python

# and
# 技術分享

100062273 資工三 吳恩新

# Outline

- Python
  - Why Python?
  - Install
  - hello world
  - Property
  - Type
  - Control Flow
  - Function
  - File I/O
  - Object and Class
  - Module
  - Useful built-in function
  - Reference

# Outline

- Django
    - Why Django?
    - Framework and Library
    - MVC Framework
    - Install
    - hello world (view only)
    - hello world (plus template)
    - Static files
    - Model
    - Admin
    - Form
    - User Authentication
    - Reference

## Python

# Why Python?

- PEP 20

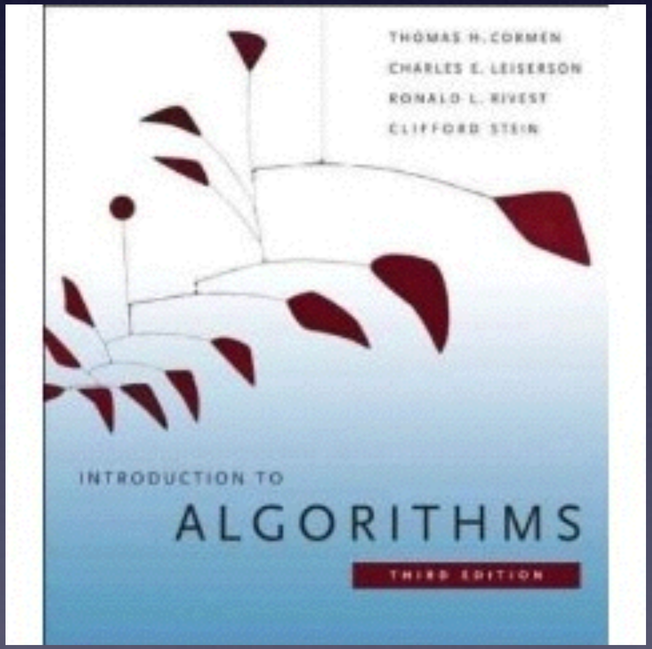
```
>>> import this
The Zen of Python, by Tim Peters

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!
```

# Why Python?

- 好寫好讀又好學
- 不用加分號「;」（想想看你被它陰過幾次）
- 利用縮行取代括號
- 好用的內建型別與內建函式

“I was translating pseudocode into Python.

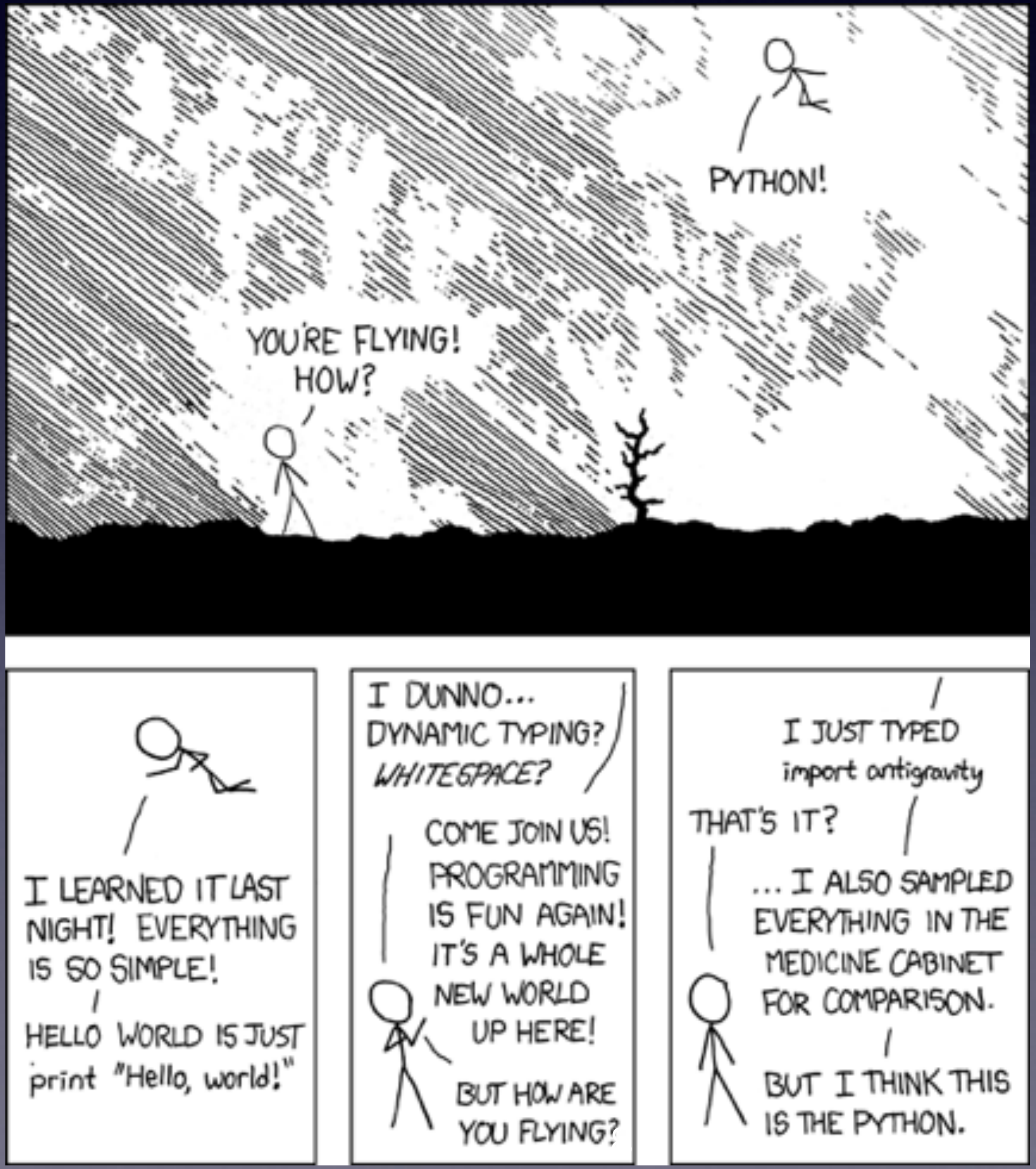
It got smaller and more readable.”



**– Allen B. Downey (author of Think Python)**

# Why Python?

```
$ python -c "import antigravity"
```


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

## Only Python 2 today…

and something important related to Django

## Install

# Install

- Linux (Debian/Ubuntu based)
  - $ sudo apt-get install python
- Mac
  - $ brew install python
  - http://brew.sh
- Windows
  - https://www.python.org/downloads/
  - 加入Python至環境變數PATH
  - (google it if you got problem, because I don’t know either T__T)

## Install

- How to check?

- type “python” in terminal / command line

```
$ python
Python 2.7.6 (default, Dec  4 2013, 01:07:30) 
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on 
darwin
Type "help", "copyright", "credits" or "license" for more 
information.
>>> 
```

- Interactive mode (very useful!!)

