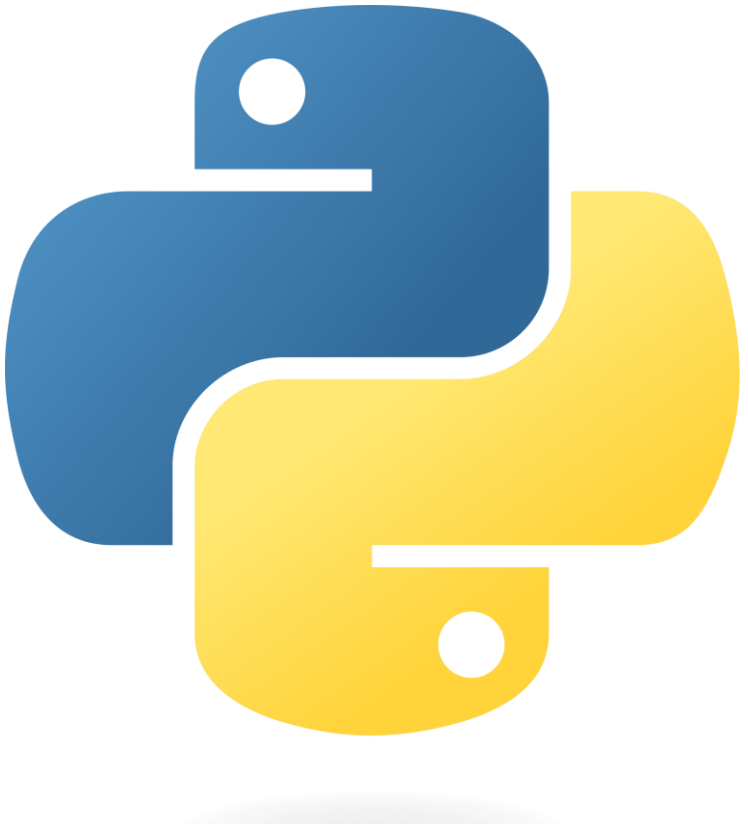
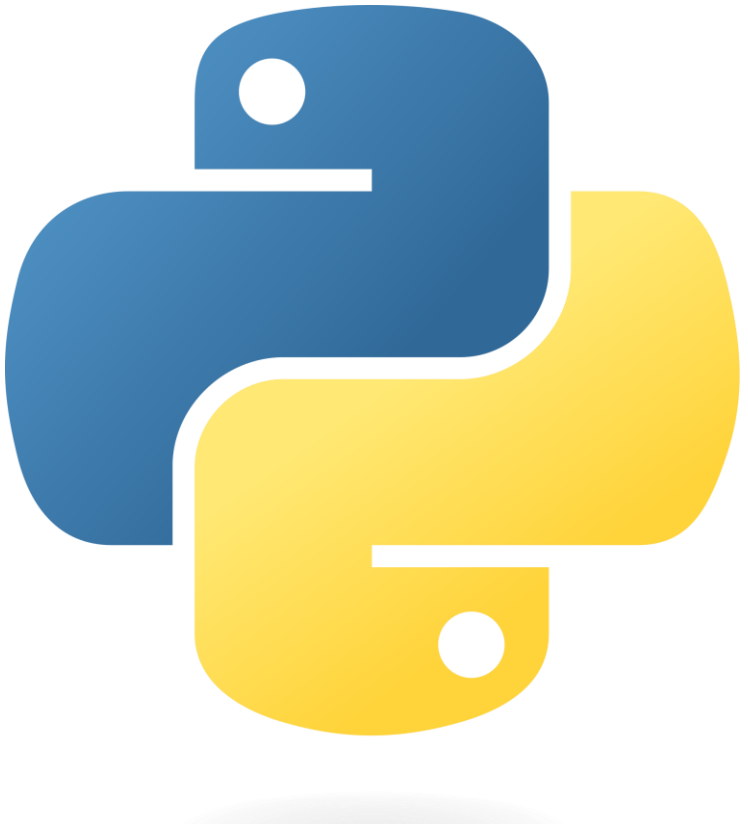
Django ORM

The Django Object-Relational Mapping (ORM) is a powerful tool that simplifies database interactions, allowing developers to manage data using Python code instead of complex SQL queries.



