1. Django Intro 73% (/m/72/5493)
2. Django 29% (/m/72/5494)
ORMs (/m/
Models/Migrations (/m/
Foreign Key Relationships (/m/
Many to Many (/m/
Shell/Commands ...
ORM Review (/m/
Users (/m/
Dojo Ninjas (/m/
Books/Authors
Likes/Books
Sports ORM (Optional)
Sports ORM II (Optional)
Adding Validations
Named Routes and Forms
Semi-Restful Users
Courses
Amadon
Bcrypt
Security Lesson
Login and Registration
The Wall
User Dashboard (Optional)
Project Guideline (important!)
Belt Reviewer
Chapter Survey
3. Django and Ajax 0% (/m/72/5495)

Django Shell

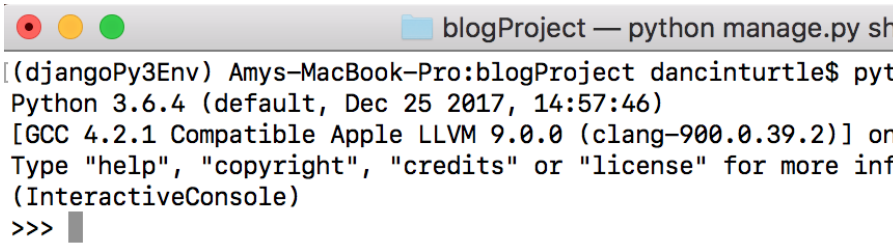
CHECKLIST

Instead of building out our app to run and serve data, we can simply stop here and open Django
We'll be using the shell to experiment with writing queries using Django's ORM. Django shell is a
your project's *manage.py* to enter a command-line interface that allows you access to your proje
connected to it.

In terminal, from your project's root directory, enter the following:

```
> python manage.py shell
```

Your terminal output should look like so:



Once you're in shell, you'll have access to the functions contained in your files. However, just like
have to import the modules (files) that you need. **Note:** be sure to replace anything with {{}}, inc
project and app names for your project. Enter the following with the appropriate values replaced

```
>>> from apps.{{app_name}}.models import *
```

Caution: Since models.py contains the classes you wrote, importing everything in models.py ma
generally when you're importing other libraries/modules, Django community discourages the pra
explanation can be found at https://stackoverflow.com/questions/2360724/what-exactly-does-i

(https://stackoverflow.com/questions/2360724/what-exactly-does-import-import). Now, for the have multiple classes, and to make your life easier as you practice on the shell, you could continu do be cautious when importing other files.

ORM Commands

Now that you have imported your models, you can use the model object to insert, delete, and m example,

1. Creating a new record
    1. *Blog.objects.create({{field1}}="{{value}}", {{field2}}="{{value}}", etc)* # creates a new re
        - Blog.objects.create(name="Star Wars Blog", desc="Everything about Star Wars") :
        - Blog.objects.create(name="CodingDojo Blog") # creates a new blog record with t
    2. Alternative way of creating a record
        1. b = Blog(name="Disney Blog", desc="Disney stuff")
        2. b.name = "Disney Blog!"
        3. b.desc = "Disney stuff!!!"
        4. b.save()
2. Basic Retrieval
    1. Blog.objects.first() - retrieves the first record in the Blog table
    2. Blog.objects.last() - retrieves the last record in the Blog table
    3. Blog.objects.all() - retrieves all records in the Blog table
    4. Blog.objects.count() - shows how many records are in the Blog table
3. Displaying records
    1. Blog.objects.first().**__dict__** - shows all the values of a single record/object as a dictiona
    2. Blog.objects.all().**values()** - as shown in the videos, shows all the values of a QuerySet (records)
4. Updating the record - once you obtain an object that has the record you want to modify, use For example
    1. b = Blog.objects.first() # gets the first record in the blogs table
    2. b.name = "CodingDojo Blog" # set name to be "CodingDojo Blog"
    3. b.save() # updates the blog record
5. Deleting the record - use delete(). For example
    1. b = Blog.objects.get(id=1)
    2. b.delete() # deletes that particular record
6. Other methods to retrieve records
    1. Blog.objects.get(id=1) - retrieves where id is 1; `get()` retrieves one and only one record. fewer than or more than one match.
    2. Blog.objects.filter(name="mike") - retrieves records where name is "mike"; returns multi
    3. Blog.objects.exclude(name="mike") - opposite of filter; returns multiple records
    4. Blog.objects.order_by("created_at") - orders by created_date field
    5. Blog.objects.order_by("-created_at") - reverses the order
    6. Blog.objects.raw("SELECT * FROM {{app_name}}_{{class/table name}}") - performs a r
    7. Blog.objects.first().comments.all() - grabs all comments from the first Blog
    8. Blog.objects.get(id=15).comments.first() - grabs the first comment from Blog id = 15
    9. Comment.objects.get(id=15).blog.name - returns the name of the blog where Comment
7. Setting Relationship
    1. Comment.objects.create(blog=Blog.objects.get(id=1), comment="test") - create a new c blog points to Blog.objects.get(id=1).

Conditions

You can do a more complicated search than just if a given field is equal to a given value. Instead a keyword argument to `.get`, `.filter`, or `.exclude`, you'd pass the field name__lookup type (t known as a "dunder" for people on-the-go).

For example

- Admin.objects.filter(first_name__startswith="S") - filters objects with first_name that start
- Admin.objects.exclude(first_name__contains="E") - excludes objects where first_name co
- Admin.objects.filter(age__gt=80) - filters objects with age greater than 80

BACK TO TRACKS

Combining queries

Queries can be chained together:

```
admin = Admin.objects.filter(last_name__contains="o").exclude(first_name__contains="O"
```

```
admin = Admin.objects.filter(age__lt=70).filter(first_name__startswith="S")
```

CHECKLIST

If it's the same type of query, instead of being chained you can add multiple arguments to the fu

```
admin = Admin.objects.filter(age__lt=70, first_name__startswith="S")
```

These are cases where the conditions are joined with AND, as in, all users younger than 70 AND you want OR, as in users who are younger than 70 OR whose first_name starts with "S", you can

```
admin = Admin.objects.filter(age__lt=70)|Admin.objects.filter(first_name__startswith="S
```

References

1. https://tutorial.djangogirls.org/en/django_orm/ (https://tutorial.djangogirls.org/en/django_o

Helpful tip

In regard to the default display when printing objects, we could create __str__ or __repr__ meth want the objects to print. This is pretty handy and shows how we can leverage some of python's easier.

```
class Blog(models.Model):
    name = models.CharField(max_length=255)
    desc = models.TextField()
    created_at = models.DateTimeField(auto_now_add = True)
    updated_at = models.DateTimeField(auto_now = True)
    def __repr__(self):
        return "<Blog object: {} {}>".format(self.name, self.desc)
```

iPython - nicer looking shell

Also, if you would like, you could also install ipython (pip install ipython). This replaces the defa (TAB indent works, line numbers, syntax highlighting, etc).

PREVIOUS (/M/72/5494/35477) NEXT (/M/72/54