Learn Django the Hard Way

by Daniel Greenfeld
assisted by Audrey Roy

Inspired by Zed Shaw

Daniel Greenfeld

- VP of development for cartwheelweb.com
- Co-Founder of LA Python
- Pinax Core Developer
- History: Worked for NASA, RevSys, Eldarion, and HoldenWeb
- Taught courses at PyCon 2010, 2011, NASA HQ, and co-authored the Python 3 classes for the O'Reilly School of Technology

Audrey Roy

- VP of design for cartwheelweb.com
- Co-Founder of LA Python
- History: Graduated from MIT, Worked for Microsoft, Sharpcast
- Consulted for various startups in Silicon Valley

What you need to have

- Basic familiarity with Python
- Laptop with permissions to install software
- Python 2.6 or higher already installed
- Willingness to type out the exercises
- Attention to detail
- Desire to learn

What we are building

- Basic familiarity with Django
- Good habits of
  - asking questions
  - following coding standards
  - always being as explicit as possible
- An application to schedule meet ups

How this works

1. You to build ‘developer’ muscle memory.
2. Type in exercises as you see them.
   Innovation comes later
3. You get lots of help to complete exercises.

Exercises

Each exercise has two stages:

1. A slide is displayed and the basic concepts are explained. **No questions yet!**
2. Everyone starts typing EXACTLY what is on the slide. Instructors walk around and help students. **Ask questions now!**

Exercise 0: Setup

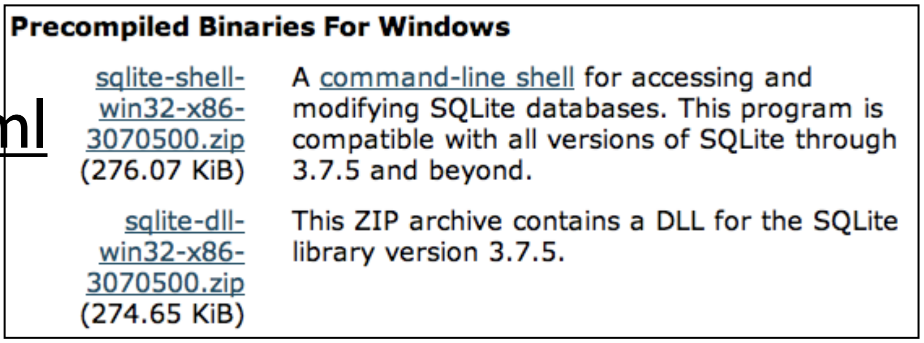
Sqlite3

Windows:

http://www.sqlite.org/download.html

Download these two files

Unzip in c:\Windows\system32



Precompiled Binaries For Windows

sqlite-shell-win32-x86-3070500.zip (276.07 KiB)	A command-line shell for accessing and modifying SQLite databases. This program is compatible with all versions of SQLite through 3.7.5 and beyond.
sqlite-dll-win32-x86-3070500.zip (274.65 KiB)	This ZIP archive contains a DLL for the SQLite library version 3.7.5.

Ubuntu:

sudo apt-get instal sqlite3

Mac OS X 10.6+:

Comes with the operating system!

-
  ```
  $ sqlite3
  SQLite version 3.6.12
  Enter ".help" for instructions
  Enter SQL statements terminated with a ";"
  sqlite> .quit
  ```

Exercise 0: Setup

Pip and Virtualenv

Commands

- curl -O http://python-distribute.org/distribute_setup.py
  Windows users: http://python-distribute.org/distribute_setup.py
- python distribute_setup.py
- curl -O https://github.com/pypa/pip/raw/master/contrib/get-pip.py
  Windows users: https://github.com/pypa/pip/raw/master/contrib/get-pip.py
- python get-pip.py
- pip install virtualenv
- virtualenv ldthw
- source ldthw/bin/activate
- (ldthw) $ python
  >>> exit()

Exercise 0: Setup

Files

https://github.com/cartwheelweb/ldthw

Exercise 1: Installing Django

Commands

- (Ldthw)$ pip install django==1.3
- (Ldthw)$ python

```
>>> import django
>>> django.get_version()
'1.3'
>>> exit()
```

