Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern, but in Django's terminology, it's referred to as the Model-View-Template (MVT) pattern. Here are some key rules and best practices when working with Django:

Don't Repeat Yourself (DRY): Django follows the DRY principle, which means avoiding redundancy in your code. Reuse components, templates, and avoid duplicating code.

Convention Over Configuration: Django relies on sensible defaults and conventions. Follow the naming conventions for models, views, and templates. This helps in making your code more readable and maintainable.

Use Django's ORM (Object-Relational Mapping): Django provides a powerful and easy-to-use ORM for database interactions. Leverage models to define your data structures and relationships.

Admin Interface: Django includes a built-in admin interface. Make use of it for managing your application's data easily. Customize the admin interface as needed.

URLs Mapping: Define clear and meaningful URLs for your views. Use the Django URL dispatcher to map URLs to views efficiently.

Template System: Take advantage of Django's template system to separate the presentation layer from the business logic. Use template inheritance to create a consistent layout across your application.

Middleware: Django middleware allows you to process requests globally before they reach the view. Use middleware for tasks such as authentication, logging, or modifying the request/response.

Security: Follow Django's security best practices. This includes using built-in tools for preventing common security issues like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Static Files and Media: Use Django's built-in tools for handling static files (CSS, JavaScript) and media files (uploads). Configure settings appropriately for production and development environments.

Testing: Write tests for your Django applications. Django provides a robust testing framework, and writing tests helps ensure the stability and correctness of your code.

Settings Module: Keep your project settings organized. Use separate settings files for development, testing, and production environments.

Version Control: Use version control systems like Git to manage your Django project. This helps in tracking changes, collaborating with others, and rolling back to previous versions if needed.

Documentation: Document your code, especially models, views, and complex functionalities. This makes it easier for other developers (and yourself) to understand and maintain the codebase.

Middleware Order: The order of middleware matters. Be mindful of the order in which middleware is defined in your MIDDLEWARE setting, as it can affect the behavior of your application.

Optimization and Caching: Optimize your queries and use caching where appropriate to improve the performance of your Django application.