Step-by-step guide to practice Django effectively, especially if you're starting from scratch and want real hands-on skills.

---

1. Learn the prerequisites first (don't skip this)

Before Django, make sure you're comfortable with:

- Basic Python (functions, classes, lists, dictionaries)
- HTML & CSS (for templates)
- Basic command line usage (cd, mkdir, etc.)

If Python is weak, spend a few days fixing that first—it will save you a lot of confusion later.

2. Set up your environment

Install tools:

- Install Python
- Install pip (Python package manager)
- Create a virtual environment:

```
python -m venv env
```

Activate it:

- Windows:

```
env\Scripts\activate
```

- Mac/Linux:

```
source env/bin/activate
```

Install Django:

```
pip install django
```

Check installation:

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

---

3. Create your first project

```
django-admin startproject myproject
cd myproject
python manage.py runserver
```

Open browser:

```
http://127.0.0.1:8000/
```

If you see the Django welcome page → success.

---

4. Create your first app

Go Inside the project(`cd myproject`):

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

Then register it in `settings.py`:

```
INSTALLED_APPS = [
    ...
    'myapp',
]
```

---

5. Learn the Django structure (very important)

Focus on understanding:

- `models.py` → database structure
- `views.py` → logic
- `urls.py` → routing
- `templates/` → HTML files
- `admin.py` → admin panel

6. Build your first mini project (VERY important step)

Start simple:

Project idea: "To-Do List App"

Learn step-by-step:

Step A: Create Model (models.py)

```
from django.db import models

class Task(models.Model):
    title = models.CharField(max_length=200)
    completed = models.BooleanField(default=False)
```

Step B: Migrate database

```
python manage.py makemigrations
python manage.py migrate
```

Step C: Register in admin (admin.py)

```
from .models import Task
admin.site.register(Task)
```

Create admin user:

```
python manage.py createsuperuser
```

7. Learn Views + Templates

Create a simple view:

```
from django.shortcuts import render
from .models import Task

def home(request):
    tasks = Task.objects.all()
    return render(request, 'home.html', {'tasks': tasks})
```

Create template:

myapp/templates/home.html

```
<ul>
{% for task in tasks %}
<li>
<span class="{% if task.completed %}completed{% endif %}">
{{ task.title }}
</span>

{% if task.completed %}
✔
{% else %}
□
{% endif %}
</li>
{% empty %}
<li>No tasks available</li>
{% endfor %}
</ul>
```

Here's a **complete start-to-end step-by-step CRUD guide using ModelForm in Django**. This is exactly how real Django apps are built.

We'll build a **To-Do app with full CRUD: Create, Read, Update, Delete**.

STEP 1: Create Project

```
django-admin startproject todoproject
cd todoproject
python manage.py startapp todoapp
```

□ STEP 2: Register App

In `settings.py`:

```
INSTALLED_APPS = [
    'todoapp',
]
```

□ STEP 3: Create Model

`todoapp/models.py`

```
from django.db import models

class Task(models.Model):
    title = models.CharField(max_length=200)
    completed = models.BooleanField(default=False)

    def __str__(self):
        return self.title
```

STEP 4: Migrate Database

```
python manage.py makemigrations
python manage.py migrate
```

STEP 5: Create ModelForm

Create `todoapp/forms.py`

```
from django import forms
from .models import Task

class TaskForm(forms.ModelForm):
    class Meta:
        model = Task
        fields = ['title', 'completed']
```

STEP 6: Create Views (CRUD Logic)

`todoapp/views.py`

---

1. READ (Show all tasks)

```
from django.shortcuts import render, redirect, get_object_or_404
from .models import Task
from .forms import TaskForm

def home(request):
    tasks = Task.objects.all()
    form = TaskForm()
    return render(request, 'home.html', {'tasks': tasks, 'form': form})
```

---

2. CREATE (Add task)

```
def add_task(request):
    if request.method == "POST":
        form = TaskForm(request.POST)
        if form.is_valid():
            form.save()
    return redirect('home')
```

---

3. UPDATE (Edit task)

```
def edit_task(request, task_id):
    task = get_object_or_404(Task, id=task_id)

    form = TaskForm(request.POST or None, instance=task)

    if request.method == "POST":
        if form.is_valid():
            form.save()
            return redirect('home')

    return render(request, 'edit.html', {'form': form})
```

---

4. DELETE (Remove task)

```
def delete_task(request, task_id):
    task = get_object_or_404(Task, id=task_id)
    task.delete()
    return redirect('home')
```

STEP 7: URLs

todo/urls.py

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

urlpatterns = [
    path('', views.home, name='home'),
    path('add/', views.add_task, name='add'),
    path('edit/<int:task_id>/', views.edit_task, name='edit'),
    path('delete/<int:task_id>/', views.delete_task, name='delete'),
]
```

In main `urls.py`:

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

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

STEP 8: HOME TEMPLATE

templates/home.html

```
<h1>To-Do List</h1>

<form method="POST" action="add/">
    {% csrf_token %}
    {{ form.as_p }}
    <button>Add</button>
</form>

<hr>

<ul>
    {% for task in tasks %}
        <li>
            {{ task.title }}

            {% if task.completed %}
                ✔
            {% else %}
                □
            {% endif %}

            <a href="edit/{{ task.id }}/">Edit</a>
            <a href="delete/{{ task.id }}/">Delete</a>
        </li>
    {% endfor %}
</ul>
```

✏□ STEP 9: EDIT TEMPLATE

templates/edit.html

```
<h1>Edit Task</h1>

<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button>Update</button>
</form>
```

STEP 10: RUN SERVER

```
python manage.py runserver
```

Open:

```
http://127.0.0.1:8000/
```

FULL CRUD FLOW (VERY IMPORTANT)

□ CREATE

User submits form → `form.save()` → new task added

□ READ

`Task.objects.all()` → shows all tasks

□ UPDATE

1. Open edit page
2. `get_object_or_404(Task, id)`
3. Fill form with existing data
4. Save changes

DELETE

1. Find task using `get_object_or_404`
2. `task.delete()`