Django Python web framework

START HERE

`django-admin startproject <name>`
New project

`python manage.py startapp <app>`
New app

`python manage.py runserver`
Dev server

`python manage.py makemigrations`
Create migrations

`python manage.py migrate`
Apply migrations

`python manage.py createsuperuser`
Admin account

`python manage.py shell`
Shell with models

`python manage.py test`
Run tests

models.py edit models → `makemigrations` → **Migration file** in migrations/ → `migrate` → **Database** schema updated → `runserver` → **Dev server** localhost:8000

CONCEPTS

manage.py is django-admin pinned to your project's settings; inside a project use manage.py.

A project holds settings and URLs; it is made of reusable apps, each with its own models, views and migrations.

runserver is for development only; deploy with a WSGI/ASGI server and run check --deploy first.

PROJECT

`django-admin startproject <name>`
Create a project
Add a trailing . to use the current directory instead of a new folder.

`python manage.py startapp <app>`
Create an app in the project
Then add it to INSTALLED_APPS in settings.

`python manage.py check`
Run system checks

`python manage.py check --deploy`
Check production settings

`django-admin startproject <name> --template <url>`
Start from a project template
Templates can run code; inspect before use.

`python manage.py diffsettings`
Settings that differ from defaults

`django-admin --version`
Print the Django version

Generated layout

Path	Purpose
`manage.py`	project command runner
`<name>/settings.py`	configuration
`<name>/urls.py`	root URL routes
`<app>/models.py`	database models
`<app>/views.py`	request handlers
`<app>/migrations/`	schema history

RUN

`python manage.py runserver`
Start the dev server
Auto-reloads on code changes.

RUN (cont.)

`python manage.py runserver <port>`
Use another port

`python manage.py runserver 0.0.0.0:8000`
Listen on all interfaces
Needs the host in ALLOWED_HOSTS.

`python manage.py runserver --noreload`
Disable auto-reload

`python manage.py shell`
Python shell with Django set up

`python manage.py shell -c "<code>"`
Run one Python statement

`python manage.py dbshell`
Open the database client

`python manage.py createsuperuser`
Create an admin user

`python manage.py changepassword <user>`
Change a user's password

MIGRATIONS

`python manage.py makemigrations`
Create migrations from model changes

`python manage.py migrate`
Apply pending migrations

`python manage.py makemigrations <app>`
Limit to one app

`python manage.py showmigrations`
List migrations and applied state

`python manage.py makemigrations --dry-run`
Preview without writing files

`python manage.py makemigrations --check`
Fail if migrations are missing
Useful in CI.

`python manage.py migrate <app> <migration>`
Migrate to a given point
Use zero to unapply all of an app.

`python manage.py sqlmigrate <app> <migration>`
Print a migration's SQL

`python manage.py makemigrations --empty <app>`
Empty migration for custom work

MIGRATIONS (cont.)

`python manage.py migrate --plan`
Show what migrate would do

`python manage.py migrate --fake <app> <migration>`
Mark applied without running
Only when the schema already matches.

`python manage.py squashmigrations <app> <migration>`
Squash into fewer migrations

TESTS

`python manage.py test`
Run all tests

`python manage.py test <app>.tests.<Class>.<method>`
Run one test

`python manage.py test --keepdb`
Reuse the test database

`python manage.py test --parallel`
Run tests in parallel

`python manage.py test --failfast`
Stop at first failure

`python manage.py test -k <pattern>`
Run tests matching a name

`python manage.py test --pdb`
Debugger on error or failure

`python manage.py test --debug-sql`
Show SQL for failing tests

DATA

`python manage.py dumpdata <app>.<Model> --indent 2`
Export data as JSON

`python manage.py loaddata <fixture>`
Import a fixture

`python manage.py flush`
Empty all tables, keep migrations
Asks for confirmation.

`python manage.py inspectdb`
Models from an existing database

`python manage.py dumpdata --natural-foreign --exclude contenttypes`
Portable export

STATIC & I18N

`python manage.py collectstatic`
Gather static files for deployment
Needs STATIC_ROOT set.

`python manage.py collectstatic --noinput`
Run without prompts

`python manage.py findstatic <path>`
Where a static file resolves

`python manage.py makemessages -l <lang>`
Extract translatable strings

`python manage.py compilemessages`
Compile .po to .mo

SHELL QUERIES

`Model.objects.all()`
All rows

`Model.objects.filter(<field>=<value>)`
Rows matching a condition

`Model.objects.get(pk=<id>)`
One row; raises if missing or many

`Model.objects.create(<field>=<value>)`
Insert a row

`print(qs.query)`
Show the SQL of a queryset

RECIPES

Change a model

```
# edit <app>/models.py
python manage.py makemigrations
python manage.py sqlmigrate
<app> 0002
python manage.py migrate
```

Fresh project

```
django-admin startproject
mysite .
python manage.py startapp polls
python manage.py migrate
python manage.py createsuperuser
python manage.py runserver
```

Move data between databases

```
python manage.py dumpdata --
natural-foreign --exclude
contenttypes > data.json
# point settings at the new DB
python manage.py migrate
python manage.py loaddata
data.json
```

CI gate

```
python manage.py check --deploy
python manage.py makemigrations
--check --dry-run
python manage.py test --parallel
```