**How to Create a Django Project: Step-by-Step Guide with Explanation**

---

## 1. Install Django

Django is not included with Python by default, so you need to install it using pip.

```
pip install django
```

- `pip` : Python package installer.
- `install` : Command to install packages.
- `django` : The name of the package to install.

To verify the installation:

```
django-admin --version
```

- `django-admin` : A command-line tool that comes with Django for creating and managing projects.
- `--version` : Displays the installed Django version.

---

## 2. Create a Django Project

```
django-admin startproject myproject
```

- `startproject` : Tells Django to create a new project.
- `myproject` : The name of your project folder.

This command creates a directory with this structure:

```
myproject/
    manage.py
    myproject/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

- `manage.py` : A script to interact with your project (e.g., run server, apply migrations).
- `__init__.py` : Marks the directory as a Python package.
- `settings.py` : Holds all settings and configuration for the project.

- `urls.py`: Defines the routing system, mapping URLs to views.
- `asgi.py`: Entry point for ASGI (Asynchronous Server Gateway Interface).
- `wsgi.py`: Entry point for WSGI (Web Server Gateway Interface).

---

### 3. Navigate into the Project Folder

```
cd myproject
```

- `cd`: Change directory to enter the project folder.

---

### 4. Run the Development Server

```
python manage.py runserver
```

- `python`: Runs Python interpreter.
- `manage.py`: Used to run commands related to Django.
- `runserver`: Starts a lightweight web server for development and testing.

Visit `http://127.0.0.1:8000/` in your browser to view the default welcome page.

---

### 5. Create a Django App

```
python manage.py startapp myapp
```

- `startapp`: Command to create a new app inside the project.
- `myapp`: Name of the app (you can use any valid Python name).

This creates:

```
myapp/
    admin.py
    apps.py
    models.py
    tests.py
    views.py
    migrations/
    __init__.py
```

- `admin.py`: File to register models for the admin interface.
- `apps.py`: App-specific configuration.

- `models.py` : Where database models (tables) are defined.
- `tests.py` : Used to write automated tests.
- `views.py` : Contains logic for handling requests and returning responses.
- `migrations/` : Stores database migration files.
- `__init__.py` : Makes this a Python package.

---

### 6. Register the App in settings.py

Open `myproject/settings.py` and add the app:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    ...
    'myapp',
]
```

- `INSTALLED_APPS` : A list of all apps that are activated in this Django project.
- `myapp` : The app we just created.

---

### 7. Create Views and URLs

**views.py**

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello, Django!")
```

- `HttpResponse` : A class that returns content to the browser.
- `home` : A function-based view that responds to a request.
- `request` : Contains metadata about the HTTP request.

**urls.py (create in app folder)**

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

urlpatterns = [
```

```
    path('', views.home, name='home'),
]
```

- `path`: Used to define URL patterns.
- `''`: The root URL.
- `views.home`: Links the URL to the `home` view function.
- `name='home'`: Names the URL pattern.

**Update main urls.py in project folder**

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

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

- `include`: Allows referencing another URL configuration.
- `admin.site.urls`: Adds the admin interface at `/admin/`.

---

### 8. Run the Server Again

```
python manage.py runserver
```

Navigate to `http://127.0.0.1:8000/` to see the message "Hello, Django!"

---

### You're Ready!

You've now created and connected a Django project and app. Each term and command above is essential for setting up and expanding your Django development process.

Next steps:

- Add models and use Django's ORM.
- Create HTML templates.
- Handle forms and user input.
- Explore Django's admin, authentication, and middleware.