

Django Beginner Question Set

Part 1: Multiple Choice Questions

Q1. Which command starts a new Django project?

- a) django-admin startproject projectname**
- b) python manage.py startproject
- c) django-admin createproject
- d) django-admin startapp

Q2. Which file inside the project contains database settings?

- a) views.py
- b) models.py
- c) settings.py**
- d) urls.py

Q3. The default database for Django projects is:

- a) PostgreSQL
- b) MySQL
- c) SQLite**
- d) Oracle

Q4. Which command runs the development server?

- a) django-admin server
- b) python manage.py runserver**
- c) django-admin runserver
- d) python runserver.py

Q5. What is the default port for Django runserver?

- a) 5000
- b) 3000
- c) 8000**
- d) 8080

Q6. To activate a virtual environment in Windows:

- a) env/bin/activate
- b) env\Scripts\activate**
- c) source env/activate
- d) activate env

Q7. manage.py is used for:

- a) Running server only
- b) Admin interface
- c) Command-line utility for project management**
- d) Creating migrations only

Q8. Which command is invalid in Django?

- a) python manage.py makemigrations
- b) python manage.py migrate
- c) python manage.py startproject
- d) python manage.py startenv**

Q9. To check Django version:

- a) django --v
- b) python -m django --version
- c) django-admin version
- d) Both b and c**

Q10. ASGI in Django is for:

- a) Database connection
- b) Template rendering
- c) Asynchronous server interface**
- d) Authentication

Q11. Which command creates an app?

- a) django-admin createapp
- b) python manage.py startapp appname
- c) django-admin startapp appname
- d) Both b and c**

Q12. Apps must be registered inside:

- a) urls.py
- b) settings.py → INSTALLED_APPS**
- c) admin.py
- d) views.py

Q13. Which file defines app configuration?

- a) apps.py**
- b) views.py
- c) models.py
- d) migrations

Q14. What does __init__.py do in an app?

- a) Marks folder as a Python package**
- b) Creates database
- c) Starts server
- d) Loads templates

Q15. Which of these is NOT generated by startapp?

- a) models.py
- b) templates/**
- c) views.py
- d) tests.py

Q16. Which field creates auto-increment primary key by default?

- a) models.IntegerField
- b) models.PrimaryKeyField
- c) models.AutoField
- d) None (Django adds automatically)**

Q17. A CharField must define:

- a) unique=True
- b) primary_key=True
- c) max_length**
- d) null=True

Q18. Which command generates migration files?

- a) python manage.py migrate
- b) python manage.py makemigrations**
- c) python manage.py runserver
- d) python manage.py build

Q19. Which command applies migrations to database?

- a) makemigrations
- b) migrate**
- c) syncdb
- d) update

Q20. In models, verbose_name is used for:

- a) Field default value
- b) Human-readable field name**
- c) Auto-increment
- d) Table name

Q21. To define relationship between two models, we use:

- a) models.ManyToOne
- b) models.ForeignKey**
- c) models.JoinField
- d) models.Relationship

Q22. Default primary key field type in Django :

- a) IntegerField
- b) BigAutoField**
- c) UUIDField
- d) CharField

Q23. To delete all migrations and reset database:

- a) flush**
- b) reset
- c) delete
- d) clear

Q24. null=True means:

- a) Field accepts NULL in database**
- b) Field accepts empty string in form
- c) Both a and b
- d) None

Q25. Which is correct model example?

- a) class Book(models.Model): title = models.Text()
- b) class Book(models.Model): title =**

models.CharField(max_length=200)

- c) class Book: title = models.CharField(200)
- d) class Book(models.Model): title = models.Char(200)

Q26. Templates folder must be added in:

- a) settings.py → TEMPLATES → DIRS**
- b) INSTALLED_APPS
- c) urls.py
- d) views.py

Q27. Template tags start with:

- a) {{ }}
- b) {% %}
- c) << >>
- d) Both a and b**

Q28. {{ variable }} in Django template means:

- a) Template tag
- b) Output variable value**
- c) Comment
- d) Loop

Q29. To extend a base template:

- a) {% extends "base.html" %}**
- b) {% include "base.html" %}
- c) {% base "base.html" %}
- d) {% load "base.html" %}

Q30. To load static files in templates:

- a) {% static 'path' %} without load
- b) {% load static %} then {% static 'path' %}**
- c) {{ static 'path' }}
- d) <static>

Q31. Which is valid loop in Django template?