## Editor / IDE

- Pick whatever you like
  - VIM, Emacs, Sublime, Notepad++…etc
- https://wiki.python.org/moin/PythonEditors

# IPython

- 神兵利器
- `$ pip install ipython`
- Demo

# hello world!

1. in Interactive mode

```
In [1]: print 'hello world!'
hello world!
```

2. in py file (hello.py)

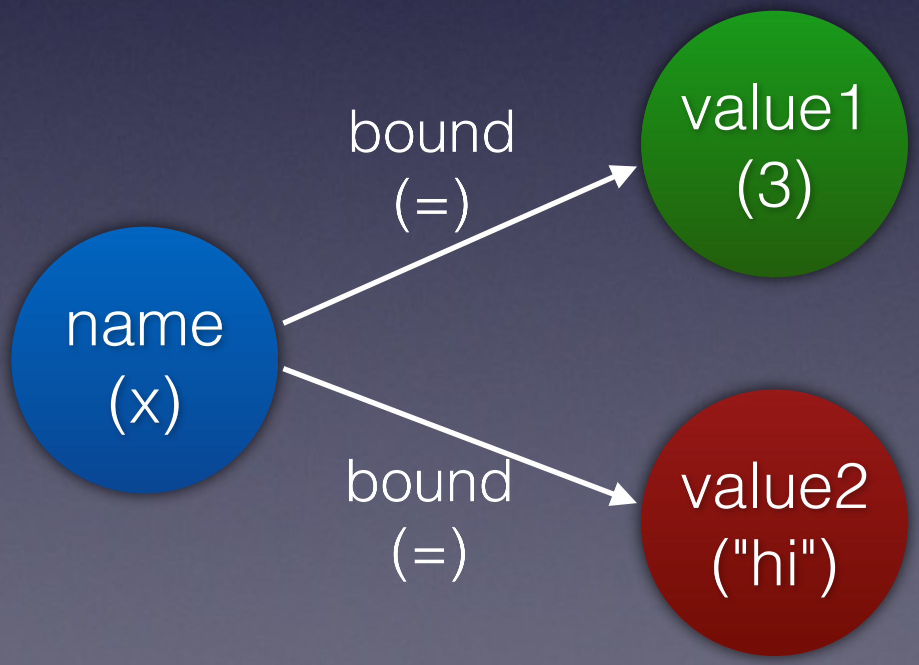
```
#!/usr/bin/env python

print 'hello world!'
```

```
 $ python hello.py
hello world!
```

# Property

- Dynamic Typing
  - checking types in run time
  - 變數不需要事先宣告
  - a name bound to a value

```
>>> x = 3
>>> x = "hi"
```



## Property

- Strong Typing
    - 不同型別無法相互運算

```
>>>'hello' + 3   # error

>>>'hello' + str(3)   # work
'hello3'
```

## Property

- mutable
  - value can be modified after created.
  - dict, list
- immutable
  - value can’t be modified after created.
  - number, str, tuple

# Type

- None
- Numbers: int, long, float, complex, bool
- Sequences: str, unicode, list, tuple, xrange
- Mapping: dict
- Sets: set, frozenset

## Type

- Numbers: int, , float, , bool
- Sequences: str, , list, tuple,
- Mapping: dict
- Sets: set,

## Numbers

## Integer

- 123, 223, 1238012380123
- 不用寫大數了
- 運算元: + - * / % **
- 2 ** 10 = 1024

# Float

- 1.2, 3.4, 5.523
- Still have to pay attention
  - 5 / 2 = 2
  - 5 / 2. = 2.5

## Boolean

- True, False (大寫)
- a in "abc" # True
- 3 in [1, 2, 3] # True
- [], {}, "", 0, None, 0.0  # False

## Sequences

## Common Operations

- s[i]: 第i個元素 (從0開始算，左邊開始)

```
>>> "123"[0] # 1
```

- s[-i]: 第i個元素 (從1開始算，右邊開始)

```
>>> "123"[-1] # 3
```

- len(s): 計算s的長度

- s[i:j]: 第i ~ j-1個元素
  - s = s[:] = s[0:len(s)]

## str (string)

- immutable

```
a = "123"
a[0] = "2" # error
```

- "I'm string"
- 'string2'
- """string""", '''string'''
  - just like <pre>string</pre> in html

# List

- mutable
- a = [1, 2, 3, 4]
    - max(a) # 4
    - min(a) # 1
    - sum(a) # 10
    - a.append(5) # [1, 2, 3, 4, 5]
- b = ['hi', "hello", 1, 2, 3.3, a]
- a = a + b
- You can put any object in list

# List

- mutable

```
In [1]: a = [1, 2, 3, 4]

In [2]: b = a

In [3]: b[2] = 5

In [4]: b
Out[4]: [1, 2, 5, 4]

In [5]: a
Out[5]: [1, 2, 5, 4]
```

## Tuple

- immutable
- a = (1, 2, 3)
  - a[0] = 2  # error
- b = ('hi', 3.3, "uccu")

## Mapping

## dict (dictionary)

- mutable
- {key: value} (implement by hash)
- d = {1:2, 'a': 'b'}
- d[1] # 2
- d['a'] # ‘b'
- d[3.3] = 4

## Sets

## set

```
In [1]: s = set()

In [2]: s.add(1)

In [3]: s.add(1)

In [4]: s.add(1)

In [5]: s.add(3)

In [6]: s
Out[6]: {1, 3}
```

## Control Flow

# if else

```
if 3 > 2:
    print 'yes'
else:
    pass # do nothing
```

- 要縮排且一致，否則error

```
if 3 > 2:
    print 'yes'
        print 'no' # IndentationError
```

## if else

```python
if x == 1:
    pass
elif x == 2:
    pass
elif x == 3:
    pass
else:
    pass
```

- there is no switch in Python

## while

```
a = []
i = 0
while i < 5:
    a.append(i*i)
    i += 1

print a # [0, 1, 4, 9, 16]
```

- 有continue, break，可搭配著用

## for in

```
a = []
for i in range(5):
    a.append(i*i)

print a # [0, 1, 4, 9, 16]

# range(5) = [0, 1, 2, 3, 4]
```

# list comprehension

```
a = [i*i for i in range(5)]
print a #[0, 1, 4, 9, 16]
```

## Functions

## example

```
def swap(a, b):
    return (b, a)

a = 30
b = 200
a, b = swap(a, b)
print a, b # 200 30
# a, b = b, a
```

## Function is First-Class Object

```
def square(x):
    return x ** 2

a = square
print a(3) # 9
```

## parameter with a default value

```
def power(a, b=2):
    return a ** b

print power(3) # 9
print power(3, 3) # 27
```

# lambda

- lambda args(input): expression(output)

```
mul = lambda x, y: x*y
x = mul(3, 5)
print x # 15
```

