Django Syllabus From **18th February**

- **By Digital Pathshala**

Contents

Week 1: Django Basics (Days 1–6)

1. Day 1: Introduction to Django

1.1. What is Django? Why use Django?

1.2. Setting up a Django project

1.2.1. Check if Python is Installed

Before installing Django, make sure Python is installed on your system.

For Windows, open Command Prompt (cmd) and run **python --version** or **python3 --version.**

If Python is not installed, download it from the official Python website and install it.

1.2.2. Install pip (Python Package Manager)

For Windows:

Download the pip installer script from: **https://bootstrap.pypa.io/get-pip.py**

Go to the download directory and run the command:

python get-pip.py

or python3 get-pip.py

Find the Scripts directory inside the Python installation folder and add it to Environment Variables.

For Mac:

Run the following command to download the pip installer script:

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

Execute the script using:

python3 get-pip.py

For Linux:

Install pip using:

sudo apt-get install python3-pip python-dev

Verify pip Installation:

Run the following command to check if pip is installed correctly:

pip --version

or pip3 –version

1.3. Installing and using a virtual environment

After successfully installing pip, the next step is to set up a **virtual environment** using **venv**. A virtual environment allows you to manage dependencies separately for each project, making it useful for deployment or transferring projects to another system.

Installing Virtual Environment

Run the following command to install **venv**:

- **pip install virtualenv**

Each project should have its own **venv** to maintain package isolation.

To create virtual environment file:

python -m venv env

Activating the Virtual Environment

- On Windows (Command Prompt or VS Code Terminal)
    - \venv_folder_name\Scripts\activate.bat
    - \venv_folder_name\Scripts\activate
- On Mac/Linux (Terminal)
    - source venv_folder_name/bin/activate

Note: If the activation does not work in **PowerShell**, open **Command Prompt** in VS Code and try again.

Generating packages on requirement.txt file:

pip freeze > requirements.txt

2. Day 2: Django Project Structure

2.1. Installation of Django on virtual environment

Once the virtual environment is activated, install Django using:

- **pip install django**

2.2. Creation of Django Project

To create a new Django project, run:

- **django-admin startproject** project_name : blogapp

Note: Replace project_name with your preferred project name.

2.3. Running the development server

Navigate to the project directory and start the Django server:

- cd project_name
- **python manage.py runserver**

2.4. Creating and exploring a Django app

Inside your project folder, create a Django app using:

- python manage.py startapp app_name

For example, to create an app for a **blog website**:

- **python manage.py startapp** blog
- **python manage.py startapp** blog # Handles blog posts
- **python manage.py startapp** users # Manages user authentication
- **python manage.py startapp** comments # Manages comments on blog posts
- **python manage.py startapp** categories # Manages categories for blog posts
- **python manage.py startapp** likes # Manages likes on blog posts and comments

2.5. Understanding Django's Folder Structure

After creating a Django project (**django-admin startproject** project_name), the structure will look like this:

Each file serves a specific role:

- **manage.py**: A command-line utility for managing the Django project.
- **settings.py**: Configures settings like databases, middleware, installed apps, etc.
- **urls.py**: Defines URL patterns and routes to views.
- **wsgi.py / asgi.py**: Entry points for web servers (**Web Server Gateway Interface** (WSGI) for synchronous apps**, Asynchronous Server Gateway Interface**(ASGI) for async apps).
- **models.py**: Defines database models (tables).
- **views.py**: Contains functions or classes that handle requests and return responses.
- **admin.py**: Registers models to the Django admin panel.
- **apps.py**: Configures app settings.
- **migrations/**: Stores database migration files.

2.6. Role of Key Files

- **settings.py** (Configures Django Settings)

This file manages configurations like installed apps, middleware, databases, static files, etc.

settings.txt

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
    'users',
    'comments',
    'categories',
    'likes',
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / "db.sqlite3",
    }
}

STATIC_URL = "/static/"

```

- **urls.py** (Defines URL Routing)

Maps URLs to views.

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

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

- **models.py** (Defines Database Tables)

Used to create and manage database tables using Django ORM.

```python
from django.db import models

# Create your models here.
class User(models.Model):
    name = models.CharField(max_length=200)
    email = models.EmailField()
    created_at = models.DateTimeField(auto_now_add=True)
```

- **views.py** (Handles Business Logic)

Defines what happens when a user visits a particular URL.

```python
from django.shortcuts import render
from django.http import HttpResponse

def home(request):
    return HttpResponse("Welcome to my blog!")

```

2.7. Configuring Django Settings

Django settings can be modified inside settings.py. Common configurations include:

- Database Configuration

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

2.8. Creating and Managing Django Apps

- Creating an App

python manage.py startapp app_name

- Registering an App

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

- Creating Views

```python
from django.shortcuts import render
from django.http import HttpResponse

def home(request):
    return HttpResponse("Welcome to my blog!")
```

- Creating URLs

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

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