Instructions for class

Installing Django

- Please open terminal on your computer
- You should first create a virtual environment on your personal account
    - `$ virtualenv yourname`
    - `$ cd yourname`
    - `$ source ./bin/activate`
    - After running this command, you should see (yourname) at the beginning of your command line
- Now use pip to install Django on your computer
    - `$ pip install django`

Start your first project

- After you have installed Django, make a random folder, eg. django, and `cd` into that directory where you'd like to store your code, then run the following command to start your very first Django project
    - `$ mkdir django`
    - `$ cd django`
    - `$ django-admin startproject mysite`
- Let's verify your Django project works. Change into the outer mysite directory, do an initial migrate, and run the following commands:
    - `$ cd mysite`
    - `$ python manage.py runserver`
    - You'll see the following output on the command line:
    -
      ```
      Performing system checks…

      System check identified no issues (0 silenced).

      You have unapplied migrations; your app may not work properly until
      they are applied.
      Run 'python manage.py migrate' to apply them.

      May 04, 2015 - 16:06:42
      Django version 1.8.1, using settings 'mysite.settings'
      Starting development server at http://127.0.0.1:8000/
      Quit the server with CONTROL-C.
      ```

    - Don’t worry about the migrations at this point. Now that the server’s running, visit http://127.0.0.1:8000/ with your Web browser. You’ll see a “Welcome to Django” page, in pleasant, light-blue pastel. It worked!
- Before we go any further into this, let’s first look at all the files created by this command.
    - Let’s look at what startproject created:
    - `mysite/`

```
db.sqlite3
manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

- Database
    - By default, the configuration uses SQLite. If you’re new to databases, or you’re just interested in trying Django, this is the easiest choice. SQLite is included in Python, so you won’t need to install anything else to support your database. When starting your first real project, however, you may want to use a more robust database like PostgreSQL, to avoid database-switching headaches down the road.
    - Some of these applications makes use of at least one database table, though, so we need to create the tables in the database before we can use them. To do that, run the following command:
        - `$ python manage.py migrate`

Start your first app

- Now that your environment – a “project” – is set up, you’re set to start doing work.
- To create your app, make sure you’re in the same directory as manage.py and type this command:
    - `$ python manage.py startapp books`
- That’ll create a directory books, which is laid out like this:
- `books/`

```
    __init__.py
    admin.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

- This directory structure will house the books app.

Swap out some files

- Now, download the files from **this link**, unzip it, and use it to replace the books folder in your directory. Then download **the database file,** and replace the database file in your root directory. This is an app that I wrote prior to class, that contains a certain view, two tables in models, and a couple templates.

Now, let's link the books app to our project

- Edit the `mysite/settings.py` file again, and change the `INSTALLED_APPS` setting to include the string 'books'. So it'll look like this:

```
● mysite/settings.py
● INSTALLED_APPS = (
      'django.contrib.admin',
      'django.contrib.auth',
      'django.contrib.contenttypes',
      'django.contrib.sessions',
      'django.contrib.messages',
      'django.contrib.staticfiles',
      'books',
  )
```

- Now Django knows to include the books app.
- Then go into `mysite/urls.py` file, and add a new urls

```
● urlpatterns = [
●     url(r'^admin/', include(admin.site.urls)),
●     url(r'^books/', include('books.urls')),
● ]
```

- Let's run another command:
- `$ python manage.py makemigrations books`
- You should see something similar to the following:

```
● Migrations for books:
    0001_initial.py:
      - Create model Author
      - Create model Books
      - Add field book to author
```

Take a look at the books app

- A model is the single, definitive source of truth about your data. It contains the essential fields and behaviors of the data you're storing.
- In our simple books app, we'll create two models: Book and Author. A Book has a name, a publisher, and a foreign key pointing to author. An Author has one field: the name of the author. Each Book is associated with an Author.
- MODELS
    - books/models.py
- URLS
    - mysite/urls.py
- VIEWS
    - books/views.py
- TEMPLATE
    - books/templates/base.html
    - books/templates/books.html

Activities

- Now the view is displaying only the first book in the database. Can you use a loop in your template and make some small changes to the getbooks function in views to see an unordered list of all the books?
- Resources:
    - Templage tages (loop) https://docs.djangoproject.com/en/1.8/ref/templates/builtins/#for
    - Retrieving objects https://docs.djangoproject.com/en/1.8/topics/db/queries/#retrieving-objects

- If you have finished the task above, can you write a new view that displays all the authors and connect it to a new url (for example, /books/authors.html) in urls.py?
    - Hint: Can you re-use books.html as your template for this view as well?