## File I/O

## read file

```
for line in open('filename.txt'):
    print line,
```

## write file

```
f = open('filename.txt', 'w')
print >>f, "content you want to write"
f.close()
```

## object and class

# example

```python
class Account(object):
    def __init__(self, name, balance):
        self.name = name
        self.balance = balance

    def deposit(self, amt):
        self.balance = self.balance + amt

    def withdraw(self, amt):
        self.balance = self.balance - amt

    def inquiry(self):
        return self.balance

a = Account('Peter', 123)
b = Account('David', 456)
```

```java
public class Account {
    private String name;
    private int balance;

    public Account(String name, int balance) {
        this.name = name;
        this.balance = balance;
    }

    public void deposit(int amt) {
        this.balance = this.balance + amt;
    }

    public void withdraw(int amt) {
        this.balance = this.balance - amt;
    }

    public int query() {
        return this.balance;
    }

    public static void main (String[] args) {
        Account a = new Account("Peter", 123);
        Account b = new Account("David", 456);
    }
}
```

## Module

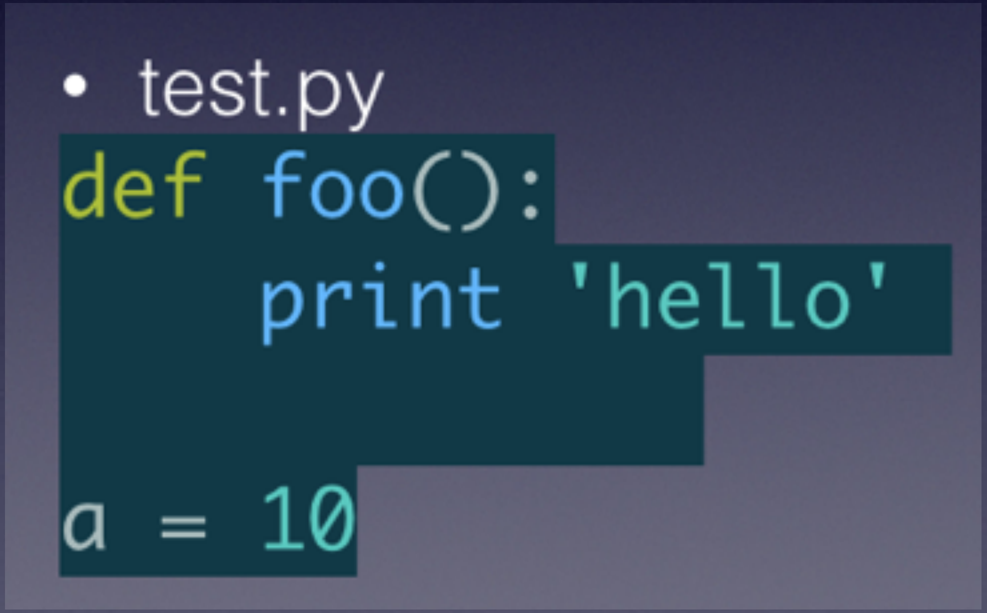
# import

- every python file is a module
- import it, then you can use its variable and function
- test.py

```
def foo():
    print 'hello'

a = 10
```

# import

```
In [1]: import test

In [2]: test.a
Out[2]: 10

In [3]: test.foo()
hello
```

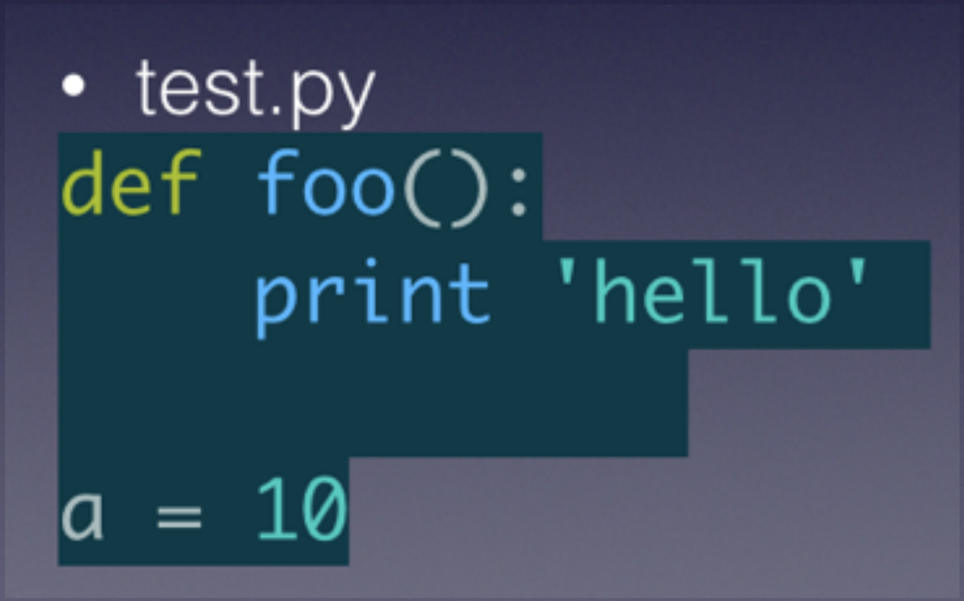


- test.py

```
def foo():
    print 'hello'

a = 10
```

## from ... import ...

```
In [1]: from test import a, foo

In [2]: a
Out[2]: 10

In [3]: foo()
hello
```



- test.py

```
def foo():
    print 'hello'

a = 10
```

## from … import …

- from module import *
  - **DON'T DO THIS!**

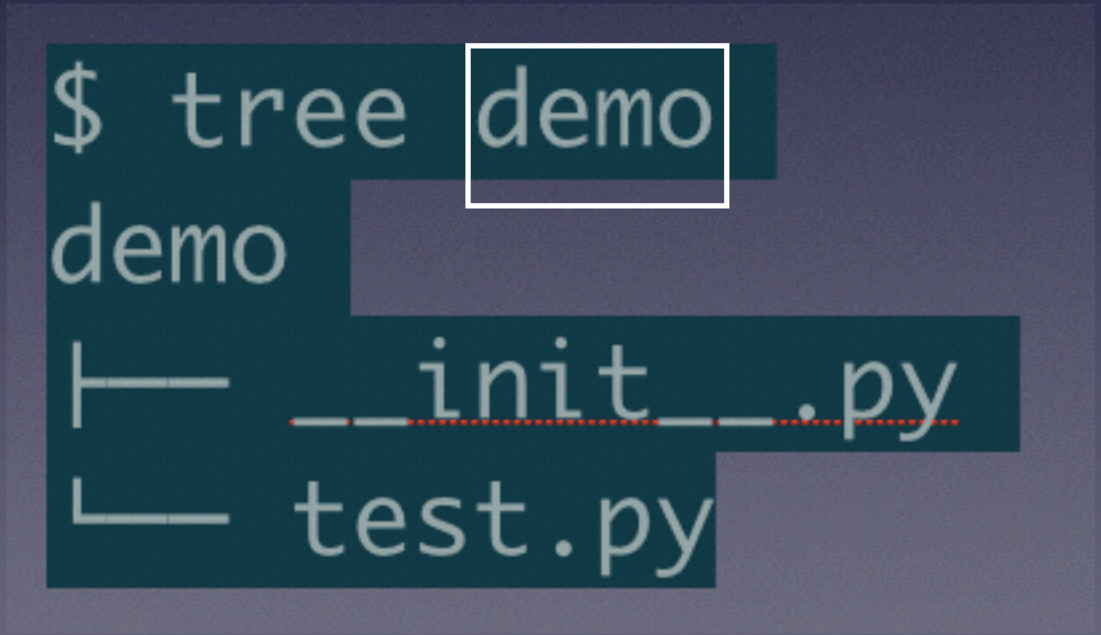
## Package

- similar conception to java package
- make a directory and make a empty file __init__.py

```
$ tree demo
demo
├── __init__.py
└── test.py
```

