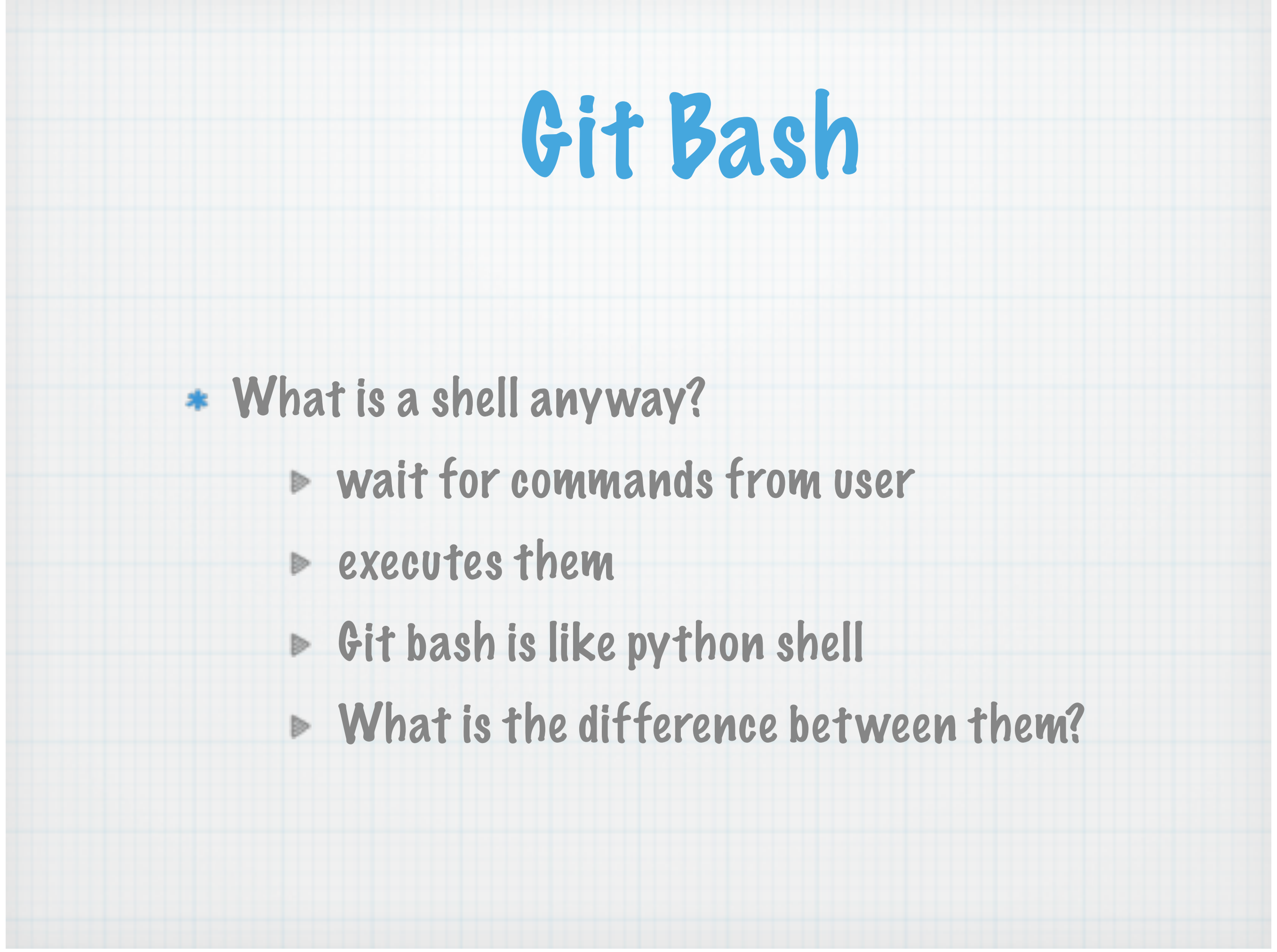
Python - Django Introduction

v1
01/12/2013

CoderDojo Malahide

Last session...

- we installed and configured machines
    - Git: powerful shell
    - Pip: package manager
    - Django: web app framework
    - Selenium: testing framework