Exercise 2: Creating a project

Commands

- (Ldthw)$ django-admin.py startproject eventme
- (Ldthw)$ cd eventme
- (Ldthw)$ ls -l

```
__init__.py
manage.py
settings.py
urls.py
```

Exercise 2: Creating a project

Commands

- (Ldthw)$ python manage.py runserver

```
Validating models...
0 errors found.

Django version 1.0, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Point your browser at http://127.0.0.1:8000

Exercise 3: database setup

Add to the bottom of settings.py:

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": "dev.db",
    }
}
```

Commands

- (Ldthw)$ python manage.py syncdb

```
Creating tables ...
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_user_permissions
Creating table auth_user_groups
Creating table auth_user
...
You just installed Django’s auth system, which means you don’t have any superusers
defined. Would you like to create one? (yes/no): yes
Username (leave blank to use ‘pydanny’): pydanny
...
```

Exercise 3: database setup

Now destroy and rebuild your work!

Commands

- (Ldthw)$ rm dev.db
- (Ldthw)$ python manage.py syncdb

```
Creating tables ...
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_user_permissions
Creating table auth_user_groups
Creating table auth_user
...
You just installed Django’s auth system, which means you don’t have any
superusers defined. Would you like to create one? (yes/no): yes
Username (leave blank to use ‘pydanny’): pydanny
...
```

Exercise 4: First Django App

Commands

- (Ldthw)$ django-admin.py startapp events
- (Ldthw)$ ls -l events/

```
-rw-r--r--  1 pydanny  staff    0 Apr  2 13:14 __init__.py
-rw-r--r--  1 pydanny  staff   57 Apr  2 13:14 models.py
-rw-r--r--  1 pydanny  staff  383 Apr  2 13:14 tests.py
-rw-r--r--  1 pydanny  staff   26 Apr  2 13:14 views.py
```

At bottom of settings.py:

```
...
INSTALLED_APPS += ('events',)
```

Pro Tip

Exercise 4: First Django App

"A Django app should focus on just one thing. If it takes more than 3 minutes to explain, then it should be broken up into two or more apps."

Exercise 5: First Django Model

In events/models.py:

```
from django.contrib.auth.models import User
from django.db import models

class Event(models.Model):

    title = models.CharField("Title", max_length=255)
    slug = models.SlugField("Slug")
    description = models.TextField("Description")
    start_datetime = models.DateTimeField("Start Date")
    attendees = models.ManyToManyField(User, blank=True,
                    null=True)
```

Exercise 5: First Django Model

Commands:

- (Ldthw)$ python manage.py validate

```
0 errors found
```

- (Ldthw)$ python manage.py syncdb

```
Creating tables ...
Creating table events_event
Installing custom SQL ...
Installing indexes ...
No fixtures found.
```

Pro Tip

Exercise 5: First Django Model

“The pixel shortage is over.”

“In this tutorial and in the open source works of respected Django developers you’ll see an absolute minimum of abbreviations. So in this class and in Cartwheel projects you’ll see:”

“**user** instead of usr”
“**attendees** instead of attnds”
“**function_that_does_things** instead of ftdt”

Exercise 6: Testing your work

In events/tests.py:

```
from datetime import datetime

from django.test import TestCase

from events.models import Event

class EventTest(TestCase):

    def test_create(self):

        event = Event(
            title = 'My First Event!',
            slug = 'my-first-event',
            description = 'huge amount of text',
            start_datetime = datetime.now(),
        )
        event.save()
        print Event.objects.all()
```

Exercise 6: Testing your work

- (Ldthw)$ python manage.py test events

```
Creating test database for alias 'default'...
[<Event: Event object>]
.
-----------------------------------------
Ran 1 test in 0.002s

OK
```

Extra credit

Exercise 6: Testing your work

In events/tests.py:

```
class EventTest(TestCase):

    def test_create(self):
        ...
        print event.title
        print event.slug
```

```
(Ldthw) $ python manage.py test events
Creating test database for alias 'default'...
[<Event: Event object>]
My First Event!
my-first-event
.
-----------------------------------------
Ran 1 test in 0.002s

OK
```

Pro Tip

Exercise 6: Testing your work

“The shell is really useful for figuring things out but when you exit the shell you’ve lost your work.”