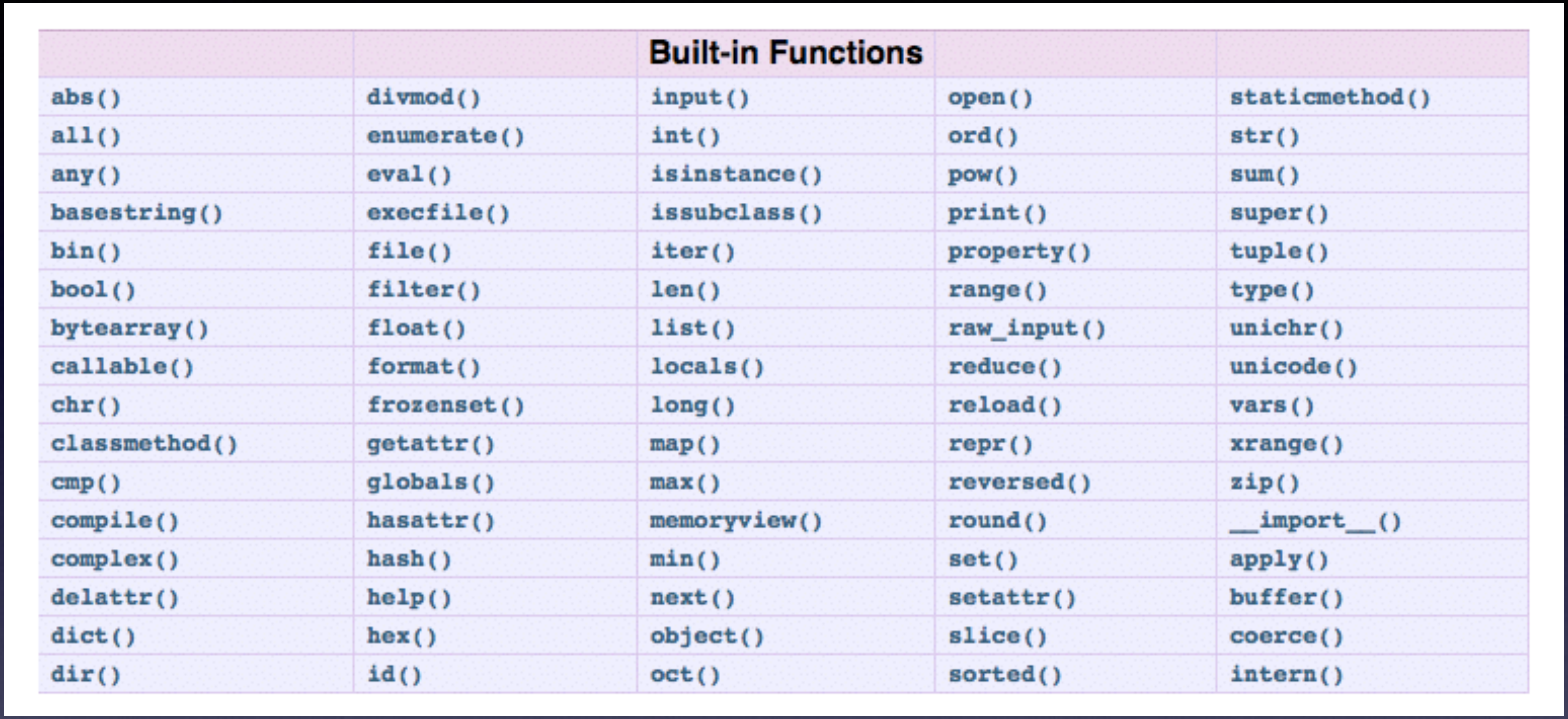
# Package

```
In [1]: from demo.test import a, foo

In [2]: foo()
hello

In [3]: a
Out[3]: 10
```



```
$ tree demo
demo
├── __init__.py
└── test.py
```

## Built-in Functions



| Built-in Functions | | | | |
|---|---|---|---|---|
| abs() | divmod() | input() | open() | staticmethod() |
| all() | enumerate() | int() | ord() | str() |
| any() | eval() | isinstance() | pow() | sum() |
| basestring() | execfile() | issubclass() | print() | super() |
| bin() | file() | iter() | property() | tuple() |
| bool() | filter() | len() | range() | type() |
| bytearray() | float() | list() | raw_input() | unichr() |
| callable() | format() | locals() | reduce() | unicode() |
| chr() | frozenset() | long() | reload() | vars() |
| classmethod() | getattr() | map() | repr() | xrange() |
| cmp() | globals() | max() | reversed() | zip() |
| compile() | hasattr() | memoryview() | round() | __import__() |
| complex() | hash() | min() | set() | apply() |
| delattr() | help() | next() | setattr() | buffer() |
| dict() | hex() | object() | slice() | coerce() |
| dir() | id() | oct() | sorted() | intern() |

# don't reinvent the wheel!!

https://docs.python.org/2.7/library/functions.html

## Reference

# If you want to learn more…

1. Dive Into Python (free)

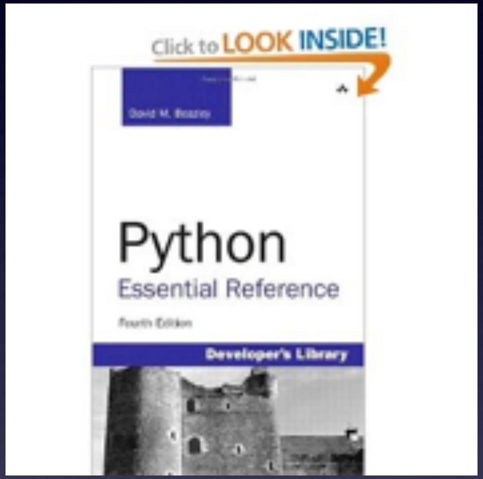
    - http://www.diveintopython.net

2. Think Python (free)

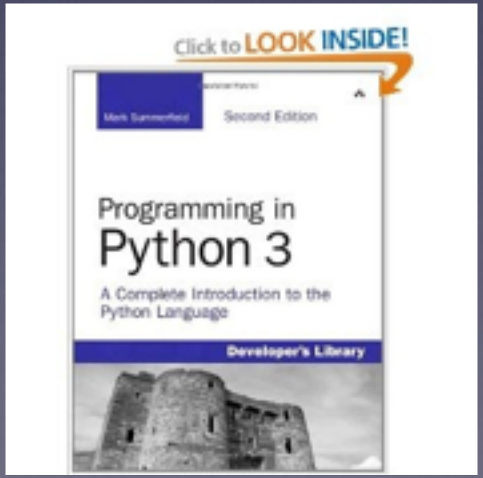
    - http://www.greenteapress.com/thinkpython/thinkpython.html