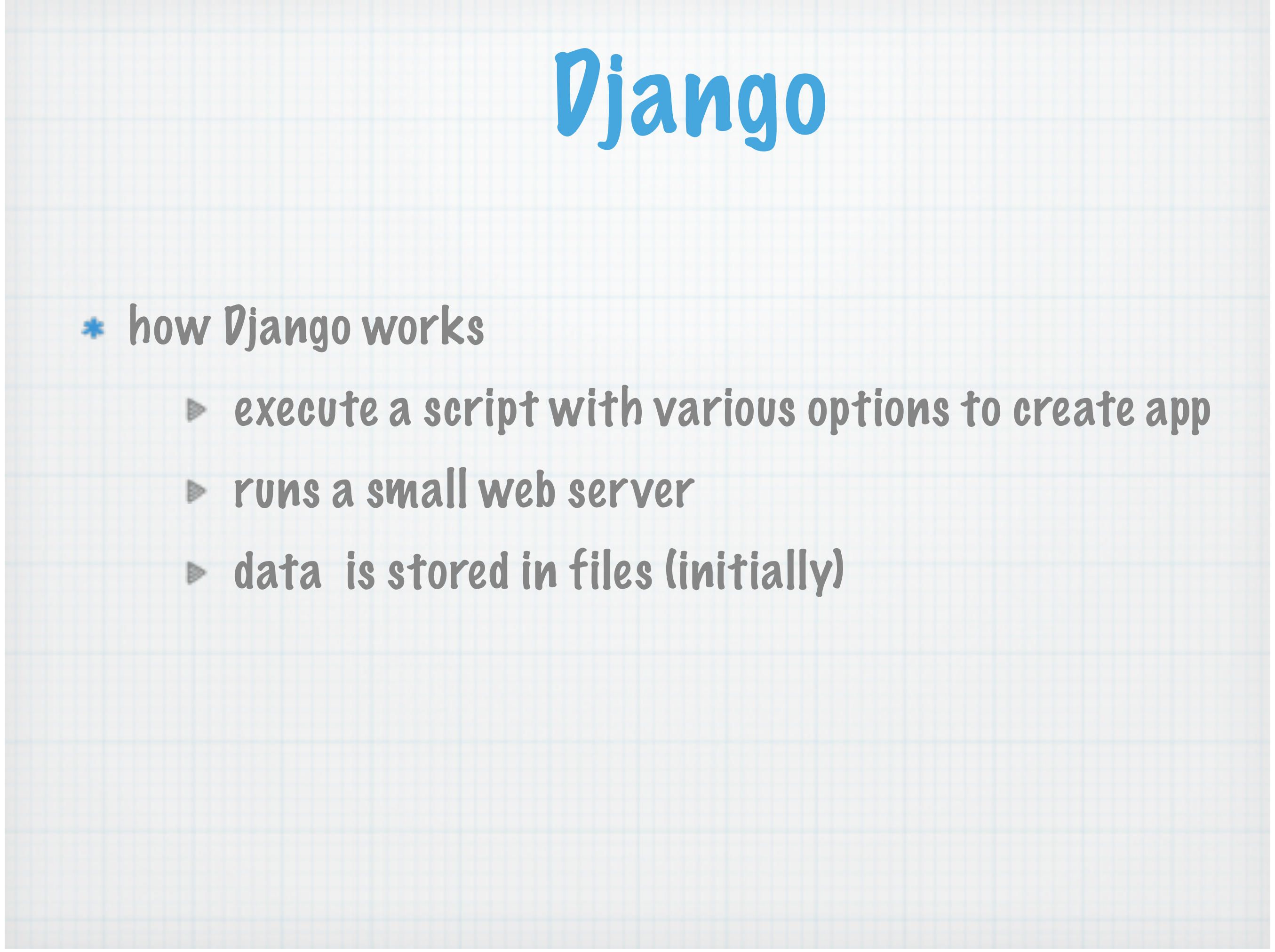
Git Bash

* What is a shell anyway?
    - wait for commands from user
    - executes them
    - Git bash is like python shell
    - What is the difference between them?

difference: language (shell commands vs Python)

Django / Selenium

- Django
  - web framework written in python
  - runs on your laptop locally
- Selenium
  - automated testing
  - write Python script
  - opens a browser, goes to a website, read from browser window



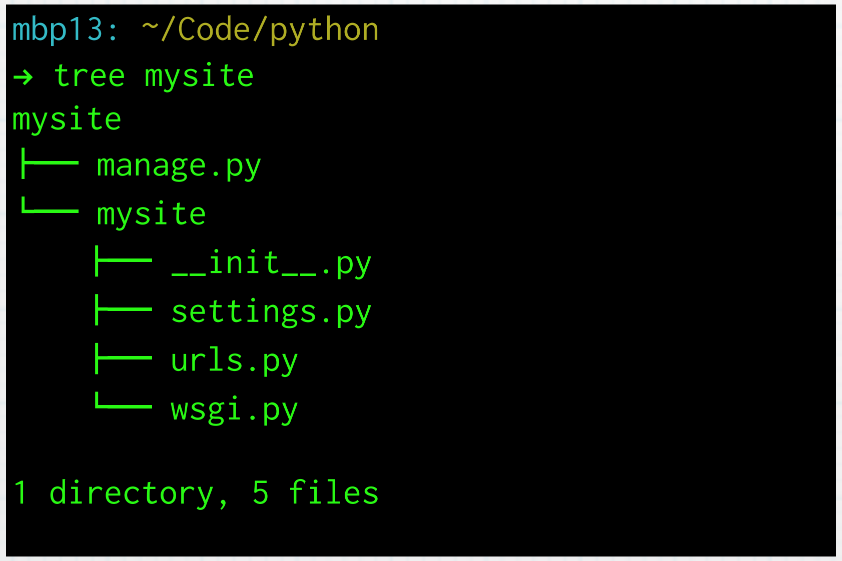
Django

- how Django works
  - execute a script with various options to create app
  - runs a small web server
  - data is stored in files (initially)

Default for data storage is sqlite3, default config stores data in file *db.sqlite3*

