Django —

DAY -01 NOTES

1. What is Django?

Django is a high-level Python web framework used to build websites, web apps, APIs, dashboards, and backend systems.
It helps developers move fast by giving pre-built components like:
URL routing
Views (business logic)
Templates
Database connection
Admin panel
Authentication system
Security features
Django follows the MVT architecture → Model, View, Template.

🎯 2. Why Do We Use Django?
Django is used because it makes web development:
✔ Faster
You don't write everything from scratch — routing, forms, admin, security are already built.
✔ Secure
Django prevents common attacks like SQL Injection, XSS, CSRF, etc.
✔ Scalable
Used by Instagram, Pinterest, NASA, Mozilla, Spotify.
✔ Organized
Follows clean structure so your project stays maintainable.
✔ Beginner-Friendly
Simple setup, Python-based, and easy to learn.

📌 3. When Do We Use Django?
Use Django when you need:
✔ Large or medium-sized project
(College ERP, e-commerce, dashboards, banking apps)
✔ Apps with authentication
(Login, signup, user roles)
✔ Fast backend + database integration
(Create/Read/Update/Delete data)

✔ Admin panel
(Automatically generated at /admin/)

✔ APIs
(Django REST Framework)
Not preferred for:

Simple static websites → use HTML/CSS only
Real-time apps → use Django Channels or Node.js

⚙️ 4. How Django Works (Technical Flow)
Django follows a request → process → response flow.

Let's break it:

Browser sends request
Example: http://127.0.0.1:8000/

URL Router (urls.py)
Django checks which function should run for this URL.

Views (views.py)
The matching function executes — this is your logic.
Example: return text, fetch data, read DB, etc.

Template (HTML file)
View renders the HTML page.

Response sent back
Django sends the final HTML to the browser.

So the flow is:
Browser → URLs → View → Template → Response
This is why Django is clean and organized.

🛠️ 5. How to Use Django (Step-by-Step Technical Explanation)
1. Create a virtual environment
Used to isolate project libraries.

2. Install Django
Adds Django framework to the environment.

3. Create Django project
Generates base structure:

settings
URLs
WSGI/ASGI
core config

4. Create an app

Each part of your website (login system, blog, shop) is an app.
5. Create a view
This is Python function that returns output.

6. Create URL mapping
Connects URL → view function.

7. Create template
HTML output displayed to users.

🌐 6. Where Django Is Used?

Django is used almost everywhere where backend logic is needed.
✔ Social Media Platforms
Instagram, Pinterest
✔ Ed-Tech & E-commerce
Udemy, Coursera, Shopify-like platforms
✔ Banking & Enterprise Apps
Secure apps needing authentication
✔ Data Science Dashboard

ML/DL model deployment
IoT dashboards
Admin dashboards
✔ Government Systems
Online applications, portals, ticket systems
Basically:
"If it needs login, database, UI + backend — Django fits perfectly."

full minimal setup to print Hello World

. I'll show **two tiny options**:
A) raw `HttpResponse` (fast)
B) rendering an HTML template (recommended)

---

0 — Prep (create project folder & venv)

```
# make a folder (optional)
mkdir django-hello && cd django-hello

# create virtualenv
python -m venv venv

# windows (cmd):
venv\Scripts\activate

# windows (PowerShell):
```

```
venv\Scripts\Activate.ps1
```

---

1 — Install Django

```
pip install django
```

---

2 — Start a Django project

```
django-admin startproject myproject
cd myproject
```

You now have:

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

---

3 — Create an app

```
python manage.py startapp main
```

o/p like : Now:

```
myproject/
  main/
    views.py
    models.py
    apps.py
    ...
```

---

4 — Register the app in settings

Open `myproject/settings.py` and add `'main',` to `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    ...
    "main",   # add this
]
```

5A — Option A: Fast Hello (HttpResponse)

Edit `main/views.py`:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello World 👋")
```

Create `main/urls.py` (new file):

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

urlpatterns = [
    path("", home, name="home"),
]
```

Edit project `myproject/urls.py` to include app routes:

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

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("main.urls")),   # routes from main app
]
```

Run the server:

```
python manage.py runserver
```

Open: `http://127.0.0.1:8000/` → should show **Hello World** 👋.

5B — Option B: HTML Template (recommended)

This shows using HTML tags.

a) Make templates folder

From project root (where `manage.py` is):

```
mkdir templates
```

Create `templates/home.html` with this content:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello</title>
</head>
<body>
  <h1>Hello World 👋</h1>
  <p>Rendered via Django template.</p>
</body>
</html>
```

b) Tell Django about templates

Open `myproject/settings.py` and ensure inside `TEMPLATES`:

```
TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [ BASE_DIR / "templates" ],   # <- ensure this
exists
        'APP_DIRS': True,
        'OPTIONS': {...},
    },
]
```

c) Update view to render template

Edit `main/views.py`:

```
from django.shortcuts import render

def home(request):
    return render(request, "home.html")
```

URLs same as in Option A (`main/urls.py`, and include in project `urls.py`).

Run server:

```
python manage.py runserver
```

Open: `http://127.0.0.1:8000/` → you'll see the HTML page.

---

7 — Useful extra commands

Make DB migrations (even if you haven't models):

```
python manage.py migrate
```

-

Create admin superuser:

```
python manage.py createsuperuser
```

TASK

Part 1: Environment Setup

1. Install Python (latest version).
2. Install Django using pip.
3. Verify installation.

Task:

- Write the commands you used for installation and verification.

📝 Part 2: Create Your First Django Project

1. Create a new Django project.
2. Run the development server.
3. Confirm Django startup page appears.

Task:

- Screenshot the terminal running the server.
- Screenshot the browser showing Django's default welcome page.

Part 3: Create a New Django App

1. Create an app called **core**.
2. Add it to `INSTALLED_APPS` in `settings.py`.

Task:

- Write the command used to create the app.
- Paste the updated `INSTALLED_APPS` section.

📝 Part 4: Display "Hello World" in Browser

1. Create a view inside `core/views.py`.
2. Add a URL mapping inside `core/urls.py`.
3. Connect app URLs to the project's main `urls.py`.

Required Output:
Visiting:

- `http://127.0.0.1:8000/`

should show:

- `Hello World`

Task:

- Paste your `views.py`
- Paste your `urls.py`
- Screenshot the browser showing “Hello World”.