Creating an MVC (Model-View-Controller) project in Django is straightforward since Django itself follows the MVC architectural pattern. In Django, the term commonly used is MTV (Model-Template-View), which aligns with the same concepts as MVC. Here are step-by-step instructions to create a basic Django project:

### Step 1: Install Django

If you haven't installed Django yet, you can do so using the following command:

```bash

pip install Django

```

### Step 2: Create a Django Project

Run the following command to create a new Django project. Replace "myproject" with your preferred project name.

```bash

django-admin startproject myproject

```

### Step 3: Navigate to the Project Directory

Move into the project directory using the following command:

```bash

cd myproject

```

### Step 4: Create a Django App

Django projects are divided into apps. Create a new app with the following command. Replace "myapp" with your preferred app name.

```bash

python manage.py startapp myapp

```

### Step 5: Define Models

Open the `models.py` file in your app directory (`myapp/models.py`) and define your models. Models represent the data structure of your application.

```python

# myapp/models.py

from django.db import models

class MyModel(models.Model):

name = models.CharField(max\_length=255)

description = models.TextField()

def \_\_str\_\_(self):

return self.name

```

### Step 6: Create Migrations

Run the following commands to create and apply database migrations:

```bash

python manage.py makemigrations

python manage.py migrate

```

### Step 7: Create Views

Open the `views.py` file in your app directory (`myapp/views.py`) and define your views. Views handle the business logic of your application.

```python

# myapp/views.py

from django.shortcuts import render

from .models import MyModel

def index(request):

my\_objects = MyModel.objects.all()

return render(request, 'index.html', {'my\_objects': my\_objects})

```

### Step 8: Create Templates

Create a `templates` directory inside your app directory (`myapp/templates`). Inside the `templates` directory, create an `index.html` file to display your data.

```html

<!-- myapp/templates/index.html -->

<!DOCTYPE html>

<html>

<head>

<title>My MVC Project</title>

</head>

<body>

<h1>Welcome to my MVC project</h1>

<ul>

{% for obj in my\_objects %}

<li>{{ obj.name }} - {{ obj.description }}</li>

{% endfor %}

</ul>

</body>

</html>

```

### Step 9: Define URLs

Create a `urls.py` file in your app directory (`myapp/urls.py`) to define the URL patterns.

```python

# myapp/urls.py

from django.urls import path

from .views import index

urlpatterns = [

path('', index, name='index'),

]

```

### Step 10: Include App URLs in Project URLs

Open the `urls.py` file in your project directory (`myproject/urls.py`) and include the app's URLs.

```python

# myproject/urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('myapp.urls')),

]

```

### Step 11: Run the Development Server

Run the development server with the following command:

```bash

python manage.py runserver

```

Visit `http://127.0.0.1:8000/` in your web browser to see your MVC (MTV) project in action.

This set of steps creates a basic Django project with a simple app, model, views, templates, and URL configuration. From here, you can continue to expand and enhance your project by adding more features, views, and templates.