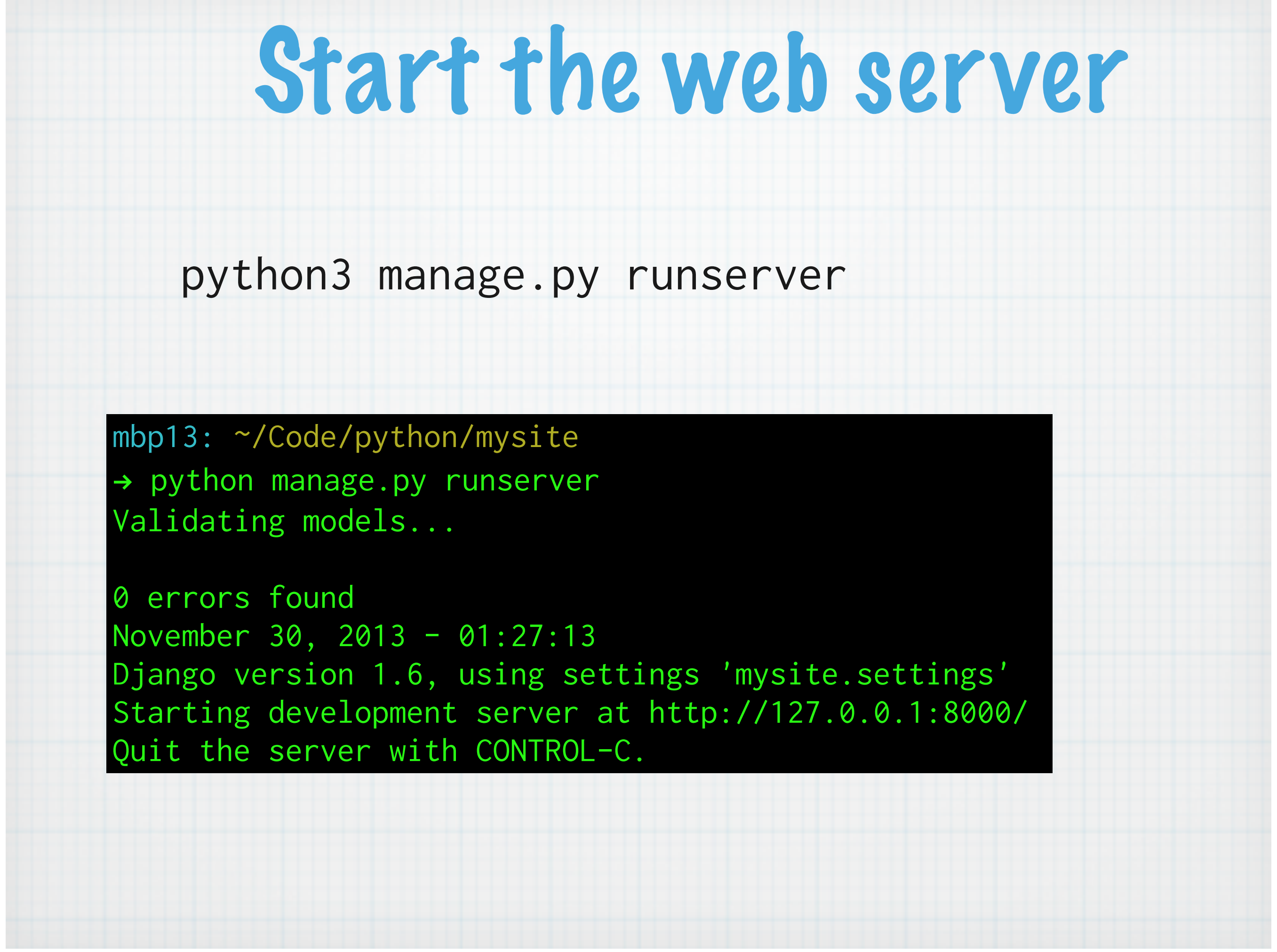
Create a Django Project

```
django-admin.py startproject mysite
```



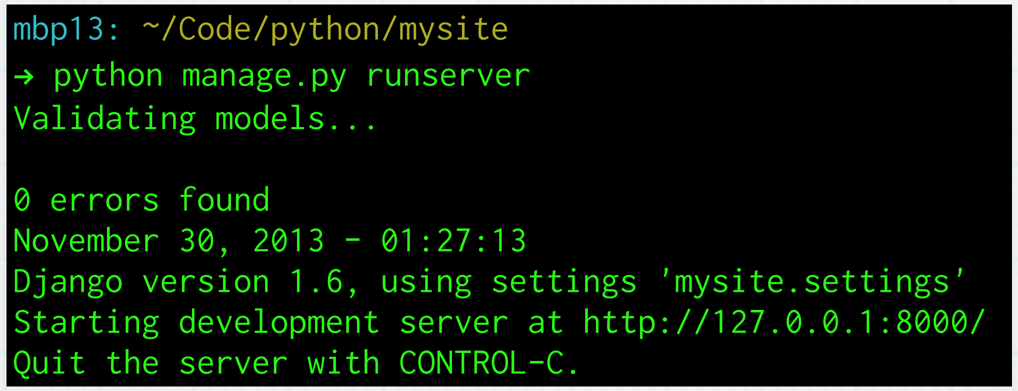
```
mbp13: ~/Code/python
➜ tree mysite
mysite
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 5 files
```

Navigate to this folder in Windows Explorer (or Mac Finder) and take a look at generated files. (Open in editor, e.g. IDLE or Notepad++)