# If you want to learn more…

3. Python Essential Reference



4. Programming in Python 3



## Q&A

django

## Django

a high-level Python Web Framework

## Why Django?

# WHY?

- 因為可以寫Python！
- 我的第一個框架，其實也無從比較

# 官方說法

- Overall
  - Loose coupling
  - Less code
  - Quick development
  - Don’t repeat yourself (DRY)
  - Explicit is better than implicit
  - Consistency
- …
- https://docs.djangoproject.com/en/1.5/misc/design-philosophies/

## Popular sites use Django

- Disqus
- Pinterest
- Instagram
- Mozilla
- many newspaper websites (The Washington Post, UK’s Guardian, The New York Times)

## What is framework?

「框架就是制定一套規範或者規則，大家（程式設計師）在該規範或者規則下工作。」

**–中文維基百科**

「就是使用別人搭好的舞台，你來做表演！」

**–中文維基百科**

## Framework and Library

## difference?

- Framework
  - “Don’t call us, we’ll call you.”
  - inversion of control
  - constrain by framework
- Library
  - you control everything
  - you decide everything
  - freedom

## 慎選好的框架很重要

## MVC Framework

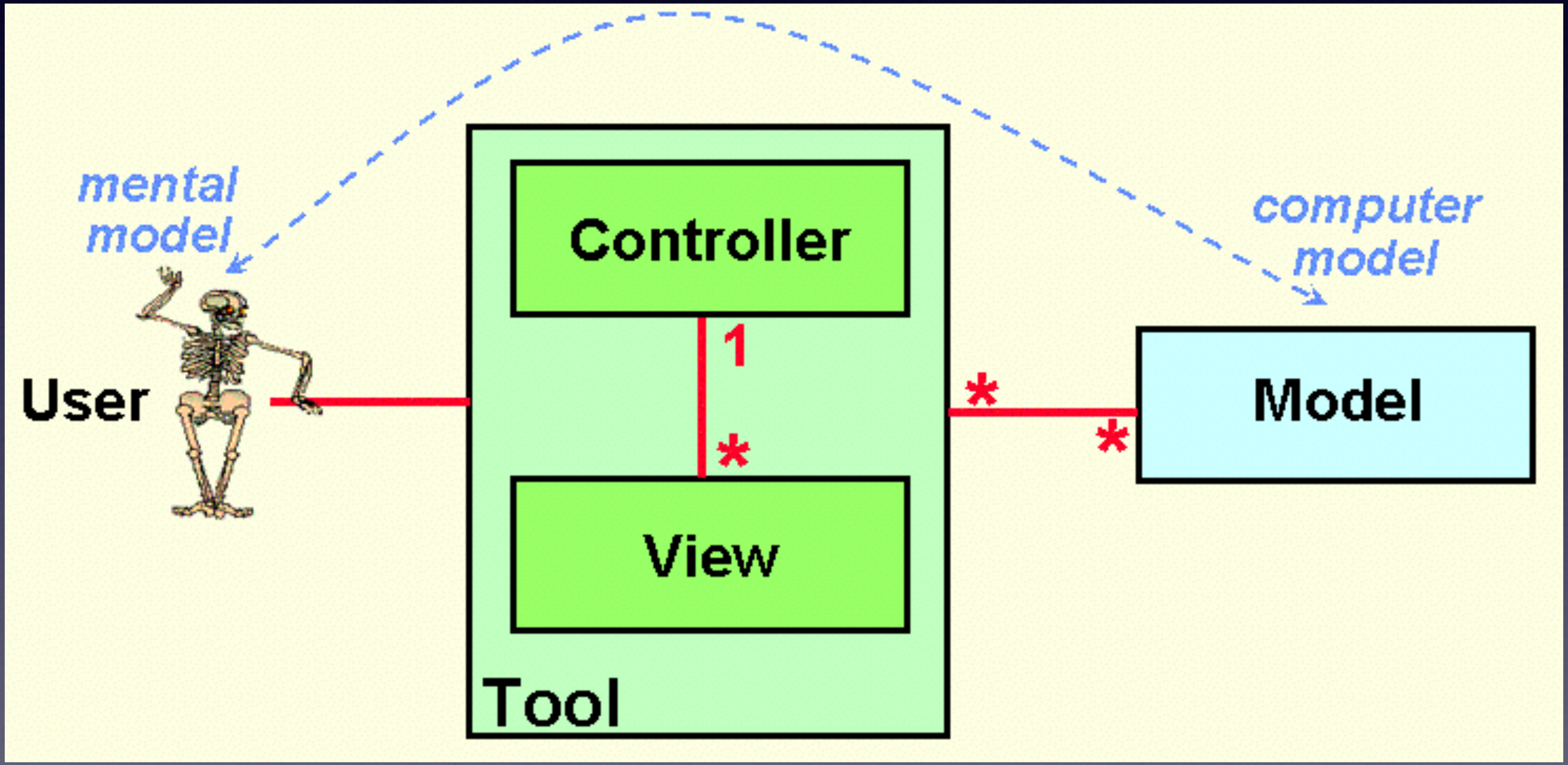
## Model-View-Controller

- MVC was originally invented by Smalltalk programmers.
- More specifically, it was invented by one Smalltalk programmer, Trygve Reenskaug.
- http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

"MVC was conceived as a general solution to the problem of users controlling a large and complex data set"

