1. **Explain about Django architecture?**

   Django follows a Model-View-Template (MVT) architecture. Models handle data storage and manipulation, Views handle user interaction and business logic, and Templates handle presentation logic.

2. **Explain about Django project directory structure?**

   In Django, a project typically consists of settings, URLs, and the main application. The project directory contains the main settings file (`settings.py`), URL configuration (`urls.py`), and the main application directory.

3. **Explain about models in Django?**

   Models in Django are Python classes that represent database tables. They define the structure of the database, including fields and relationships between them. Models are typically located in the `models.py` file of each application.

4. **Explain the purpose of templates in Django?**

   Templates in Django are used to generate HTML dynamically. They allow you to separate the design from the business logic, making it easier to maintain and modify your application's appearance. Templates are typically stored in the `templates` directory of each application.

5. **Explain about views in Django?**

   Views in Django are Python functions or classes that receive web requests and return web responses. They contain the logic that processes the request and decides what to do with it. Views are typically located in the `views.py` file of each application.

6. **How many types of views are there in Django?**

   In Django, views can be function-based views or class-based views. Function-based views are simple functions, while class-based views are based on Django's class-based view system.

7. **Explain about Django ORM?**

   Django's ORM (Object-Relational Mapping) is a powerful feature that allows you to interact with the database using Python objects instead of SQL queries. It abstracts away the details of the database and makes it easier to work with data.

8. **What is the purpose of static files in Django?**

   Static files in Django are files like CSS, JavaScript, and images that are served directly to the client without any processing by the server. They are used for styling and client-side behavior and are typically stored in the `static` directory of each application.

9. **How to configure static files inside template files of Django?**

   In Django templates, you can use the `{% static %}` template tag to refer to static files. For example, `{% static 'css/style.css' %}` would refer to a CSS file named `style.css` in the `css` directory of your static files directory.

10. **Explain about Django template language/jinja templating?**

    Django template language is a simple, yet powerful syntax for creating dynamic HTML templates. It allows you to include logic, loops, and conditionals directly in your templates.

11. **Explain about django-admin command?**

    `django-admin` is a command-line utility that comes with Django and is used for various administrative tasks such as creating projects, applications, running development servers, and managing database migrations.

12. **Explain about the purpose of manage.py script?**

    `manage.py` is a script that automatically gets created when you create a new Django project. It's a thin wrapper around the `django-admin` command that allows you to run management commands specific to your project.

13. **Explain about Django URLs?**

    In Django, URLs are used to map web requests to view functions or classes. They are defined in the `urls.py` file of each application and can include regular expressions to match different patterns.

14. **Explain about the difference between project and app in Django?**

    A Django project is a collection of configuration and applications for a particular website. An app is a web application that does something – e.g., a blog, a database of public records, or a simple contact form.

15. **Explain about model inheritance and specify types of model inheritance?**

    In Django, model inheritance allows you to create new models that inherit fields and methods from existing models. There are three types of model inheritance: abstract base classes, multi-table inheritance, and proxy models.

16. **Explain the purpose of signals in Django?**

    Signals in Django allow certain senders to notify a set of receivers when some action has taken place. They are used for decoupled applications where certain actions need to trigger other actions.

17. **What are the various built-in signals of Django?**

    Some built-in signals in Django include `pre_save`, `post_save`, `pre_delete`, and `post_delete`, which are triggered before and after saving or deleting a model instance, respectively.

18. **Explain the importance of caching in Django?**

    Caching in Django helps improve the performance of web applications by storing frequently accessed data in memory or on disk, reducing the need to regenerate it for each request.

19. **Explain various caching strategies of Django?**

Django supports various caching strategies such as per-view caching, template fragment caching, and low-level caching using the cache API.

20. **How authentication will be performed in Django?**

Django provides a built-in authentication system that allows users to authenticate using usernames and passwords. It also supports third-party authentication using libraries like OAuth.

21. **Explain Django request-response cycle?**

In the Django request-response cycle, a request is received by the server, which passes it to the appropriate view function. The view processes the request and returns a response, which is then sent back to the client.

22. **What are various databases supported by Django?**

Django supports various databases including PostgreSQL, MySQL, SQLite, Oracle, and others through its database backend API.

23. **What is the purpose of the session framework in Django?**

The session framework in Django allows you to store user-specific data across requests. It uses cookies or other methods to track sessions and associate data with individual users.

24. **Explain the way of using cookies for session management in Django?**

Django uses cookies to store a session ID on the client side. This session ID is then used to retrieve session data stored on the server side.

25. **Explain the purpose of middleware in Django?**

Middleware in Django is a framework of hooks into Django's request/response processing. It's a lightweight, low-level plugin system for globally altering Django's input or output.

26. **What is the context object in Django?**

The context object in Django is a dictionary-like object that is passed to the template renderer and contains the data to be displayed in the template.

27. **What is the purpose of `django.shortcuts.render()` function?**

`django.shortcuts.render()` is a shortcut function that simplifies the process of rendering a template with a given context and returning an `HttpResponse`.

28. **Explain the need for the `settings.py` file in the Django directory structure?**

The `settings.py` file in Django contains all the configuration settings for your Django project, including database settings, middleware, installed apps, static files configuration, and much more.

29. **How to view all items of the model?**

    You can view all items of a model by querying the database using Django's ORM. For example, `ModelName.objects.all()` will return all objects of the specified model.

30. **How to filter items from the model?**

    You can filter items from a model by using the `filter()` method of the model's manager. For example, `ModelName.objects.filter(field_name=value)` will return objects where the specified field matches the given value.