Start the web server

```
python3 manage.py runserver
```



```
mbp13: ~/Code/python/mysite
➔ python manage.py runserver
Validating models...

0 errors found
November 30, 2013 - 01:27:13
Django version 1.6, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

http://127.0.0.1:8000

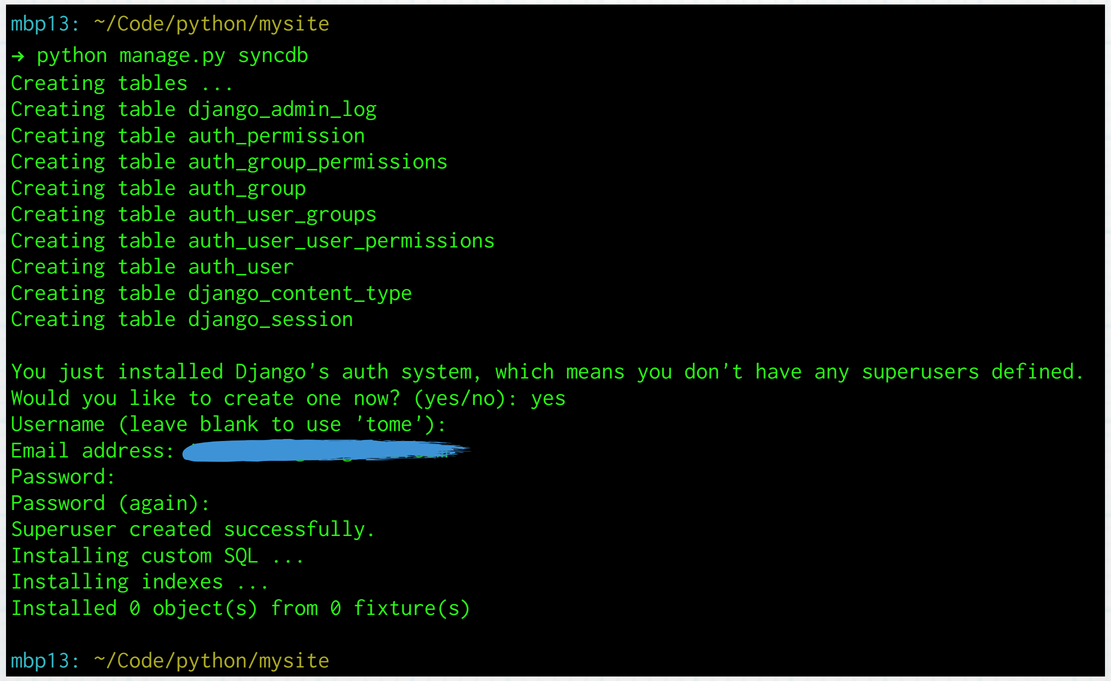
http - the protocol used by browsers

127.0.0.1 - aka *localhost*, this is an internal IP address of your computer

8000 - the port used for this application.

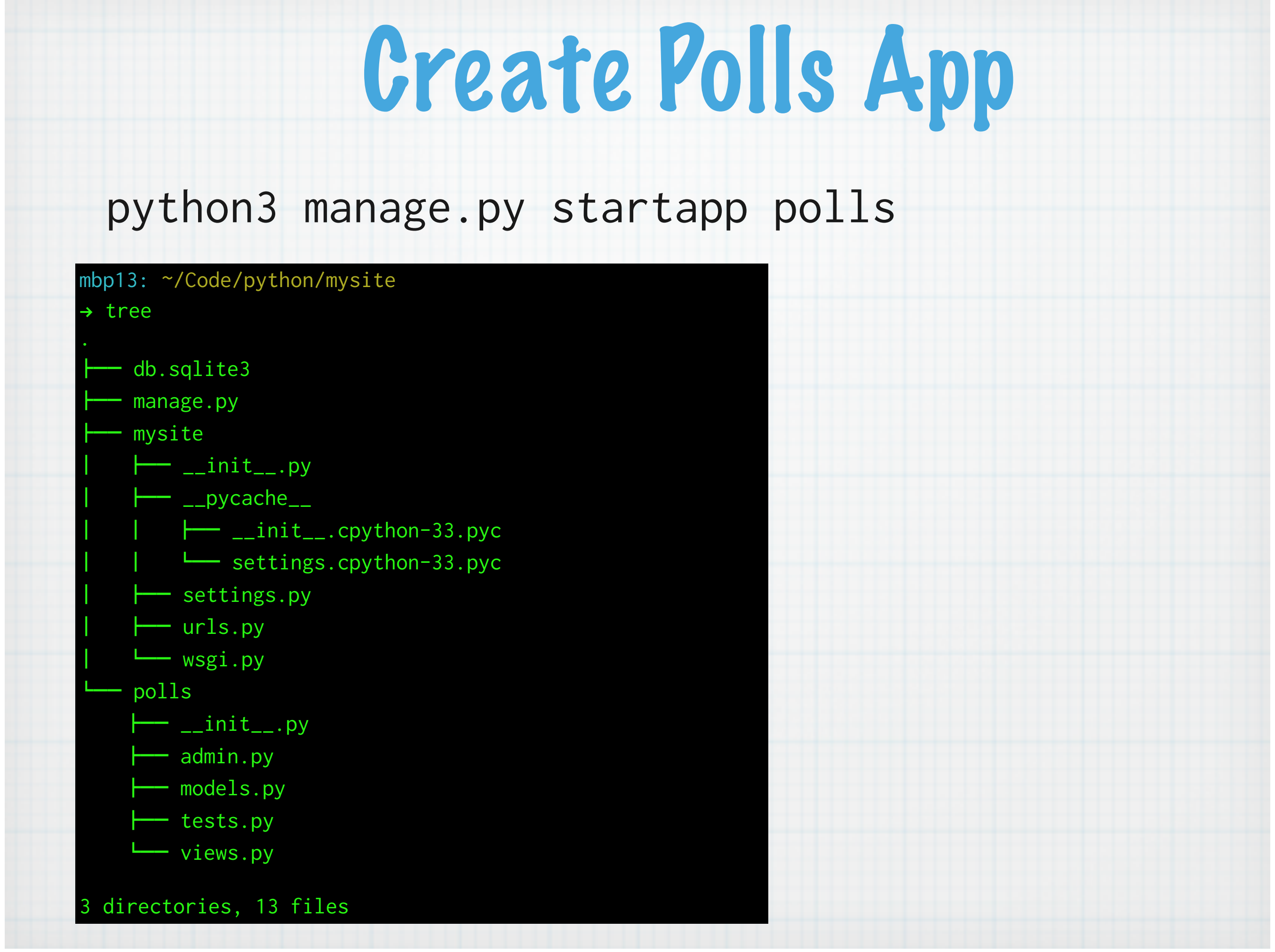
Initialise Database