**–Trygve Reenskaug**

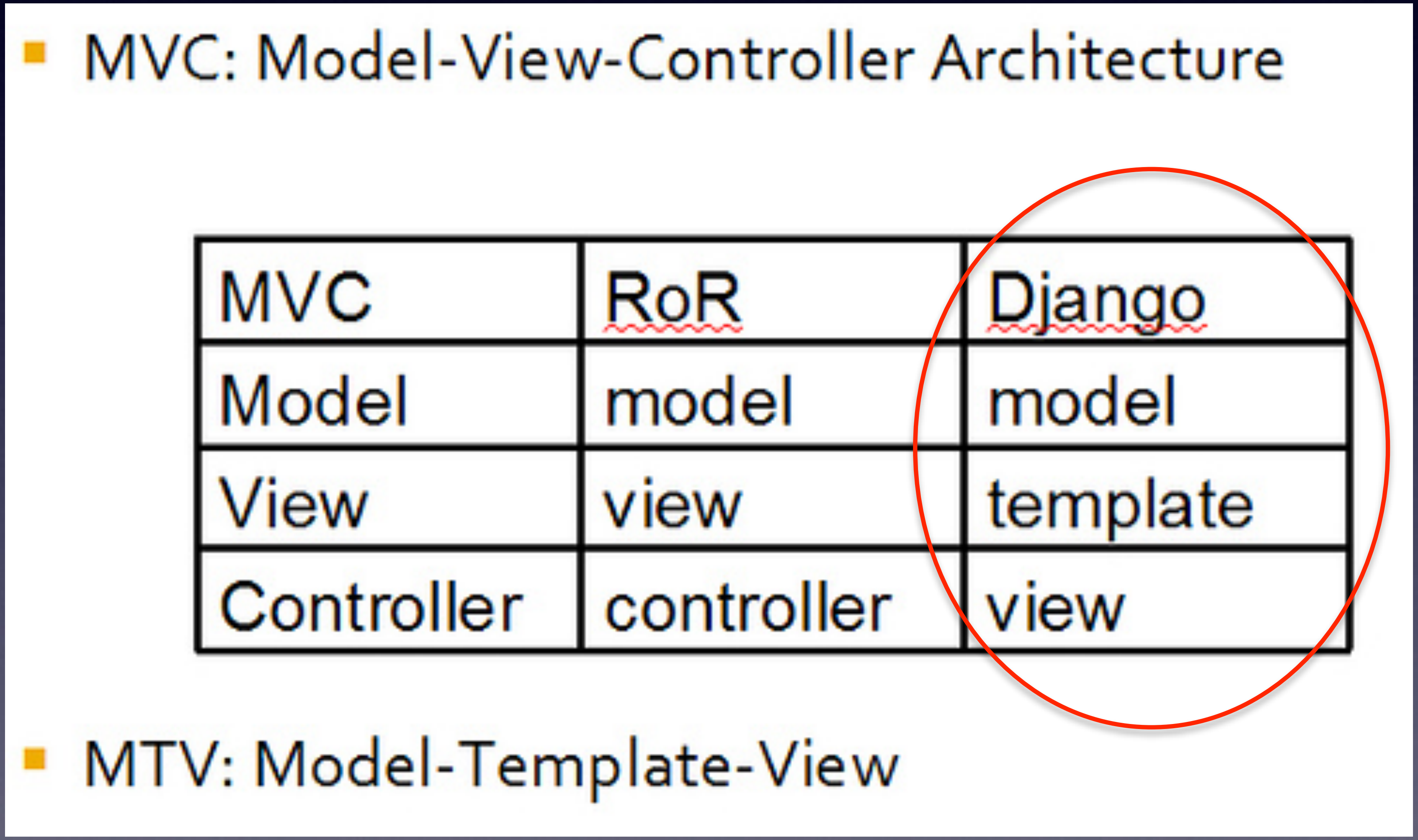
# MVC


mental model
computer model
Controller
1
*
View
Tool
User
*
*
Model

# 簡單的說

- Model
  - 定義資料（同步）
- View
  - 呈現資料
- Controller
  - 操作資料的呈現方式

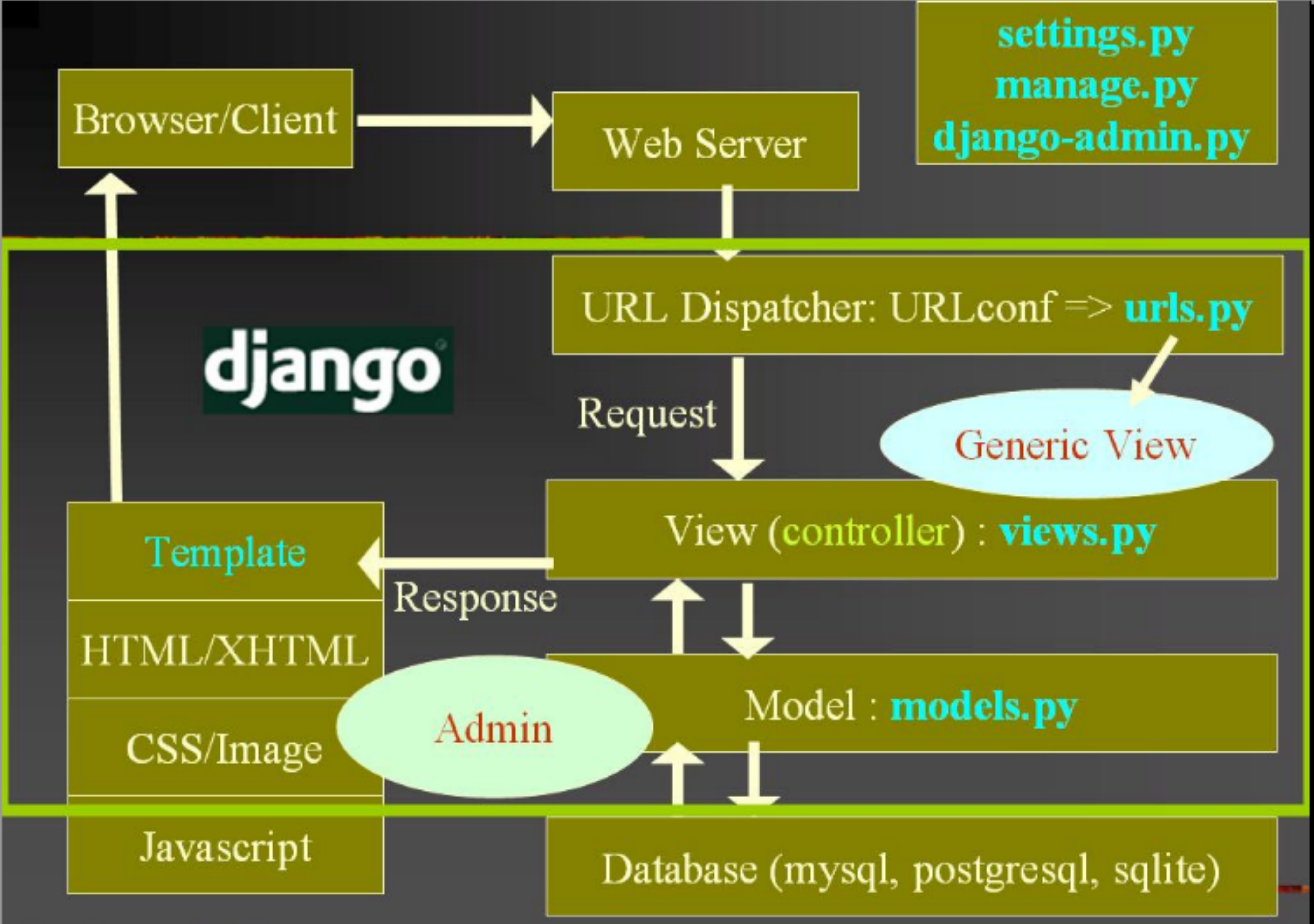
## MTV Framework



- MVC: Model-View-Controller Architecture

| MVC | RoR | Django |
|---|---|---|
| Model | model | model |
| View | view | template |
| Controller | controller | view |