“On the other hand, tests require a little more work up front, are on your filesystem, and you can use them as reference material later.”

“One more thing, tests are extremely useful in upgrading to the latest version of Django or Python.”

Exercise 7: Model unicode

Add to events/models.py:

```
class Event(models.Model):
    ...
    def __unicode__(self):
        return self.title
```

- (Ldthw)$ python manage.py test events

```
Creating test database for alias 'default'...
[<Event: My First Event!>]
.
-----------------------------------------
Ran 1 test in 0.002s

OK
```

Exercise 8: Test setUp

Add to events/tests.py:

```
class EventTest(TestCase):

    def setUp(self):

        event = Event(
            title = 'My First Event!',
            slug = 'my-first-event',
            description = 'huge amount of text',
            start_datetime = datetime.now(),
        )
        event.save()

        event = Event(
            title = 'My second Event!',
            slug = 'my-second-event',
            description = 'even more text!',
            start_datetime = datetime.now(),
        )
        event.save()
```

Exercise 8: Test setUp

- (Ldthw)$ python manage.py test events

```
Creating test database for alias 'default'...
[<Event: My Second Event!>, <Event: My Third Event!>,
<Event: My First Event!>]

.
-----------------------------------------
Ran 1 test in 0.002s

OK
```

Pro Tip

Exercise 8: Test setUp

“The unittest **setUp** method is great for laying out data you will want to check again and again.”

Exercise 9: ORM filters

Add to events/tests.py:

```
def test_filter(self):

    print 'All'
    print Event.objects.all()
    print 'by ID'
    print Event.objects.filter(id=1)
    print 'by title'
    print Event.objects.filter(title__startswith='My second')
    print 'by title case insensitive'
    print Event.objects.filter(title__istartswith='my second')
    print 'by year'
    print Event.objects.filter(start_datetime__year=2011)
    print 'date is before right now'
    print Event.objects.filter(start_datetime__lt=datetime.now())
    print 'date is after right now'
    print Event.objects.filter(start_datetime__gt=datetime.now())
```

Exercise 9: ORM filters

- (Ldthw)$ python manage.py test events

```
Creating test database for alias 'default'...
[<Event: My Second Event!>, <Event: My Third Event!>, <Event: My First Event!>]
All
[<Event: My Second Event!>, <Event: My Third Event!>]
by ID
[<Event: My Second Event!>]
by title
[<Event: My Second Event!>]
by title case insensitive
[<Event: My Second Event!>]
by year
[<Event: My Second Event!>, <Event: My Third Event!>]
date is before right now
[<Event: My Second Event!>, <Event: My Third Event!>]
date is after right now
[]

.
----------------------------------------
Ran 1 test in 0.002s

OK
```

Why ORM?

Development:

- Don’t have to write same queries 1000 times
- Lazy evaluation is fun
  - events = Event.objects.filter(id < 50)
  - events = events.exclude(title__istartswith=‘rails’)
  - print events
- Lots of handy methods:
  - all(), count(), filter(), exclude()

Security:

- Blocks SQL injection (mysql.com anyone?)

Exercise 10: Using the Admin

Append to settings.py:

```
...
INSTALLED_APPS += ('django.contrib.admin',)
```

Edit eventme/urls.py

```
from django.conf.urls.defaults import url, patterns, include
from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',
    url(
        regex = r'^admin/',
        view = include(admin.site.urls)
        ),
)
```

Exercise 10: Using the Admin

- (Ldthw)$ python manage.py syncdb

```
Creating tables ...
Creating table django_admin_log
Installing custom SQL ...
Installing indexes ...
No fixtures found.
```

- (Ldthw)$ python manage.py runserver

Point your browser at http://127.0.0.1:8000/admin

Pro Tip

Exercise 10: Using the Admin

Do it this way!
Don’t do * imports!

```
from django.conf.urls.defaults import url, patterns, include
from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',
    url(
        regex = r'^admin/',
        view = include(admin.site.urls)
        ),
)
```

Exercise 11: Events in the Admin

Create and edit eventme/events/admin.py:

```
from django.contrib import admin

from events.models import Event

admin.site.register(Event)
```

Restart the server:

- ctrl-c
- (Ldthw)$ python manage.py syncdb

Refresh http://127.0.0.1:8000/admin

Exercise 11: Events in the Admin

Now add 5 events!