```
python3 manage.py syncdb
```



```
mbp13: ~/Code/python/mysite
➔ python manage.py syncdb
Creating tables ...
Creating table django_admin_log
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_content_type
Creating table django_session

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'tome'):
Email address: [redacted]
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)

mbp13: ~/Code/python/mysite
```

remember username and password because you'll need them later

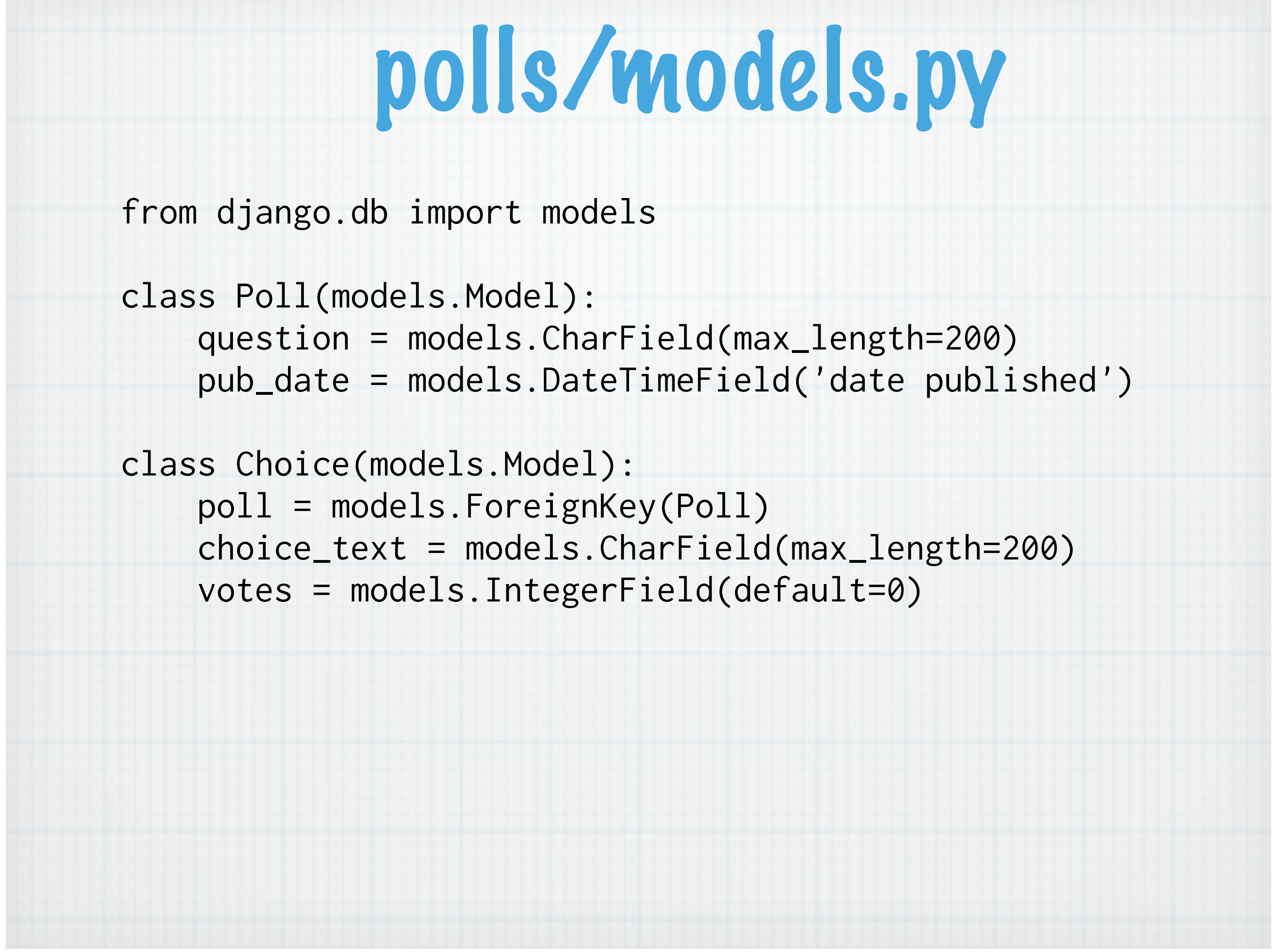


Create Polls App