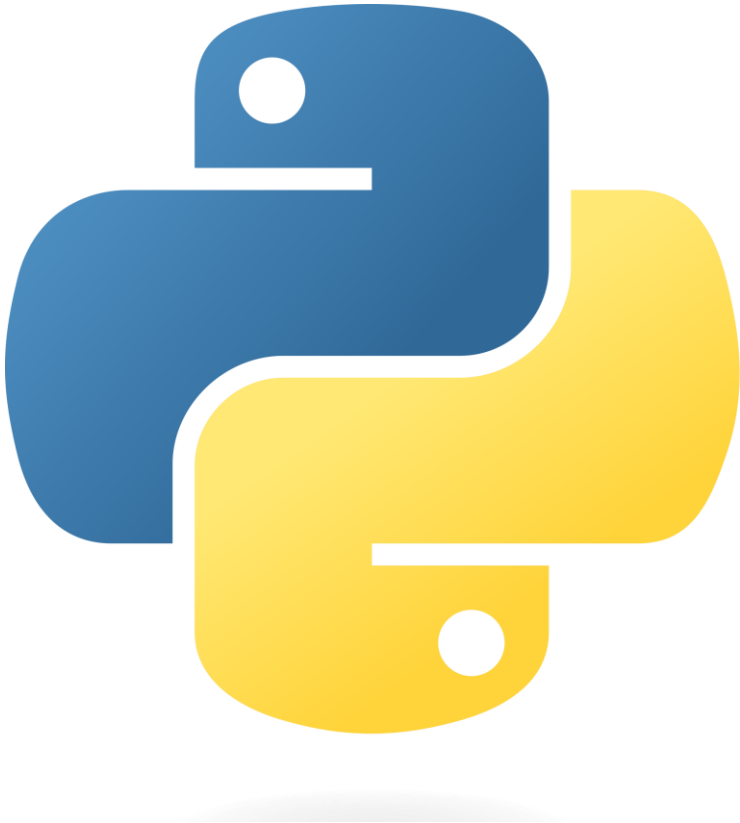
Django ORM

With its intuitive syntax and versatile features, the Django ORM empowers developers to build scalable and maintainable web applications.



Defining Models

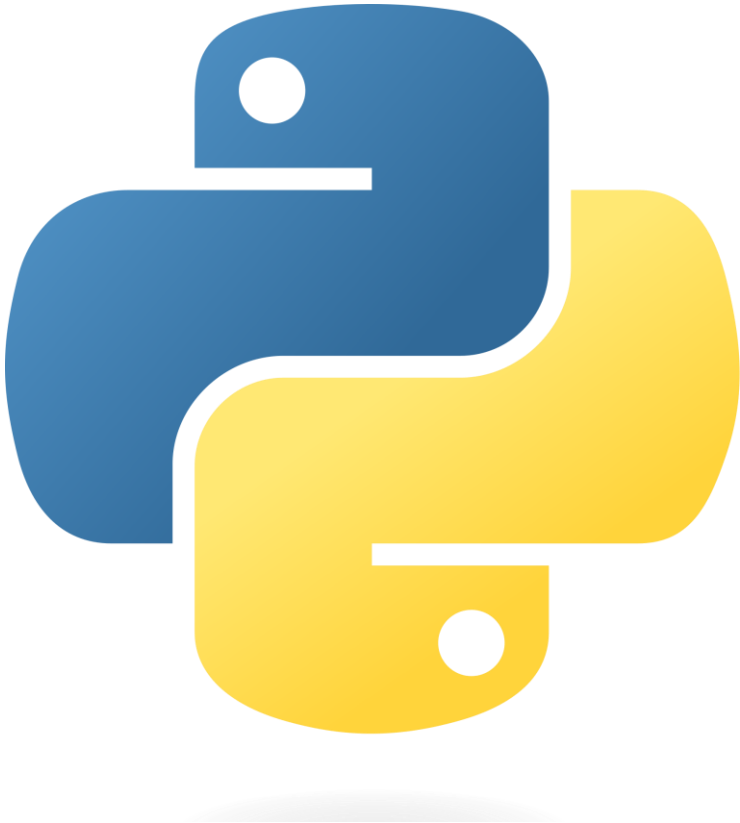
Intuitive Model Definitions : Django's ORM enables developers to define database models that closely reflect the business logic, making the codebase more readable and maintainable.



