Django

Models, queries, views and templates in Python.

01 Install & start a project

```
python -m venv venv
source venv/bin/activate
pip install django
django-admin startproject myproject .
python manage.py startapp myapp
```

Django 6.0 needs Python 3.12+. Add the app to INSTALLED_APPS in settings.py.

02 manage.py commands

runserver	Dev server on port 8000
makemigrations	Write migrations from models
migrate	Apply migrations to the DB
createsuperuser	Create an admin login
shell	Python shell, models imported
collectstatic	Copy static files for deploy

Prefix each with python manage.py.

03 Define a model

```
from django.db import models
class Product(models.Model):
    name = models.CharField(max_length=200)
    price = models.DecimalField(
        max_digits=10, decimal_places=2)
    created = models.DateTimeField(
        auto_now_add=True)
    def __str__(self):
        return self.name
```

auto_now_add stamps the creation time; auto_now updates on every save. __str__ labels rows in the admin.

04 Relationships & Meta

```
class Product(models.Model):
    category = models.ForeignKey(
        Category, on_delete=models.CASCADE,
        related_name='products')
    created_by = models.ForeignKey(
        User, on_delete=models.SET_NULL,
        null=True)
    class Meta:
        ordering = ['-created']
```

related_name names the reverse side: category.products.all()

05 Query basics

```
Product.objects.all()
Product.objects.filter(price__lt=100)
Product.objects.exclude(stock=0)
Product.objects.get(id=1)
Product.objects.order_by('-price').first()
Product.objects.create(name='Pen', price=2)
product.save()    # insert or update
product.delete()
```

get() raises DoesNotExist when no row matches.

06 Field lookups

price__lt=100	Less than (lte, gt, gte)
name__icontains=	Contains, any case
id__in=[1, 2, 3]	Value in a list
created__year=	Year of a date
category__name=	Field on a related model

A plain name=value is an exact match.

07 Function views

```
from django.shortcuts import (
    get_object_or_404, render)
def product_detail(request, slug):
    product = get_object_or_404(
        Product, slug=slug)
    return render(request, 'detail.html',
                  {'product': product})
```

get_object_or_404 turns a missing row into a 404 page.

08 URL routing

```
app_name = 'myapp'  # myapp/urls.py
urlpatterns = [
    path('products/<slug:slug>/',
         views.product_detail,
         name='product_detail'),
]
# myproject/urls.py
path('', include('myapp.urls')),
```

Converters: str, int, slug, uuid. Link by name: reverse() in Python, {% url %} in templates.

09 Template syntax

```
{{ product.price|floatformat:2 }}
{% if product.stock %}In stock{% endif %}
{% for product in products %}
  <h2>{{ product.name }}</h2>
{% empty %}
  <p>No products found.</p>
{% endfor %}
```

{{ }} outputs a value; | applies a filter. {% empty %} renders when the loop has no items.

Django

Class views, forms, admin, auth and deployment.

10 Template inheritance

```
<!-- base.html -->
{% load static %}
<img src="{% static 'img/logo.png' %}">
{% block content %}{% endblock %}
<!-- products/list.html -->
{% extends 'base.html' %}
{% block content %}<p>Hi</p>{% endblock %}
```

Child templates fill the parent's named blocks.

11 Class-based views

```
class ProductListView(ListView):
    model = Product
    paginate_by = 12
    def get_queryset(self):
        q = self.request.GET.get('q', '')
        return Product.objects.filter(
            name__icontains=q)
```

paginate_by adds page_obj. Route it with .as_view().

12 ModelForm & validation

```
class ProductForm(forms.ModelForm):
    class Meta:
        model = Product
        fields = ['name', 'price', 'stock']
    def clean_price(self):
        price = self.cleaned_data['price']
        if price <= 0:
            raise forms.ValidationError(
                'Price must be above zero')
        return price
```

clean() checks fields against each other.

13 Handle a form

```
def create_product(request):
    if request.method == 'POST':
        form = ProductForm(request.POST)
        if form.is_valid():
            item = form.save(commit=False)
            item.created_by = request.user
            item.save()
            return redirect('myapp:index')
    else:
        form = ProductForm()
    return render(request, 'form.html',
                  {'form': form})
```

Put {% csrf_token %} inside every POST form.

14 Admin

```
@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    list_display = ['name', 'price']
    list_filter = ['category', 'created']
    search_fields = ['name', 'description']
```

Open /admin/ after running createsuperuser.

15 Testing

```
class ProductTests(TestCase):
    def setUp(self):
        Product.objects.create(
            name='Test', price=10)
    def test_list(self):
        r = self.client.get('/products/')
        self.assertEqual(r.status_code, 200)
```

self.client simulates requests. Run python manage.py test.

16 Faster queries

```
Product.objects.filter(
    Q(name__icontains='phone') |
    Q(description__icontains='phone'))
Product.objects.filter(id=1).update(
    stock=F('stock') - 1)
Product.objects.aggregate(Avg('price'))
Product.objects.select_related('category')
Product.objects.prefetch_related('tags')
```

F() avoids race conditions. select_related joins ForeignKeys; prefetch_related batches many-to-many.

17 Authentication

```
@login_required
def profile(request):
    return render(request, 'profile.html')
user = authenticate(
    request, username=name, password=pw)
if user is not None:
    login(request, user)
```

LoginRequiredMixin guards class views.
include('django.contrib.auth.urls') adds login views.

18 Production settings

```
DEBUG = False
ALLOWED_HOSTS = ['yourdomain.com']
SECRET_KEY = os.environ['SECRET_KEY']
STATIC_ROOT = BASE_DIR / 'staticfiles'
SECURE_SSL_REDIRECT = True
```

```
gunicorn myproject.wsgi --bind 0.0.0.0:8000
```

Run collectstatic and check --deploy before going live.