Extra Credit

Exercise 11: Events in the Admin

Add 5 more events!

Exercise 12: Building Fixtures

Adding events got old, right?

From the root of eventme

- (Ldthw)$ mkdir events/fixtures
- (Ldthw)$ python manage.py dumpdata events --indent=4

  > events/fixtures/initial_data.json

One big line of text!

Exercise 12: Building Fixtures

Does your events/fixtures/events.json look something like this?:

```
[
  {
    "pk": 1,
    "model": "events.event",
    "fields": {
      "start_datetime": "2011-04-02 17:15:37",
      "description": "Zed Python Shaw Guitar San Francisco
Mongrel Rant",
      "slug": "lpthw",
      "title": "Learn Python the Hard Way",
    }
  },
  {
    "pk": 2,
    "model": "events.event",
    "fields": {
      "start_datetime": "2011-04-02 17:16:16",
      "description": "Danny Audrey Pycon Scared of Rabbits",

...
```

Exercise 12: Building Fixtures

Delete and sync your database

- (Ldthw)$ rm dev.db
- (Ldthw)$ python manage.py syncdb
- (Ldthw)$ python manage.py runserver

Check: http://127.0.0.1:8000/admin/events/event

Pro Tip

Exercise 12: Building Fixtures

"Good fixtures are a must. Working solo they save enormous chunks of time, working in a group they help people quickly understand your work without having to resort to getting a copy of production data."

"And you can use them in tests."

Exercise 13: Custom admin lists

Edit events.admin.py

```
from django.contrib import admin

from events.models import Event

class EventAdmin(admin.ModelAdmin):

    list_display = [‘title’,‘start_datetime’,‘slug’,]
    search_fields = [‘title’,]

admin.site.register(Event, EventAdmin)
```

Check: http://127.0.0.1:8000/admin/events/event

Exercise 14: Home page

New directory:

```
eventme/templates
```

New file:

```
eventme/templates/home.html
```

Exercise 14: Home page

Contents of templates/home.html

```
<h1>Eventme Home</h1>

<p>The home page for Eventme!</p>

{{ object_list }}
```

Append to settings.py:

```
...
import os.path
PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))
TEMPLATE_DIRS += (os.path.join(PROJECT_ROOT, "templates"),)
```

Exercise 14: Home page

Append to eventme/urls.py

```
...
from django.views.generic import ListView, DetailView

from events.models import Event

urlpatterns += patterns('',

    url(
        regex = r'^$',
        view = ListView.as_view(
            template_name="home.html",
            queryset=Event.objects.all()),
        name = "home"
    ),
)
```

Exercise 14: Home page

Is your server running?

- (Ldthw)$ python manage.py runserver

Check: http://127.0.0.1:8000

Pro Tip

Exercise 14: Home page

"You can lessen the amount of typing required to describe URL routes by removing keyword arguments. Which makes your code more terse."

"The downside to terse code is that with complex URL schemes terse code makes things harder to maintain."

Exercise 15: Testing attendees

Append to events/tests.py

Fixtures

Remove print

Assertion

```
from django.contrib.auth.models import User

class AttendeeTest(TestCase):

    fixtures = ['events.json',]

    def setUp(self):

        self.user = User.objects.create_user('joe','pw','joe@cw.com')

    def test_attendee_list(self):
        event = Event.objects.all()[0]
        print event.attendees.all()
        self.assertTrue(event.attendees.all())
```

Exercise 16: Home page plus

Change eventme/templates/home.html:

```
{% block content %}
<h1>Eventme Home</h1>

<p>The home page for Eventme!</p>

<ul>
    {% for event in object_list %}
        <li>{{ event }}
            {% if user not in event.attendees.all %}
                [ <a href="TODO">ATTEND</a> ]
            {% endif %}
            <p>
                <strong>{{ event.start_datetime }}</strong>
                {{ event.description }}
            </p>
        </li>
    {% endfor %}
</ul>
{% endblock %}
```

Pro Tip

Exercise 16: Home page plus

“Put each new HTML and Django Template Language element on their own line and with appropriate whitespace. It will save you a lot of grief in debugging your work later and will prevent others from accusing you of delivering spaghetti.”