Defining Models

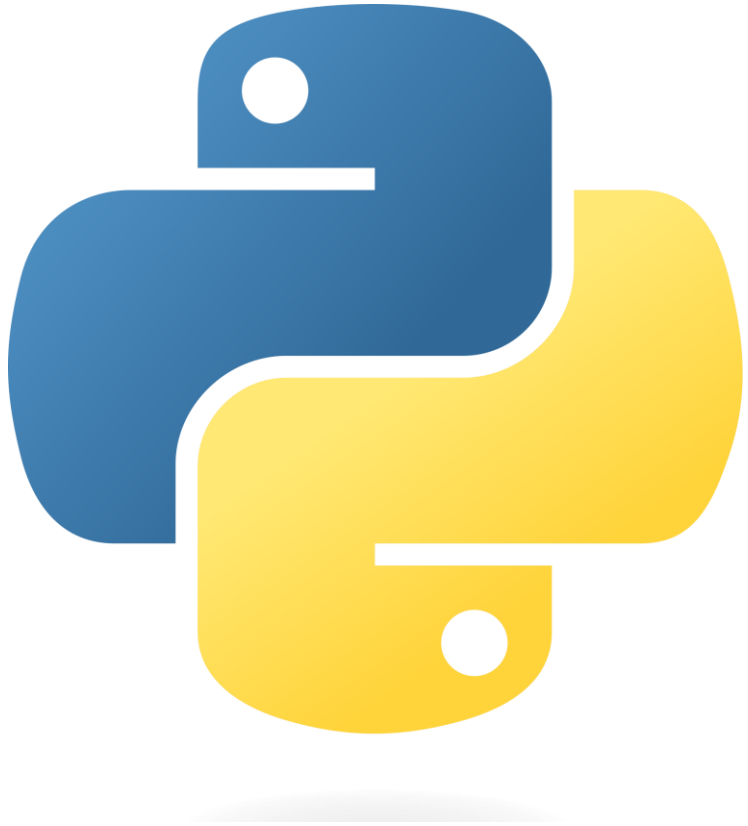
Automatic Table Creation: The ORM handles the translation of model definitions into SQL, automatically creating the necessary database tables and schema.

Flexible Field Types: The ORM supports a wide range of field types, allowing developers to accurately represent the data structure of their application.



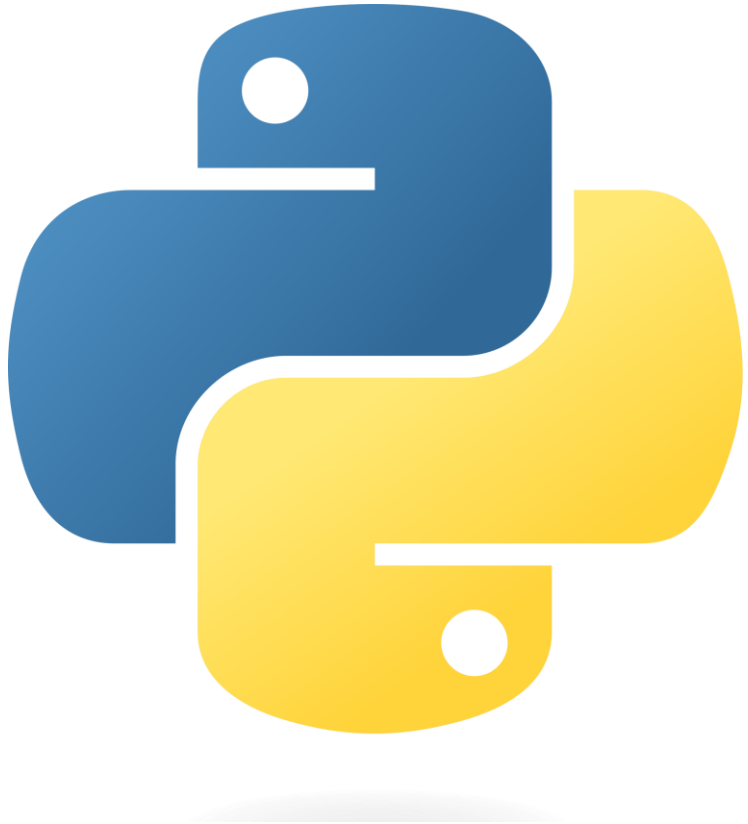
Relationships

In Django, relationships between models are defined using **ForeignKey**, **OneToOneField**, and **ManyToManyField**. These relationships efficiently connect and query data across database tables, ensuring data integrity and coherence.



