Django Complete Cheat Sheet

1. Project & App Setup

- Create project: `django-admin startproject projectname`
- Create app: `python manage.py startapp appname`
- Run server: `python manage.py runserver`
- Apply migrations: `python manage.py migrate`
- Create migration: `python manage.py makemigrations`
- Create superuser: `python manage.py createsuperuser`

2. Settings Overview

- `INSTALLED_APPS`: register your apps
- `DATABASES`: configure database
- `TEMPLATES`: template engine settings
- `STATIC_URL`, `STATICFILES_DIRS`: static files
- `MEDIA_URL`, `MEDIA_ROOT`: file uploads

3. URL Routing

```
# project/urls.py
from django.urls import path, include
urlpatterns = [ path('', include('app.urls')) ]

# app/urls.py
from django.urls import path
from . import views
urlpatterns = [ path('', views.home, name='home') ]
```

4. Views

Function-based:

```
def home(request):
    return render(request, 'home.html')
```

Class-based:

```
from django.views import View
class HomeView(View):
    def get(self, request):
        return render(request, 'home.html')
```

5. Models

```
from django.db import models
class Item(models.Model):
    name = models.CharField(max_length=100)
    price = models.IntegerField()
    created = models.DateTimeField(auto_now_add=True)
```

QuerySet Basics

- `Item.objects.all()`
- `Item.objects.get(id=1)`
- `Item.objects.filter(price__gte=100)`
- `Item.objects.create(name="A")`
- `item.save()`
- `item.delete()`

6. Templates

- Use `{% %}` for logic, `{{ }}` for data.
- Template example:

```
<h1>{{ item.name }}</h1>
{% for obj in items %}
  <p>{{ obj.name }}</p>
{% endfor %}
```

7. Static & Media

- Static: JS/CSS/images

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

- Media uploads with `ImageField` / `FileField`

8. Forms

Django Forms

```
from django import forms
class ItemForm(forms.Form):
    name = forms.CharField(max_length=100)
```

Model Forms

```
class ItemForm(forms.ModelForm):
    class Meta:
        model = Item
        fields = '__all__'
```

9. Authentication

- Login: `from django.contrib.auth import login, authenticate`
- Decorator: `@login_required`
- Default User model: `from django.contrib.auth.models import User`
- Custom user via `AbstractUser`

10. Django Admin

```
from django.contrib import admin
from .models import Item
admin.site.register(Item)
```

11. Middlewares

- All middlewares located in `MIDDLEWARE` list
- Create custom middleware:

```
class MyMiddleware:
    def __init__(self, get_response): self.get_response = get_response
    def __call__(self, request): return self.get_response(request)
```

12. Django REST Framework (DRF)

Serializers

```
from rest_framework import serializers
class ItemSerializer(serializers.ModelSerializer):
    class Meta: model = Item; fields = '__all__'
```

ViewSets

```
from rest_framework.viewsets import ModelViewSet
class ItemViewSet(ModelViewSet): queryset = Item.objects.all();
serializer_class = ItemSerializer
```

Router

```
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register('items', ItemViewSet)
urlpatterns = router.urls
```

13. ORM Advanced

- `select_related()` for FK optimization
- `prefetch_related()` for M2M optimization

14. Signals

```
from django.db.models.signals import post_save
from django.dispatch import receiver
@receiver(post_save, sender=Item)
def after_save(sender, instance, created, **kwargs): pass
```

15. Caching

- `from django.core.cache import cache`
- `cache.set('key', value)`
- `cache.get('key')`

16. Deployment Notes

- Use gunicorn/uwsgi
- Use nginx reverse proxy
- Set `DEBUG = False`
- Serve static with `collectstatic`

Let me know if you want this expanded into a **full Django handbook**, or broken into multiple focused cheat sheets!