Lesson 4

User Authentication

Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions.

Lesson 4

Authentication

User authentication enables secure user logins and registrations, protecting your application from unauthorized access.

Management

Django provides built-in tools for user management, including user creation, editing, password resets, and group assignment.

Password Management

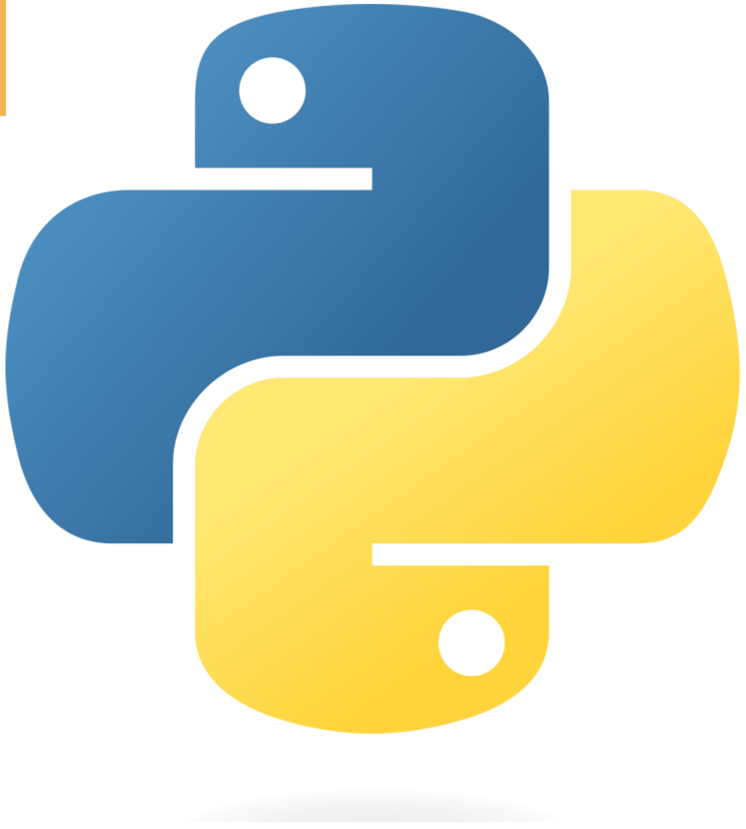
- **Hashing:** Django uses strong hashing algorithms to securely store passwords, preventing them from being stored in plain text.

- **Salt:** Salting adds random values to passwords before hashing, making them even more secure.

Lesson 4

Password Management

- **Password Reset:** Django provides a built-in feature for users to reset forgotten passwords through email verification.



Session Management

Django manages user sessions to track logged-in users and maintain their authentication state.

Lesson 4

Authorization

User authorization controls which actions users can perform within your application.

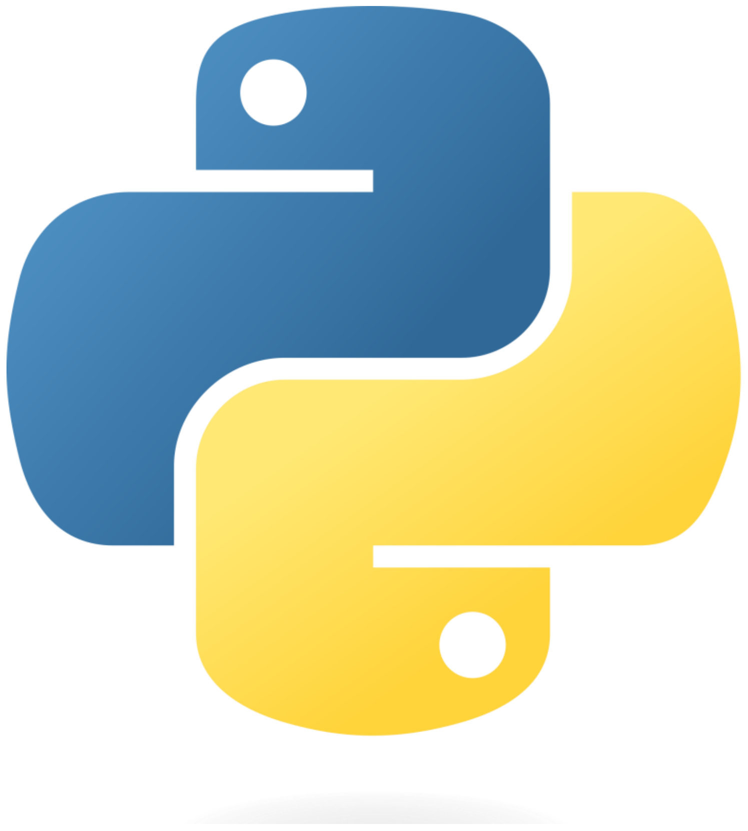
Permission

Permissions define specific actions that users can perform, such as creating, modifying, or deleting objects.