```
python3 manage.py startapp polls
```

```
mbp13: ~/Code/python/mysite
→ tree
.
├── db.sqlite3
├── manage.py
├── mysite
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-33.pyc
│   │   └── settings.cpython-33.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── polls
    ├── __init__.py
    ├── admin.py
    ├── models.py
    ├── tests.py
    └── views.py

3 directories, 13 files
```

An *app* is part of a *site*. A site can have many apps - ours will have just one for now.



polls/models.py

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Type this code into file polls/models.py.

Model is a word for the data we will store on our site. Search for Model-View-Controller (MVC) if you want to know more.

Here we define the fields we want to store for each thing. A poll has a question and publishing date, a choice has a text and number of votes (and a link to a poll, meaning that a choice is always tied to a specific poll)

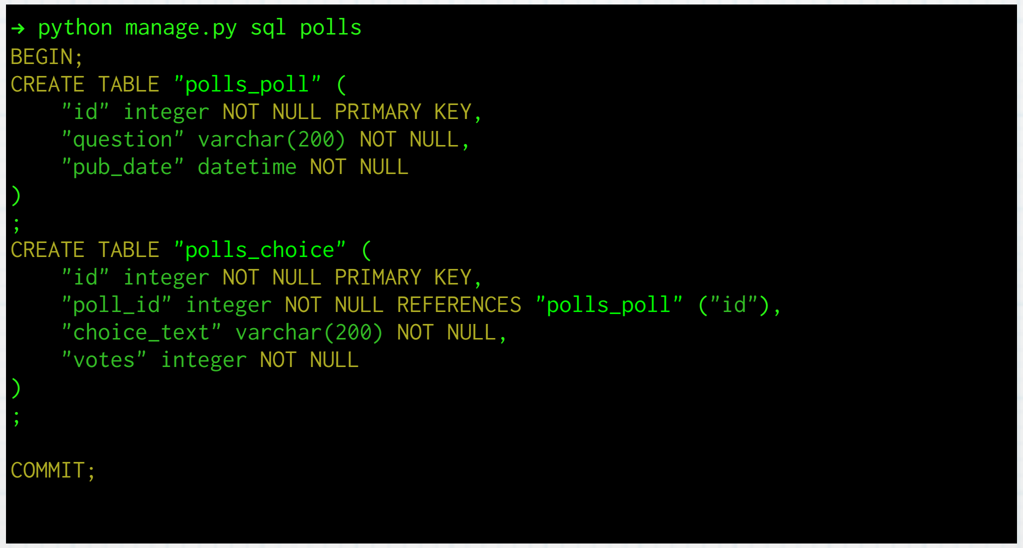
mysite/settings.py

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls'
)
```

Add the polls app

Activate the model

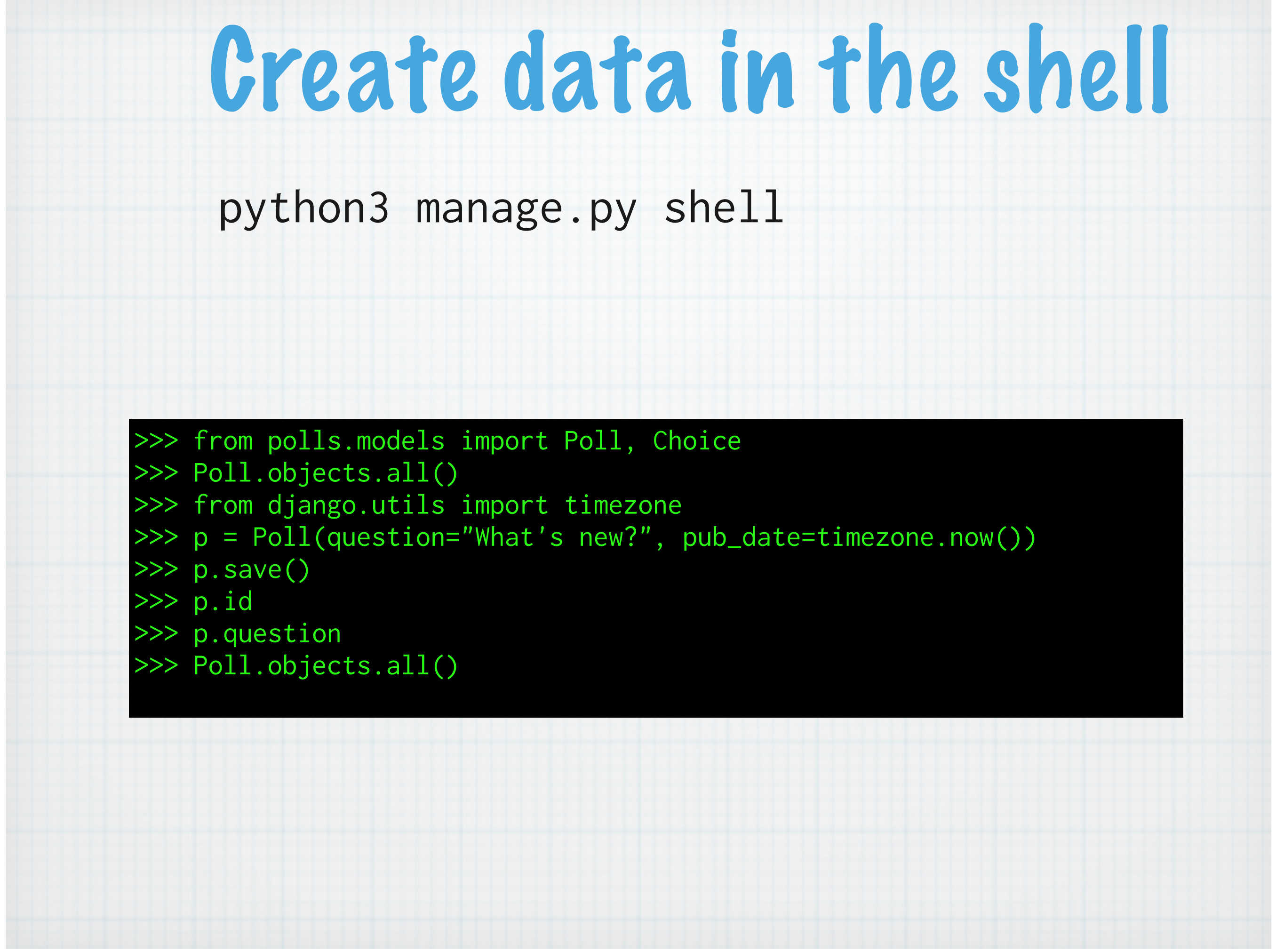
```
python3 manage.py sql polls

python3 manage.py syncdb
```



```
→ python manage.py sql polls
BEGIN;
CREATE TABLE "polls_poll" (
    "id" integer NOT NULL PRIMARY KEY,
    "question" varchar(200) NOT NULL,
    "pub_date" datetime NOT NULL
)
;
CREATE TABLE "polls_choice" (
    "id" integer NOT NULL PRIMARY KEY,
    "poll_id" integer NOT NULL REFERENCES "polls_poll" ("id"),
    "choice_text" varchar(200) NOT NULL,
    "votes" integer NOT NULL
)
;

COMMIT;
```

