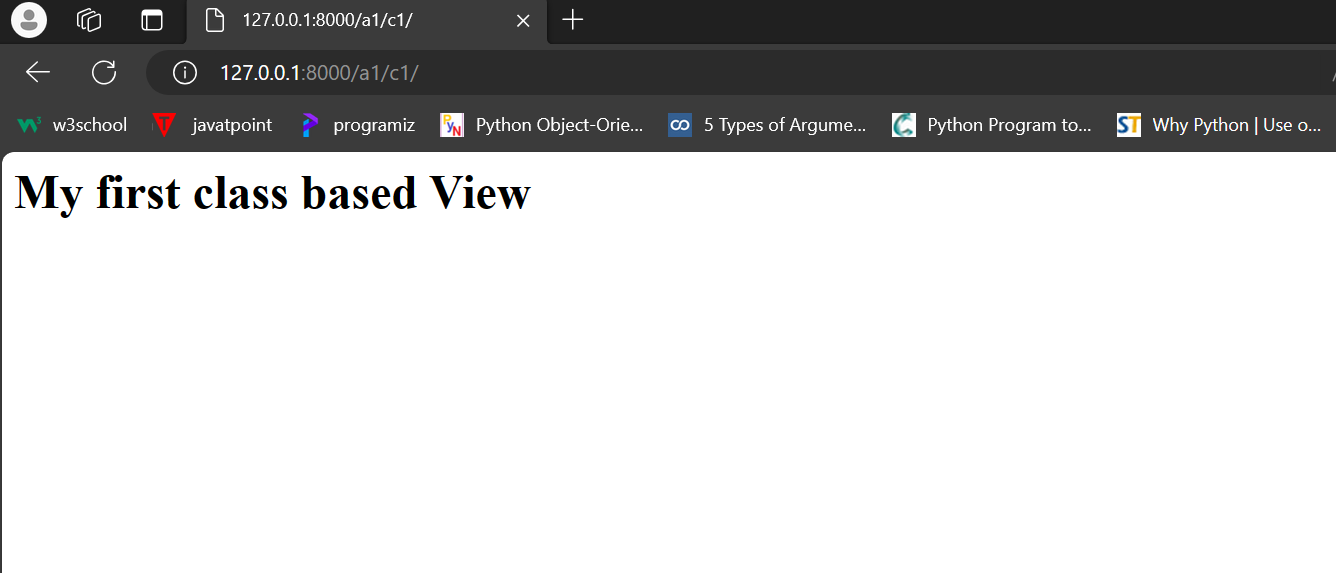
**Step 5:- Run Server**

**Py manage.py runserver**

Now, start the server and enter **localhost:8000/a1/v1** to the browser. This URL will be mapped into the list of URLs and then call the corresponding function from the views file.

In this example, v1 will be mapped and call view1 function from the views file. It is called **URL mapping**.

* First, the web browser sends an HTTP request to the URL http://127.0.0.1:8000/a1/
* Second, Django executes the urls.py in the django\_project directory. It matches the a1/ with the URL in the urlpatterns list in the urls.py. As a result, it sends '' to the urls.py of the app application.
* Third, Django runs the urls.py file in the app application. It matches the '' URL with the views.view1 function and execute it, which returns an HTTP response that outputs a h1 tag.
* Finally, Django returns a webpage to the web browser.



A screenshot of a computer

Description automatically generated