“This also works smashingly well in other languages and tools.”

Exercise 17: Creating a base page

Add eventme/templates/base.html

```
<html>

<title>Eventme</title>

<link href="http://cartwheelweb.com/media/css/blueprint/screen.css"
        rel="stylesheet" type="text/css"  />
<body>

{% block content %}{% endblock %}
</body>


</html>
```

Prepend eventme/templates/home.html

```
{% extends “base.html” %}
```

Exercise 18: Testing attendees Continued

Append to events/tests.py

```
....
def test_attendee_add(self):

    event = Event.objects.all()[0]
    event.attendees.add(self.user)

    self.assertTrue(event.attendees.all())
```

Don't forget whitespace!

Pro Tip

Exercise 18: Testing attendees

“Assertions are what test harnesses are all about. Well-tested systems tend to be easier to maintain and upgrade.”

“The downside to tests is writing them takes time - which is why it can be handy to use them instead of the shell.”

Exercise 19: Action views

Change eventme/events/views.py

```
from django.contrib.auth.decorators import login_required
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404

from events.models import Event

@login_required
def attend_event(request, slug):

    event = get_object_or_404(Event, slug=slug)
    event.attendees.add(request.user)
    return HttpResponseRedirect(reverse('home'))
```

Exercise 19: Action views

Add the view function into eventme/urls.py

```
from events.views import attend_event

urlpatterns += patterns('',

    url(
        regex = r'^attend-event/(?P<slug>[\w-]+)$',
        view = attend_event,
        name = "attend_event"
    ),
)
```

Exercise 20: Action views

Change TODO at eventme/templates/home.html

```
{% extends "base.html" %}
{% block content %}
<h1>Eventme Home</h1>

<p>The home page for Eventme!</p>

<ul>
    {% for event in object_list %}
        <li>{{ event }}
            {% if user not in event.attendees.all %}
                [ <a href="{% url attend_event event.slug %}">ATTEND</a> ]
            {% endif %}
            <p>
                <strong>{{ event.start_datetime }}</strong>
                {{ event.description }}
            </p>
        </li>
    {% endfor %}
</ul>
{% endblock %}
```

Test at http://127.0.0.1:8000

Extra Credit

Exercise 20: Action views

Allow unattendance at events you've marked as attending.

```
{% if user in event.attendees.all %}
        [ <a href="{% url unattend_event event.slug %}">UNATTEND</a> ]
{% endif %}
```

```
@login_required
def attend_event(request, slug):

    event = get_object_or_404(Event, slug=slug)
    event.attendees.remove(request.user)
    return HttpResponseRedirect(reverse('home'))
```

```
    url(
        regex = r'^unattend-event/(?P<slug>[\w-]+)/$',
        view = unattend_event,
        name = "unattend_event"
    ),
```

Pro Tip

Exercise 20: Action views

```
from events.views import attend_event

urlpatterns += patterns('',

    url(
        regex = r'^attend-event/(?P<slug>\d+)$',
        view = attend_event,
        name = "attend_event"
    ),
)
```

Declare your views explicitly!

Leave this empty so debugging is easier!

The view and name arguments should have the same name in order to make code searches easier!

Exercise 21: Why test?

- (Ldthw)$ python manage.py test events

Add this to eventme/events/models.py:

```
class Event(models.Model):
    ...
    @property
    def title(self):
        return "django"
```

- (Ldthw)$ python manage.py test events

EXPLOSION OF ERRORS!

Exercise 21: Why test?

Remove from

eventme/events/models.py:

```
class Event(models.Model):
    ...
    @property
    def title(self):
        return "django"
```

- (Ldthw)$ python manage.py test events

Pro Tip

Exercise 21: Why test?