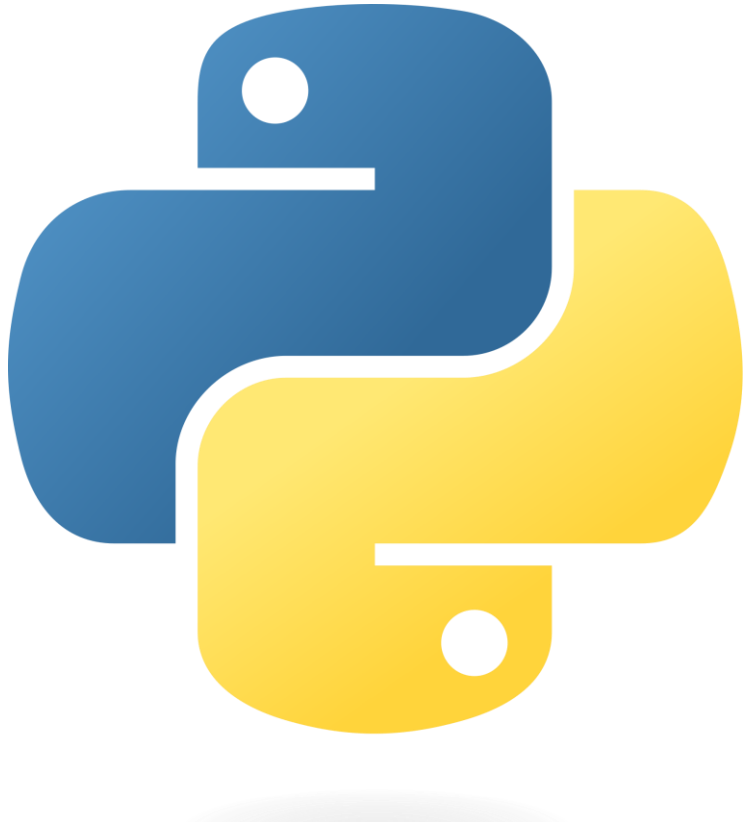
ForeignKey

A **ForeignKey** in Django is a field used to create a one-to-many relationship between models, linking each record in the child model to a single record in the parent model.



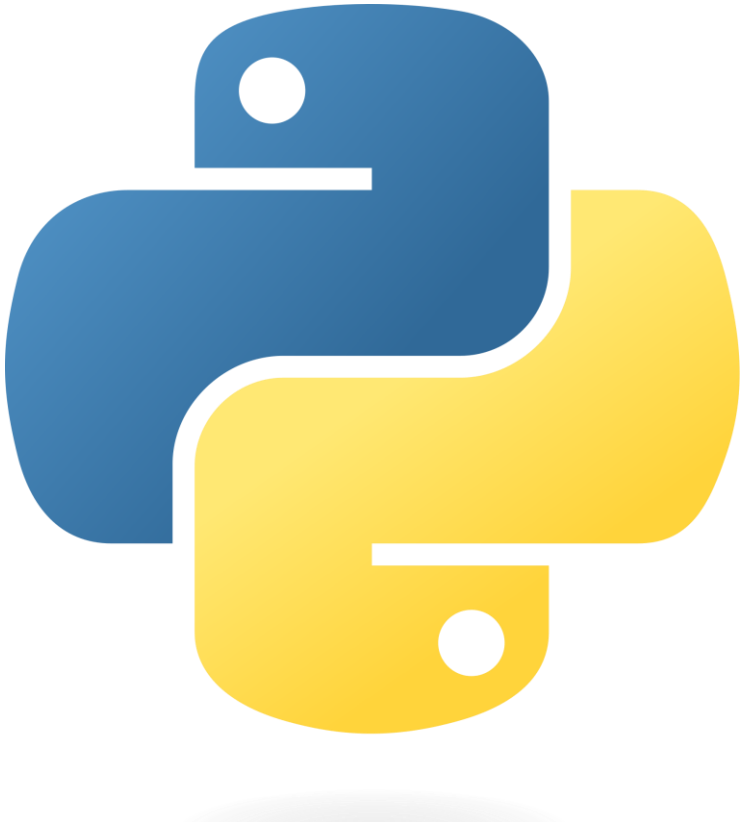
ManyToManyField

A **ManyToManyField** in Django creates a bidirectional relationship where each record in one model can be linked to multiple records in another, and vice versa.