- MTV: Model-Template-View



timchen119.blogspot.com

## Install

## Install Django

- $ pip install django==1.5.4
- Latest release: 1.6.4
- $ python -c "import django; print(django.get_version())”
  1.5.4

# Install database

- MySQL
  - $ brew install mysql  (Mac)
  - $ sudo apt-get install  mysql-server (debian/ubuntu)
- mysql-python
  - $ pip install mysql-python

or just use **sqlite3**
Choose whatever you like…

## hello world
## (view only)

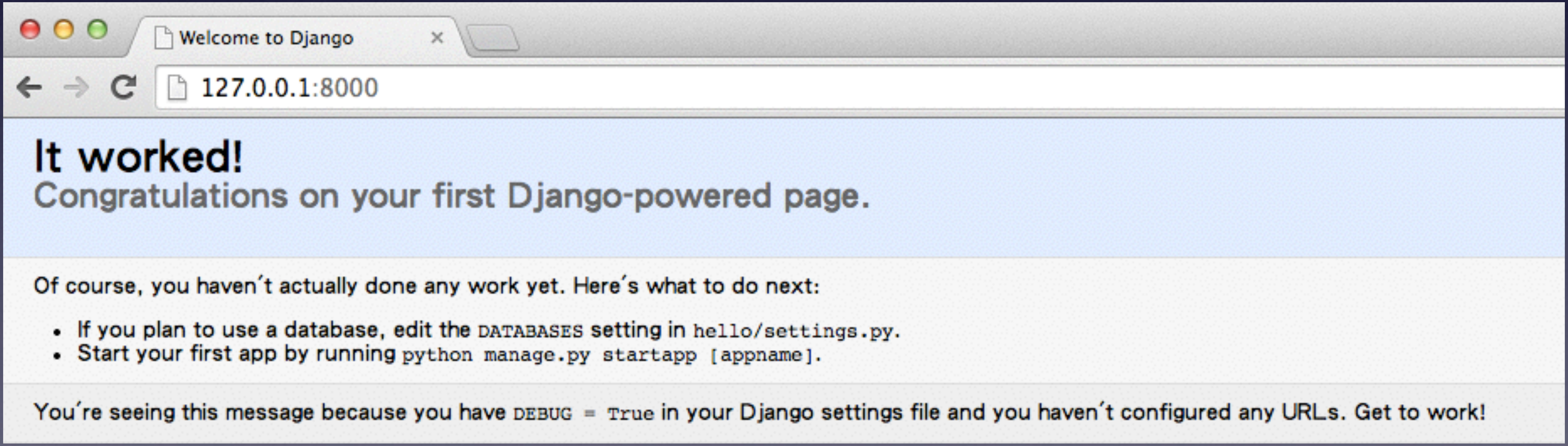
# start a new project

- $ django-admin.py startproject hello

```
> tree hello/
hello/
├── hello
|   ├── __init__.py
|   ├── __init__.pyc
|   ├── settings.py
|   ├── settings.pyc
|   ├── urls.py
|   ├── urls.pyc
|   ├── wsgi.py
|   └── wsgi.pyc
└── manage.py
```

## start a new project

- $ cd hello
- python manage.py runserver <your_ip>:port



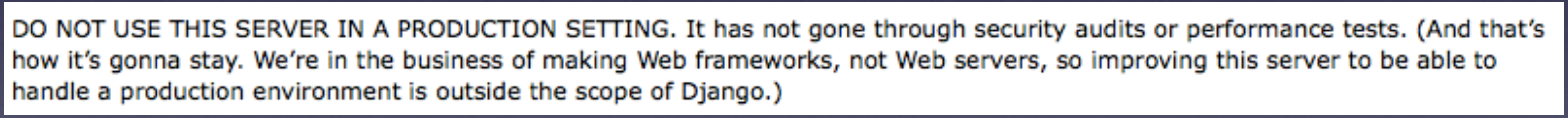
It worked!
Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Here's what to do next:

- If you plan to use a database, edit the `DATABASES` setting in `hello/settings.py`.
- Start your first app by running `python manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

## DO NOT USE DEV SERVER TO DEPLOY!


DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests. (And that's how it's gonna stay. We're in the business of making Web frameworks, not Web servers, so improving this server to be able to handle a production environment is outside the scope of Django.)

django-admin-runserver

# start a new app

- $ python manage.py startapp hihi

```
> tree hihi
hihi
├── __init__.py
├── models.py
├── tests.py
└── views.py
```

# start a new app

- put app name in settings.py
- edit hello/settings.py
- 116行

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'hihi',
)
```

# creating a view

- 讓project/urls.py指向app/urls.py **(只做一次)**
- 在views.py寫好相對應的function
- 讓app/urls.py指向views.py相對應的function

## project/urls.py指向app/urls.py

- edit hello/urls.py

```
url(r'hihi/', include('hihi.urls')),
```

RegEx module

## views.py裡相對應的function

- cd to your app directory, edit views.py

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("hello world!")
```

## app/urls.py指向views.py相對應的function

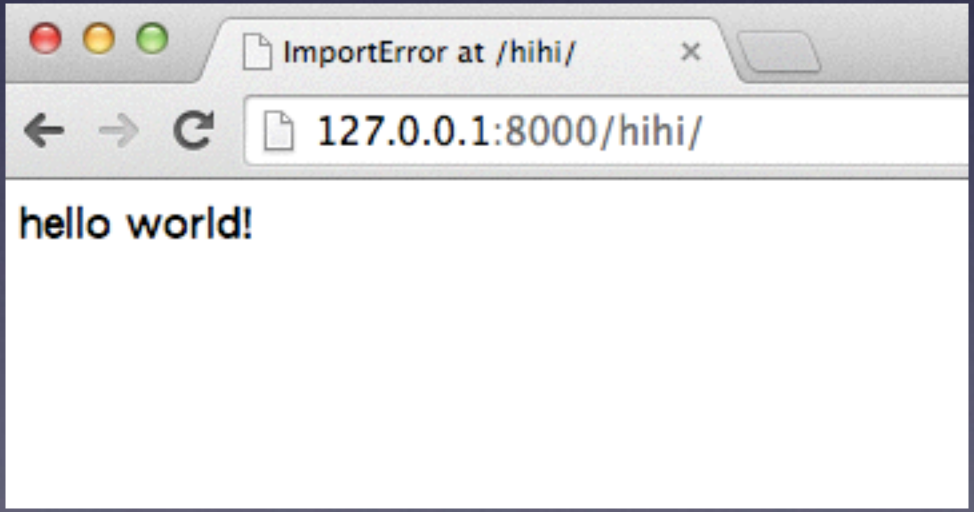
- create hihi/urls.py

```
from django.conf.urls import patterns, url
from hihi import views

urlpatterns = patterns('',
    url(r'^$', views.index, name='index'),
)
```

