Django Roadmap (5 Weeks)

A Comprehensive Guide to Learning Django with Code Examples

Created on September 2, 2025

1 Week 1: Python & Setup

This week focuses on setting up the development environment and learning Python basics necessary for Django.

- Install Python & pip
- Learn Python basics: Variables, Data types, Operators, Functions, Loops, Conditions, Lists, Tuples, Sets, Dictionaries, File Handling, OOP (Class, Object, Inheritance)
- Set up virtual environment and install Django
- Create and run your first Django project

1.1 Example: Python Basics

```python
# Variables and Data Types
name = "John"   # String
age = 25        # Integer
height = 5.9    # Float
is_student = True  # Boolean

# Function and Loop
def greet(name):
    for i in range(3):
        print(f"Hello, {name}!")

# Class and Inheritance
class Person:
    def __init__(self, name):
        self.name = name

class Student(Person):
    def study(self):
        print(f"{self.name} is studying")

# File Handling
with open("example.txt", "w") as file:
    file.write("Hello, Django!")
```

1.2 Setting Up Django

```bash
# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate     # Windows

# Install Django
pip install django
```

```

# Start Django project
django-admin startproject mysite
cd mysite
python manage.py runserver
```

Goal: Create a Django project and run the server at `http://127.0.0.1:8000`.

2 Week 2: Django Basics

Understand Django's structure and create a basic blog application.

- Learn project structure: manage.py, settings.py, urls.py
- Create an app: `python manage.py startapp blog`
- Work with URLs, views, and templates
- Set up static files and Django admin
- Define models and run migrations

2.1 Example: Blog App Setup

```python
# blog/models.py
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

# blog/views.py
from django.shortcuts import render
from .models import Post

def post_list(request):
    posts = Post.objects.all()
    return render(request, 'blog/post_list.html', {'posts': posts
       })

# mysite/urls.py
from django.contrib import admin
from django.urls import path, include

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

# blog/urls.py
from django.urls import path
from . import views

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

2.2 Template Example

```
<!-- blog/templates/blog/post_list.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Blog</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <h1>My Blog</h1>
    {% for post in posts %}
        <h2>{{ post.title }}</h2>
        <p>{{ post.content }}</p>
    {% endfor %}
</body>
</html>
```

Goal: Build a blog app with a Post model, display it in admin, and show posts on a webpage.

3 Week 3: Intermediate Django

Deepen your understanding with ORM, forms, authentication, and media handling.

- Perform CRUD with Django ORM
- Use relationships: OneToMany, ManyToMany
- Implement forms and ModelForms
- Add user authentication
- Use template inheritance
- Handle media uploads

3.1 Example: User Authentication and Forms

```python
# blog/forms.py
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'content']

# blog/views.py
from django.contrib.auth.decorators import login_required
from .forms import PostForm

@login_required
def post_create(request):
    if request.method == 'POST':
        form = PostForm(request.POST)
        if form.is_valid():
            form.save()
    else:
        form = PostForm()
    return render(request, 'blog/post_form.html', {'form': form})

# mysite/settings.py (add media settings)
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
```

Goal: A blog where users can register, log in, and create/edit posts with images.

4 Week 4: Advanced Django

Explore advanced features like APIs, permissions, and testing.

- Use Class-Based Views
- Build APIs with Django Rest Framework
- Implement user roles and permissions
- Create custom template tags
- Use signals
- Add pagination and basic testing

4.1 Example: Django Rest Framework API

```
# Install DRF
pip install djangorestframework

# blog/serializers.py
from rest_framework import serializers
from .models import Post

class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = ['id', 'title', 'content', 'created_at']

# blog/views.py
from rest_framework import generics
from .serializers import PostSerializer

class PostListAPI(generics.ListCreateAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

# blog/urls.py
from django.urls import path
from .views import PostListAPI

urlpatterns = [
    path('api/posts/', PostListAPI.as_view(), name='post_list_api
       '),
]
```

Goal: Blog with API endpoints and user role permissions.

5 Week 5: Deployment & Scaling

Deploy your Django project and optimize for production.

- Configure production settings
- Use .env for secrets
- Deploy with Gunicorn + Nginx or Docker
- Switch to PostgreSQL
- Set up static and media files
- Add caching
- Secure the application

5.1 Example: Production Settings

```python
# mysite/settings.py
import os
from pathlib import Path
from dotenv import load_dotenv

load_dotenv()
SECRET_KEY = os.getenv('SECRET_KEY')
DEBUG = False
ALLOWED_HOSTS = ['yourdomain.com', '127.0.0.1']
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME'),
        'USER': os.getenv('DB_USER'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Goal: Deployed Django project online, ready for real users.

6 Final Outcome

After 5 weeks, you will have:

- Strong Django fundamentals
- A working blog app with login, CRUD, and API
- Deployment experience