- a) {% for item in list %} ... {% endfor %}**
- b) for item in list:
- c) <loop>
- d) foreach

Q32. Template comments are written as:

- a) <!-- comment -->
- b) {# comment #}**
- c) // comment
- d) /* comment */

Q33. In urls.py, URL patterns are defined as:

- a) urlpatterns = []**
- b) urls = { }
- c) routes = ()
- d) path = []

Q34. Correct syntax for mapping a view:

- a) path('home/', views.home)
- b) path('home/', home)
- c) url('home', views.home)
- d) Both a and b**

Q35. include() in urls.py is used for:

- a) Importing views
- b) Linking app urls**
- c) Rendering templates
- d) Serving static files

Q36. If no URL matches, Django raises:

- a) 404 Not Found**
- b) 500 Internal Server Error
- c) 403 Forbidden
- d) 400 Bad Request

Q37. Which file is responsible for connecting project URLs to app URLs?

- a) views.py
- b) models.py
- c) urls.py**
- d) settings.py

Q38. To redirect URLs, Django provides:

- a) redirect()
- b) return HttpResponseRedirect()
- c) Both a and b**
- d) url_redirect()

Q39. Command to create superuser:

- a) python manage.py createsuper
- b) python manage.py createsuperuser**
- c) python manage.py superuser
- d) django-admin createsuperuser

Q40. Default superuser fields required:

- a) username, password
- b) username, email, password**
- c) username, email, first name, last name
- d) only password

Q41. Django admin site URL by default:

- a) /dashboard/
- b) /admin/**
- c) /root/
- d) /superuser/

Q42. Which file is NOT created in Django project root?

- a) manage.py
- b) settings.py
- c) urls.py
- d) models.py**

Q43. Which file registers models for admin site?

- a) views.py
- b) urls.py
- c) admin.py**
- d) apps.py

Q44. Django follows which design pattern?

- a) MVC
- b) MVT**
- c) MVP
- d) MTV

Q45. settings.DEBUG = True means:

- a) Production mode
- b) Development mode**
- c) Disable database
- d) Disable static files

Part 2: Written Questions

Q1. Explain steps to create a Django project.

Answer:

- Create environment
- Install Django
- `django-admin startproject projectname`
- Run server.

Q2. What is the purpose of virtual environment in Django?

Answer: Isolates dependencies for each project.

Q3. Difference between `django-admin` and `manage.py`.

Answer:

- `django-admin` → global commands.
- `manage.py` → project-specific commands with settings.

Q4. How do you check installed apps in Django?

Answer: Check `INSTALLED_APPS` in `settings.py`.

Q5. What is `settings.py`?

Answer: Stores configurations like database, middleware, installed apps.

Q6. What is an app in Django?

Answer: A module that handles a specific functionality (blog, shop, etc.).

Q7. Steps to create and register an app.

Answer:

- `python manage.py startapp appname`
- add `appname` to `INSTALLED_APPS`.

Q8. What is the role of `apps.py`?

Answer: Stores app configuration.

Q9. What is the difference between a Django project and an app?

Answer:

- A **project** is the entire Django setup (settings, database config, global URLs).
- An **app** is a module inside a project that handles one specific feature (e.g., blog, shop, accounts).
- One project can contain multiple apps.

Q10. Why is `__init__.py` important?

Answer: Converts folder into Python package.

Q11. Explain migrations in Django.

Answer: Process of applying model changes to database. When we change our models (add fields, delete fields, rename something), we create a migration file that records the change.

Key commands:

- makemigrations → creates migration files from model changes.
- migrate → applies migration files to the database.

Q12. What is difference between null=True and blank=True?

Answer:

- null=True → database field can store NULL.
- blank=True → form field can be left empty.

Q13: What is ORM in Django?

Answer: Object-Relational Mapping that lets you query database using Python instead of SQL.

Q14: What are template tags?

Answer: Special Django syntax like {% for %}, {% if %}, {% block %} for logic in HTML.

Q15. Write a model for Student with name and age.

Answer:

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

Q16. What are Django templates?

Answer: HTML files with dynamic placeholders.

Q17. Which function loads HTML templates in a view?

Answer: return render(request, 'index.html', context)

Q18. Explain extends and include in templates.

Answer:

- extends → inherit base template.
- include → embed another template inside.

Q19. What is {% block %} in templates?

Answer: Section that child templates can override.

Q20. How to load static files?

Answer: {% load static %} then {% static 'path' %}.

Q21. How does URL mapping work in Django?

Answer:

- In `urls.py`, you define `urlpatterns = [path('home/', views.home)]`.
- When a user visits `/home/`, Django runs `home` view.

Q22. What is the difference between `models.TextField()` and `models.CharField()`?

Answer:

- `CharField` → small strings, requires `max_length`.
- `TextField` → large text, no `max_length` required.