this creates the database, based on the code you wrote in models.py.

Any errors, check models.py

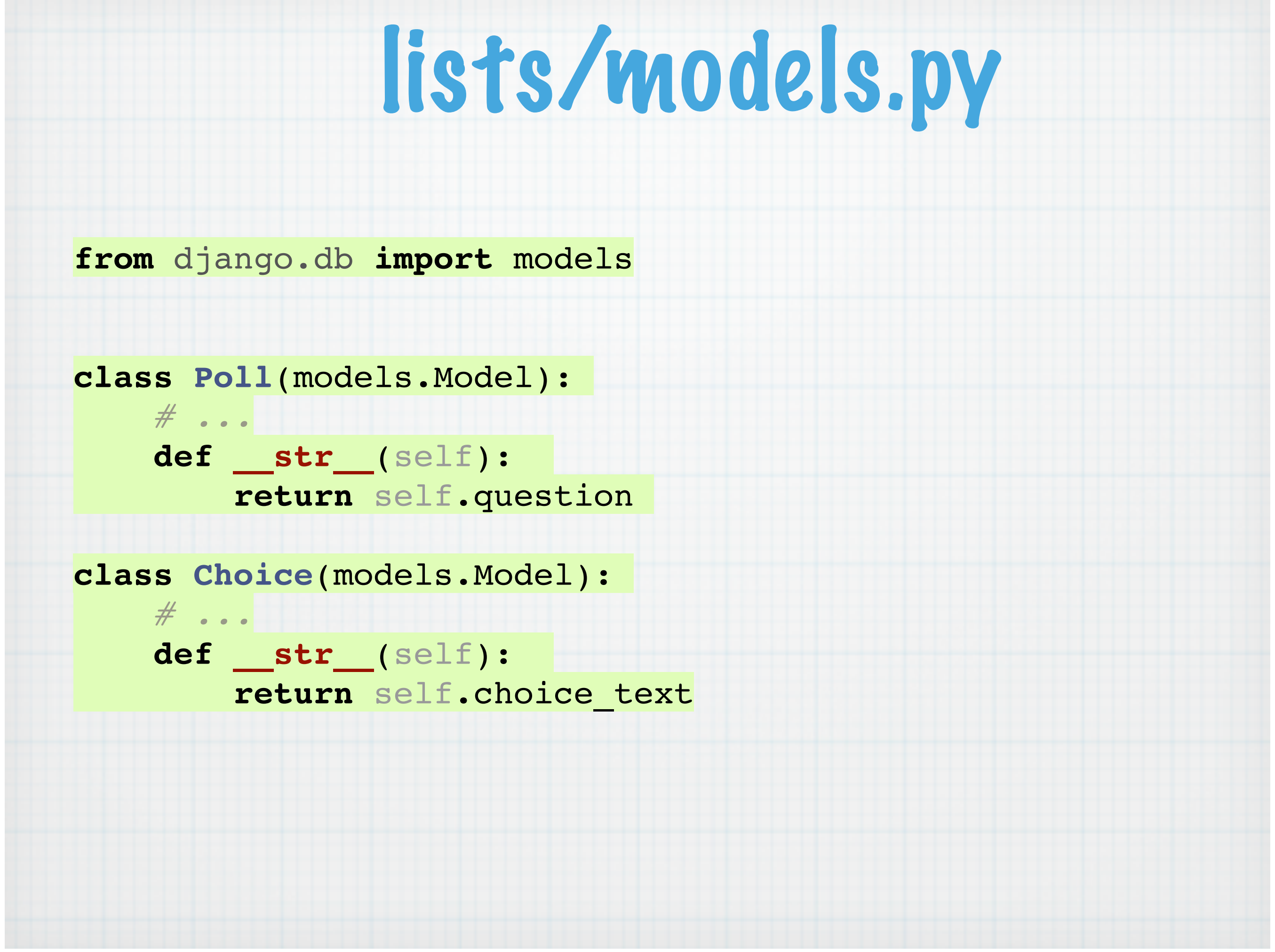


Create data in the shell

```
python3 manage.py shell
```

```
>>> from polls.models import Poll, Choice
>>> Poll.objects.all()
>>> from django.utils import timezone
>>> p = Poll(question="What's new?", pub_date=timezone.now())
>>> p.save()
>>> p.id
>>> p.question
>>> Poll.objects.all()
```

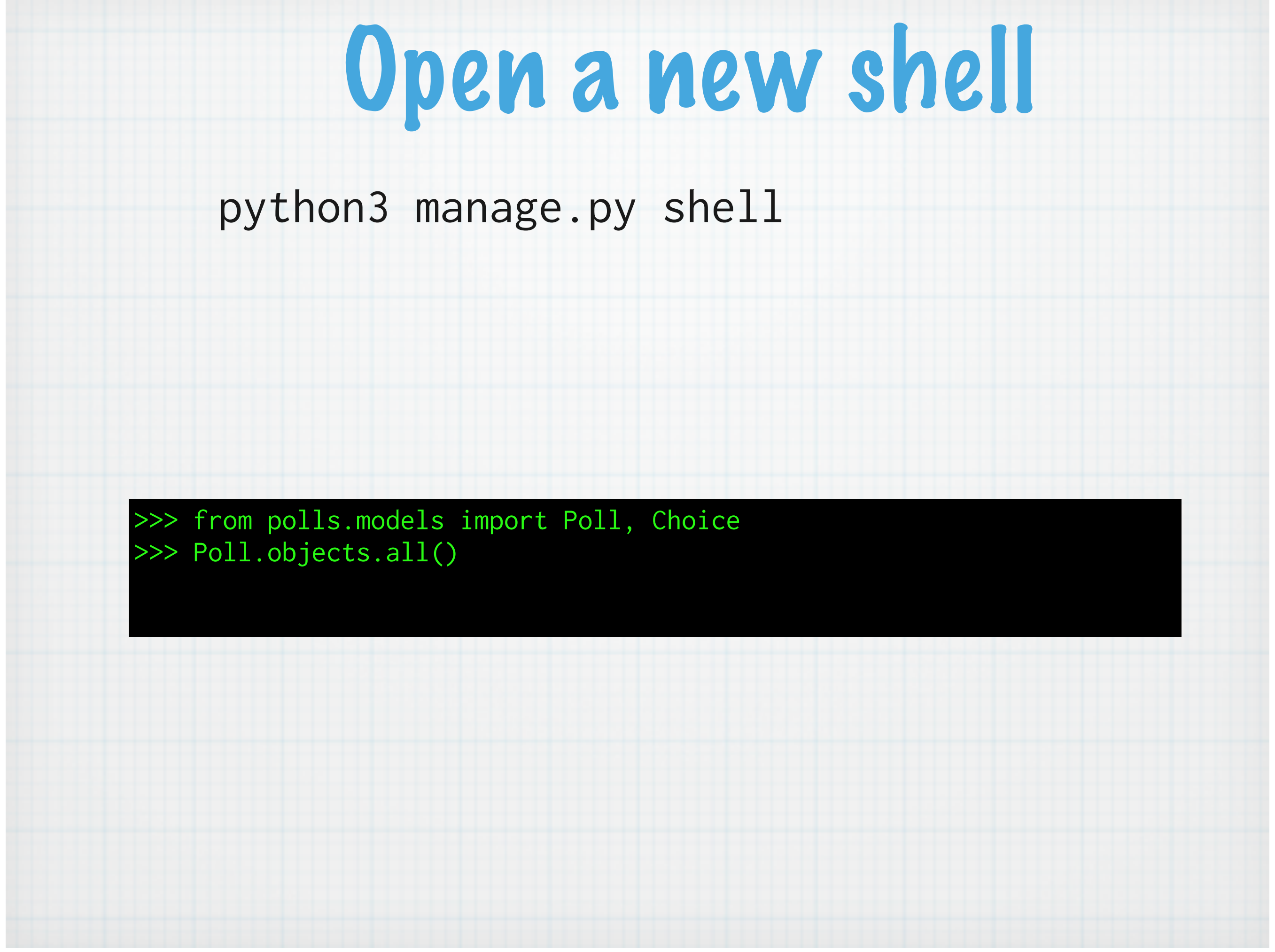
Python shell allows us to create poll objects and store them in the database



lists/models.py

```python
from django.db import models


class Poll(models.Model):
    # ...
    def __str__(self):
        return self.question

class Choice(models.Model):
    # ...
    def __str__(self):
        return self.choice_text
```

Add some more code to our model classes. This will allow objects to print something meaningful about themselves.



Open a new shell

```
python3 manage.py shell
```

```
>>> from polls.models import Poll, Choice
>>> Poll.objects.all()
```

You need to quit out of the python shell and open a new one to activate the updated model code.