Lesson 4

Groups

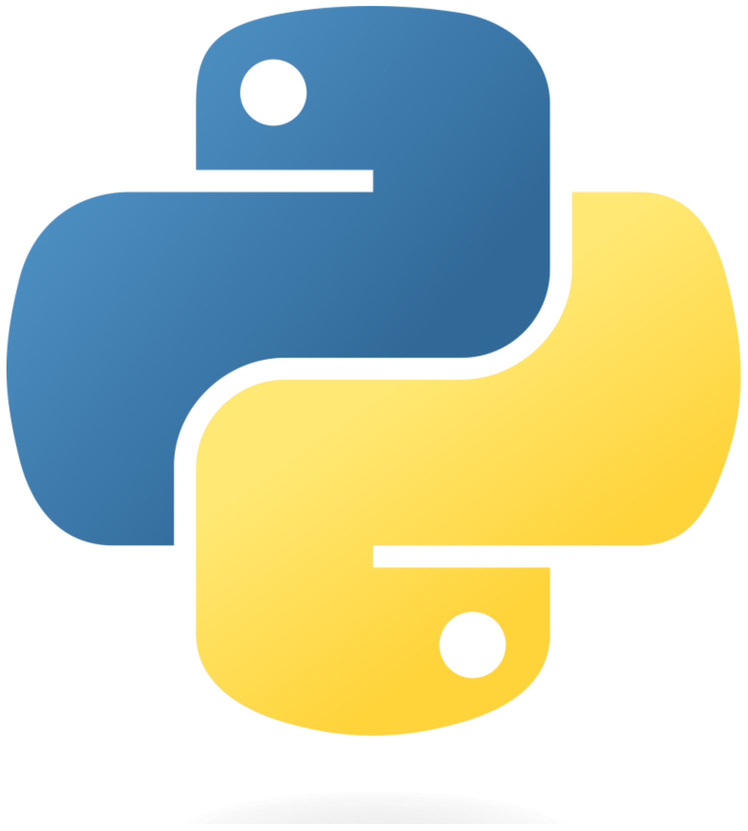
Groups allow you to organize users with similar permissions, simplifying the management of user access levels.



Lesson 4

Customize Auth.

Django's authentication system is flexible, allowing you to create custom authentication backends to integrate with different login methods.



Third-party Integration

Integrate social logins, such as Google, Facebook, and Twitter, for easy user access.

User Model

The **User model** is the foundation of authentication. It represents users in your application and stores essential information like username, email, and password.

User Model

The primary attributes of the default user are:

- username
- password
- email
- first_name
- last_name

AbstractUser

AbstractUser is a convenient base class that provides a streamlined way to create custom User models.

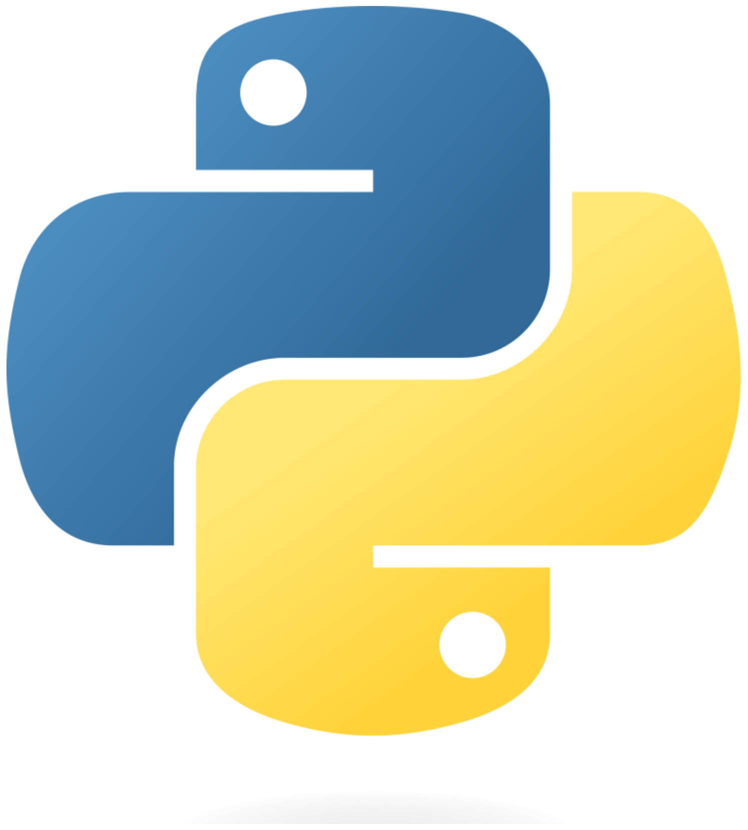
It offers pre-defined fields and methods for common user attributes like usernames, email addresses, and password management.

AbstractUser

Inherit from **AbstractUser** and override existing methods or add new fields to tailor the User model to your application's requirements.

BaseUserManager

Override methods like **create_user** and **create_superuser** to add custom validation logic, such as ensuring unique usernames or email addresses.



Lesson 4

BaseUserManager

Extend the **User model** with additional fields like phone number, address, or user roles to store essential user information.