1. What is a REST API?

A REST API (Representational State Transfer Application Programming Interface) is a way for different software systems to communicate over HTTP in a standardized, resource-oriented manner. Key ideas:

- **Resource-oriented: Everything is treated as a resource (e.g. users, posts, products).**
- **Stateless: Each request from client to server must contain all the information needed; the server doesn't store client context between requests.**
- **Uniform interface: A consistent way to identify and interact with resources using URLs and standard HTTP methods.**

Common REST API request methods:

Method	Purpose
GET	Retrieve one or more resources
POST	Create a new resource
PUT	Update an existing resource (full replacement)
PATCH	Partially update an existing resource
DELETE	Remove a resource
OPTIONS	Discover available methods on a resource

Example

- **GET /api/posts/ → list all posts**
- **GET /api/posts/42/ → retrieve post with ID 42**
- **POST /api/posts/ with a JSON body → create a new post**
- **PATCH /api/posts/42/ with { "title": "New" } → update only the title of post 42**

---

2. What is Django REST Framework?

Django REST Framework (DRF) is a powerful and flexible toolkit for building Web APIs in Django. It provides:

- **Serializers to convert between Django models and JSON (or other formats).**

- **ViewSets and Routers to automate URL routing and view logic.**
- **Authentication and Permissions classes to secure your API.**
- **Browsable API: a web-based interface for testing endpoints directly in your browser.**

DRF sits on top of Django, letting you leverage all of Django's features (ORM, middleware, etc.) while focusing on API development.

---

3. What are Serializers and Deserializers in DRF?

- **A Serializer in DRF is similar to a Django Form:**
  - **Takes Django model instances or querysets**
  - **Converts them into native Python datatypes (e.g. dict, list)**
  - **Those datatypes are then rendered into JSON, XML, etc.**
- **A Deserializer does the reverse:**
  - **Takes incoming data (usually JSON)**
  - **Validates it against defined rules**
  - **Converts it into a Django model instance or Python object**

Basic Example of a DRF Serializer

```
from rest_framework import serializers
from .models import Post


class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = ['id', 'title', 'content', 'author', 'created_at']
```

- **Serialization (model → JSON):**

```
post = Post.objects.get(pk=1)
serializer = PostSerializer(post)
serializer.data
```

 # => { "id": 1, "title": "...", "content": "...", ... }

- **Deserialization (JSON → model):**

```
data = { "title": "New", "content": "Hello", "author": 3 }
serializer = PostSerializer(data=data)
if serializer.is_valid():
    post = serializer.save()  # creates a Post instance
```