“During the **PyCon 2011** Packaginator sprint, someone attached a property decorator to a function. Without tests, this show stopping bug got past 28 people and into Django Packages production before it was discovered.”

“Finding the bug and implementing a fix took hours.”

Exercise 22: Detail Views

In eventme/templates/home.html change:

```
{% for event in object_list %}
    <li>{{ event }}
       {% if user not in event.attendees.all %}
```

to:

```
{% for event in object_list %}
    <li><a href="{% url event_detail event.slug %}">{{ event }}</a>
       {% if user not in event.attendees.all %}
```

Test at http://127.0.0.1:8000

Exercise 22: Detail Views

Add to
eventme/urls.py
assumes default HTML file of:
“templates/events/event_detail.html”

```
    url(
        regex = r'^event/(?P<slug>[\w-]+)/$',
        view = DetailView.as_view(
            queryset=Event.objects.all()
            ),
        name = "event_detail"
    ),
```

Create
templates/events/event_detail.html

```
{% extends "base.html" %}
{% block content %}
<h1>Event {{ object.title }}</h1>

<p>{{ object.description }}</p>

<h2>Attendees</h2>
<ul>
    {% for attendee in object.attendees.all %}
        <li>{{ attendee }}</li>
    {% endfor %}
</ul>
{% endblock %}
```

Pro Tip

Exercise 22: Detail Views

“Class Based Views (CBVs) which we’ve used to display all our HTML via generic views are incredible time savers.”

“However, CBVs outside of generic views is a very advanced topic and according to at least one BDFL, you should use regular view functions for non-generic views until better documentation is available.”

Exercise 23: Specific Views

Append to eventme/events/views.py:

```
from django.conf import settings
from django.shortcuts import render_to_response
from django.template import RequestContext

def event_detail_plus(request, slug,
                template_name="events/event_detail_plus.html"):

    event = get_object_or_404(Event, slug=slug)
    attendees = event.attendees.all()

    return render_to_response(
        template_name,
            {
            'event': event,
            'attendees': attendees,
            'ROOT_URLCONF': settings.ROOT_URLCONF
            },
         context_instance = RequestContext(request)
    )
```

Exercise 23: Specific Views

Add eventme/templates/events/event_detail_plus.html:

```
{% extends "base.html" %}
{% block content %}
<h1>Event {{ event.title }}</h1>

<p>{{ event.description }}</p>

<p>{{ ROOT_URLCONF }}</p>

<h2>Attendees</h2>
<ul>
    {% for attendee in attendees %}
        <li>{{ attendee }}</li>
    {% endfor %}
</ul>
{% endblock %}
```

Exercise 23: Specific Views

Append to eventme/urls.py:

```
from events.views import event_detail_plus
urlpatterns += patterns('',
    url(
        regex = r'^event_plus/(?P<slug>[\w-]+)/$',
        view = event_detail_plus,
        name = "event_detail_plus"
    ),
)
```

In eventme/templates/home.html change:

```
{% for event in object_list %}
    <li><a href="{% url event_detail_plus event.slug %}">{{ event }}</a>
        {% if user not in event.attendees.all %}
```

Pro Tip

Exercise 23: Specific Views

“Specific views can grow out of control really quickly.”

“Any time you write the same code more than once, don’t forget to wrap it up in a function!”

Exercise 24: Remove prints

In events/tests.py,

replace ‘**print**’ statements with assertions.

```
(Ldthw)$ python manage.py test events
Creating test database for alias 'default'...
...
---------------------------------------
Ran 3 tests in 0.024s

OK
```

Exercise 25: Zen of Python

```
(Ldthw)$ python -c 'import this'
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Finding good code

The next step is find a project known for good code examples and follow their patterns forever. We recommend the following projects:

http://djangopackages.com/packages/p/packaginator/

http://djangopackages.com/packages/p/pinax/

http://djangopackages.com/packages/p/pycon/

http://djangopackages.com/packages/p/django-taggit/

Other resources

Django Tutorial: http://docs.djangoproject.com/en/dev/intro/tutorial01/