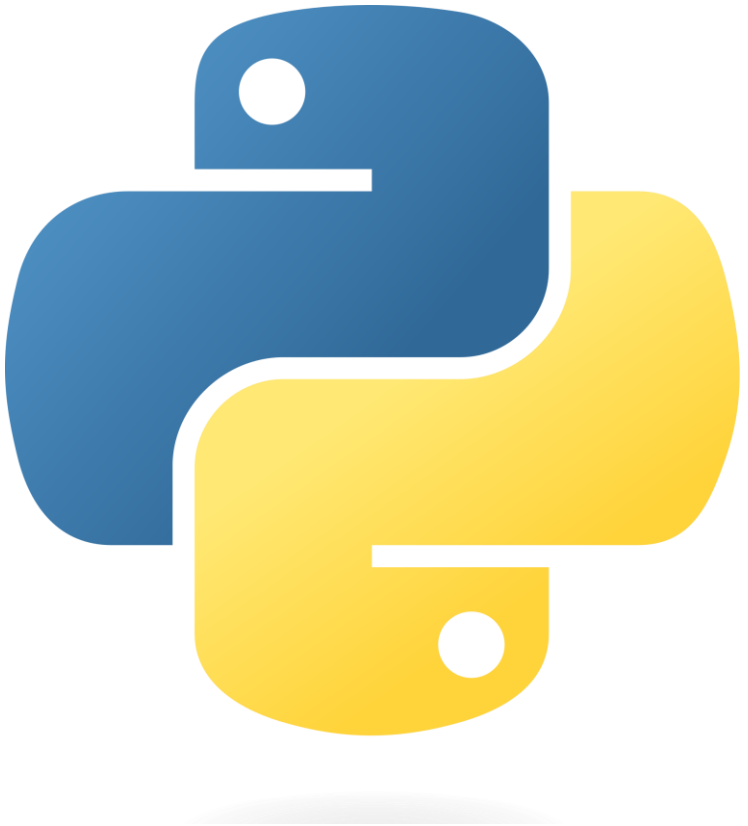
OneToOneField

A **OneToOneField** in Django establishes a one-to-one relationship between models, meaning each record in one model is linked to a single unique record in another model.



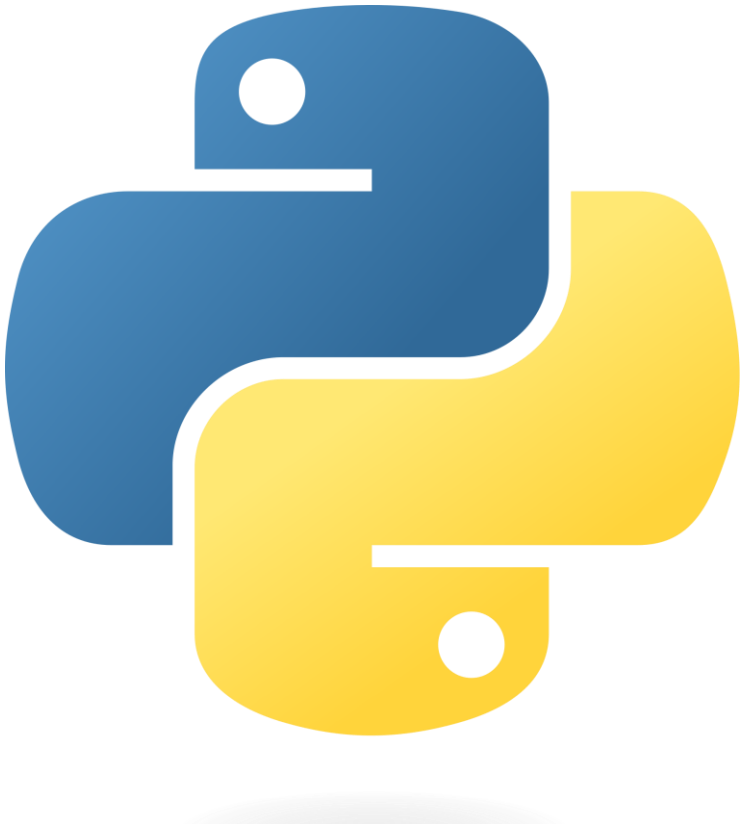
Migrations

- Automatic Schema Changes
- Versioning and Rollbacks
- Seamless Deployment
- Data Preservation



