use this **15-minute daily Django warm-up checklist**.

perform these steps from scratch at the start of every class or study session until they can do it without looking.

Phase 1: Environment Setup (4 Minutes)

The goal is to automate the "getting started" commands.

- **Create a fresh directory:** `mkdir recap && cd recap`
- **Initialize a Virtual Environment:** `python -m venv env`
- **Activate the Environment:**
  - *Windows:* `env\Scripts\activate`
  - *Mac/Linux:* `source env/bin/activate`
- **Install Django:** `pip install django`
- **Create the Project:** `django-admin startproject myproject .` (Note the `.` at the end to keep the folder structure flat)

Phase 2: App & Database Initialisation (4 Minutes)

These commands connect the logic to the framework.

- **Create an App:** `python manage.py startapp myapp`
- **Register the App:** Add `'myapp',` to the `INSTALLED_APPS` list in `settings.py`.
- **Initial Migrations:** `python manage.py migrate` (to set up default admin/auth tables).
- **Create a Superuser:** `python manage.py createsuperuser` (use 'admin' for both for speed).

Phase 3: The "Hello World" Logic (7 Minutes)

This reinforces the **MVT (Model-View-Template)** flow.

- **Define a View:** In `myapp/views.py`, write a simple function that returns an `HttpResponse("Hello")`.

1. Define the View

In `myapp/views.py`:

```
from django.http import HttpResponse

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

- **Map the URL (App level):** Create `myapp/urls.py` and link the view to a path.

2. Map the URL (App Level)

Create a new file `myapp/urls.py`:

```

from django.urls import path
from . import views

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

- **Map the URL (Project level):** Include the app's URLs in `myproject/urls.py`.

3. Map the URL (Project Level)

In your main project folder `myproject/urls.py`:

```

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

urlpatterns = [
    path('admin/', admin.admin.site.urls),
    path('', include('myapp.urls')), # This connects your app to the
project
]
```

- **Run the Server:** `python manage.py runserver` and verify the "Hello" message in the browser

To master Django, students must shift from "reading" to "typing." This daily practice plan focuses on the **Model-Migration-Admin** flow, which is the backbone of most Django applications.

The 15-Minute "Database & Admin" Daily Drill

Have your students perform these steps from scratch every day. They should delete their progress and restart if they make a mistake.

1. **Define a Model (3 mins):** Create a simple entity (e.g., a "Student") in `myapp/models.py`.
2. **Run Migrations (2 mins):** Create and apply the database schema changes.
3. **Register with Admin (3 mins):** Make the model visible in the dashboard.
4. **Create a Superuser (2 mins):** Set up access to view the data.
5. **Verify (5 mins):** Log in to the admin panel and manually add one record.

The Practice Code

1. Define the Model

In `myapp/models.py`, use basic fields to learn types and the `__str__` method.

python

```python
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    email = models.EmailField()

    def __str__(self):
        return self.name  # Shows the actual name in the Admin panel
```

2. Register in Admin

In `myapp/admin.py`, connect the model to the interface.

```
from django.contrib import admin
from .models import Student  # Import the model you just made

admin.site.register(Student)
```

3. Terminal Commands

Students must type these manually every single time to memorize them:

- **Step A: Generate instructions:** `python manage.py makemigrations`
- **Step B: Apply to database:** `python manage.py migrate`
- **Step C: Create admin access:** `python manage.py createsuperuser`
- **Step D: Start server:** `python manage.py runserver`

Why This Works

- **Muscle Memory:** Typing `admin.site.register()` and `makemigrations` daily makes them second nature.
- **Visual Reward:** Seeing the "Student" table appear in the browser at `127.0.0.1:8000/admin` gives students an immediate sense of accomplishment.
- **The "Why" Behind the "How":** They learn that `models.py` defines the **data**, while `admin.py` defines the **view** of that data