Q23. What is the difference between `extends` and `block` in Django templates?

Answer:

- `extends` → inherit layout from base template.
- `block` → define sections child templates can replace.

Q24. What is CSRF token in Django templates?

Answer: Security token to prevent Cross-Site Request Forgery. Used in forms as `{% csrf_token %}`.

Q25. What happens if no URL matches?

Answer: Django returns 404 error page.

Q26. What is Django admin?

Answer: Built-in admin panel for managing models.

Q27. How to register model in admin?

Answer:

```
from django.contrib import admin
```

```
from .models import Student
```

```
admin.site.register(Student)
```

Q28. How to change password of superuser?

Answer: `python manage.py changepassword username`

Q29. How to customize admin list display?

Answer: Use `list_display` in `ModelAdmin`.

Q30. Difference between staff user and superuser.

Answer:

- Staff user = access admin but limited.
- Superuser = full access.

Q31. What is a view in Django?

Answer: A view is a Python function or class that receives a web request and returns a web response (HTML, JSON, redirect, etc.).

Q32. Difference between function-based view (FBV) and class-based view (CBV).

Answer:

- **FBV** → simple Python functions that handle requests.
- **CBV** → Python classes that provide built-in methods (get, post, etc.), reusable and extendable.


Q33. What is the difference between `all()`, `filter()`, and `get()` in Django ORM?

Answer:

- `all()` → returns all records (QuerySet).
- `filter()` → returns matching records (QuerySet).
- `get()` → returns a single record (Object) or raises error if not found.

Q34. Example of rendering a template in a view.

Answer:

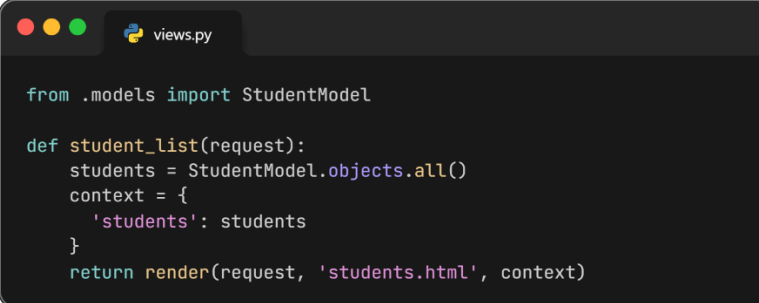


```
from django.shortcuts import render

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

Q35. How to fetch all objects from a model in a view?

Answer:



```
from .models import StudentModel

def student_list(request):
    students = StudentModel.objects.all()
    context = {
        'students': students
    }
    return render(request, 'students.html', context)
```

Q36. How to filter objects in a view?

Answer:

```
views.py

from .models import StudentModel

def student_list(request):
    students = StudentModel.objects.filter(age__gte=18)
    context = {
        'students': students
    }
    return render(request, 'students.html', context)
```

Q37. How to get a single object from the database?

Answer:

```
views.py

from django.shortcuts import get_object_or_404
from .models import StudentModel

def student_list(request):
    def student_detail(request, id):
        student = get_object_or_404(StudentModel, id=id)
        context = {
            'student': student
        }
        return render(request, 'students.html', context)
```

Q38. How to display QuerySet data in a template?

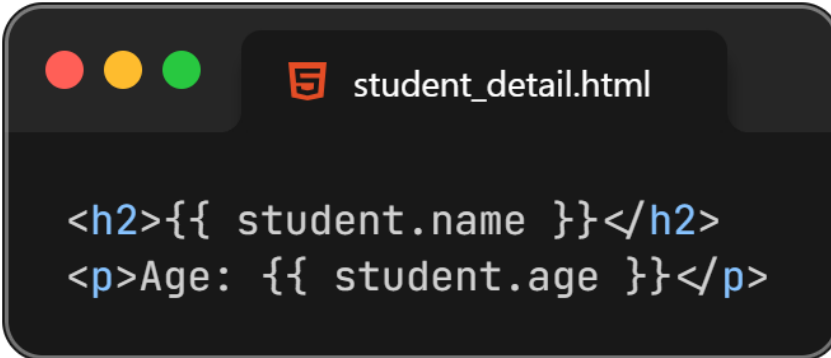
Answer (students.html):

```
students.html

<ul>
{% for student in students %}
    <li>{{ student.name }} - {{ student.age }}</li>
{% endfor %}
</ul>
```


Q39. How to display a single object in a template?

Answer (student_detail.html):



```
<h2>{{ student.name }}</h2>  
<p>Age: {{ student.age }}</p>
```