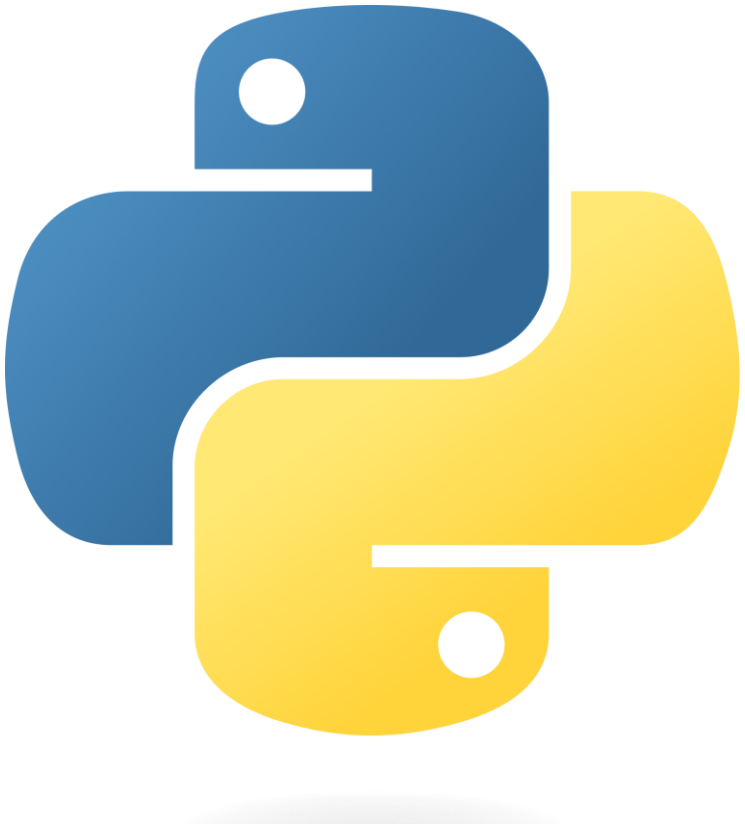
Querying

Django ORM simplifies database interactions by providing an intuitive API for filtering, ordering, and aggregating data, allowing developers to write cleaner, more readable code.



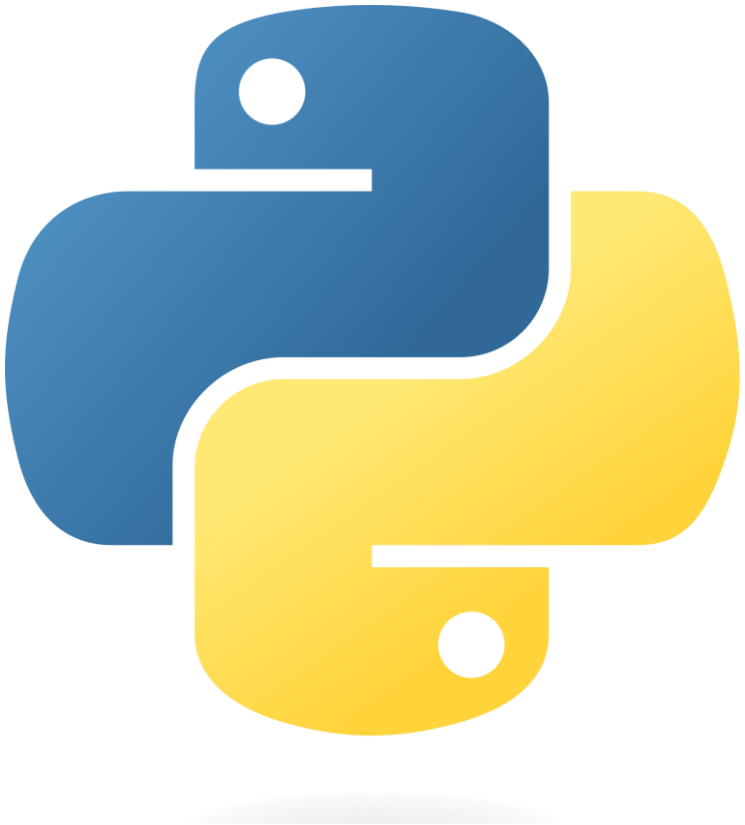
Filtering

The **ORM's** query API allows for advanced filtering, including complex boolean logic, range-based queries, and field lookups.



Sorting

Developers can easily sort query results based on one or more fields, enabling dynamic sorting and presentation of data.



Aggregation

The **ORM** provides powerful aggregation functions, such as count, sum, and average, that can be used to perform complex data analysis.