Model

```
from django.db import models
from django.core.validators import (
    MaxValueValidator,
    MinValueValidator,
class Author(models.Model):
    # A very simple model example
    name = models.CharField(max_length=100)
class Book(models.Model):
    # Use a "ForeignKey"
                         for a ManyToOne relationshi
    author = models.ForeignKey(
        Author,
        on_delete=models.CASCADE,
    title = models.CharField(max_length=100)
    release_date = models.DateField()
    # Very useful fields: "created" stores when it
    # originally was created, "last_updated" stores
    # whenever edited
    created = models.DateTimeField(
        auto_now_add=True)
    last_updated = models.DateTimeField(
        auto_now=True)
    # Multiple-choice fields, pattern is:
    # "internal code, human-readable label"
    CATEGORIES = (
    ("fict", "Fiction"),
        ("nonfict", "Non-fiction"),
    category = models.CharField(
        max_length=10,
        default="fict'
        choices=CATEGORIES,
    )
    # example with custom validators
    num_stars = models.IntegerField(
        validators=[MaxValueValidator(5)
                    MinValueValidator(1)],
    )
    def __str__(self):
                           # Define __str__ to give
        return self.title # string description
# ManyToMany relationships
class ReadingList(models.Model):
    books = models.ManyToManyField(Book)
Key words
```

id Automatically incrementing integer included with all Models

queryset Django terminology for the list-like data returned from database

CRUD Create, Read, Update, Delete - The four main operations of web application development

MIGRATION WORK-FLOW

```
# Generate migrations for recent model changes
python manage.py makemigrations
# Double check where we're at
python manage.py showmigrations
# Apply the generated migrations
python manage.py migrate
```

DB RELATIONSHIPS

One-to-many An instance can be associated with an arbitrary number of other instances. Example: Artist can have released multiple Albums.

Many-to-many Freely associate any number to any number. Example: Album can have many Tags, and each Tag can have many albums. For Twitter, User can follow an arbitrary number of other Users.

One-to-one For every insteance of one model, there exists exactly one instance of another, effectively "splitting a model into two". Example: Album with AlbumArtwork.

DJANGO CRUD EXAMPLES

book.delete()

ORM operations like this typically go in your views.py to accomplish the business logic necessary based on your application goals.

```
from .models import Book
### CREATE
book = Book.objects.create(
    title="Great Expectations",
   num_stars=4,
### READ
# Get all fiction books to loop through
fiction_books = Book.objects.filter(category="fict")
# Get all 4+ star books, newest first
new_good_books = (
    Book.objects.filter(num_stars__gt=3)
    .order_by("-date")
)
### UPDATE
book = Book.objects.get(title="Great Expectations")
book.num_stars = 5 # Updates a single property
                    # Saves the change to the DB
book.save()
nonfict = Book.objects.filter(category="nonfict")
nonfict.update(num_stars=5) # Updates all books
### DELETE
book = Book.objects.get(title="Great Expectations")
```

bad_books = Book.objects.filter(num_stars__lt=3) bad_books.delete() # Delete every 1 or 2 star book

Only delete a single book