RegEx　mapping func　alias name

# Result

- $ python manage.py runserver



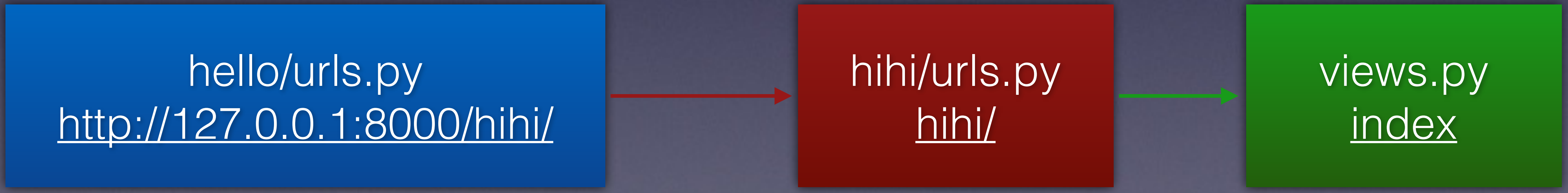
```
127.0.0.1:8000/hihi/
```

hello world!

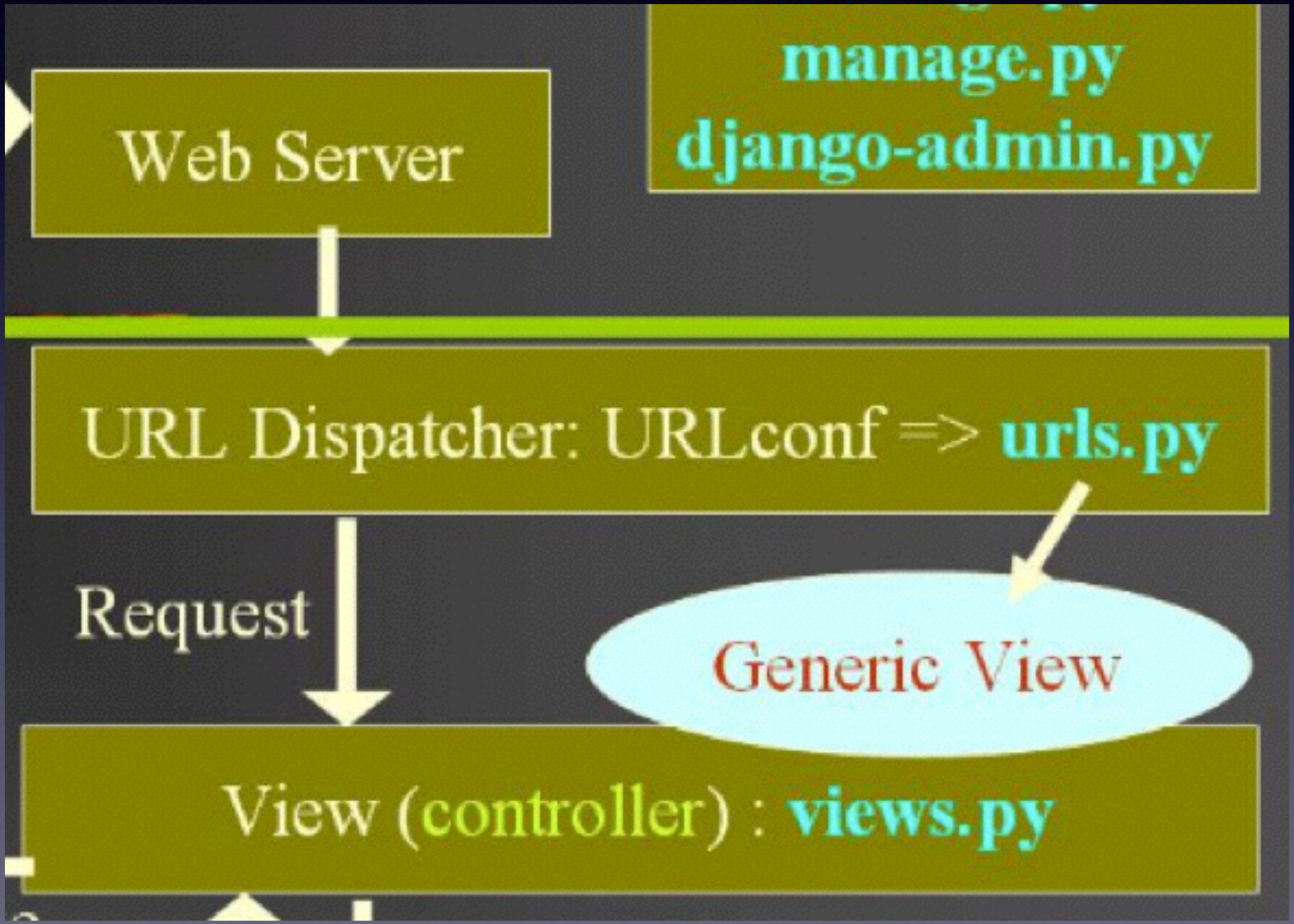
## Explanation

# URLs Mapping

- hello/urls.py把hihi/丟給hihi/urls.py處理
- hihi/urls.py傳request給相對應的views.function
- views.py中相對應的function回傳HttpResponse()



# URLs Mapping


manage.py
django-admin.py
Web Server
URL Dispatcher: URLconf => urls.py
Request
Generic View
View (controller) : views.py

## View

- Each view is responsible for doing one of two things

  1. returning an **HttpResponse** object containing the content for the requested page

  2. raising an exception such as **Http404**.

- The rest is up to you.

# Q & A

- Why not put everything in hello/urls.py ?
- Why not write all of your code in main() ?
- 降低耦合，提高內聚，以後可以重用

writing html in HttpResponse()?

```
    return HttpResponse("<!doctype html><html> ... </html>")
```

m(_ _)m (worship)

That's why we need template!

## hello world
## (plus template)

## setting

- make a templates directory under your app(hihi)
- put your templates under “**templates/app_name/**“
    - 避免不同app間的templates同名問題

```
$ mkdir -p templates/hihi
```

## setting

- editing your **settings.py** to tell django your templates directory

- 110行附近

```
TEMPLATE_DIRS = (
    # Put strings here, like "/home/html/
django_templates" or "C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative
paths.
    '/Users/mythnc/test/django_test/hello/hihi/
templates/hihi',
)
```

## creating view plus template

- 讓project/urls.py指向app/urls.py **(只做一次)**
- 在views.py寫好與**網頁名稱**相對應的function
- 讓app/urls.py指向views.py相對應的function
- 製作相對應的網頁檔案(template)

## create a template

- create **index.html** in your template directory

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>UccU</title>
    </head>
    <body>
        <h1>Hello World</h1>
    </body>
</html>
```

## modify your views.py

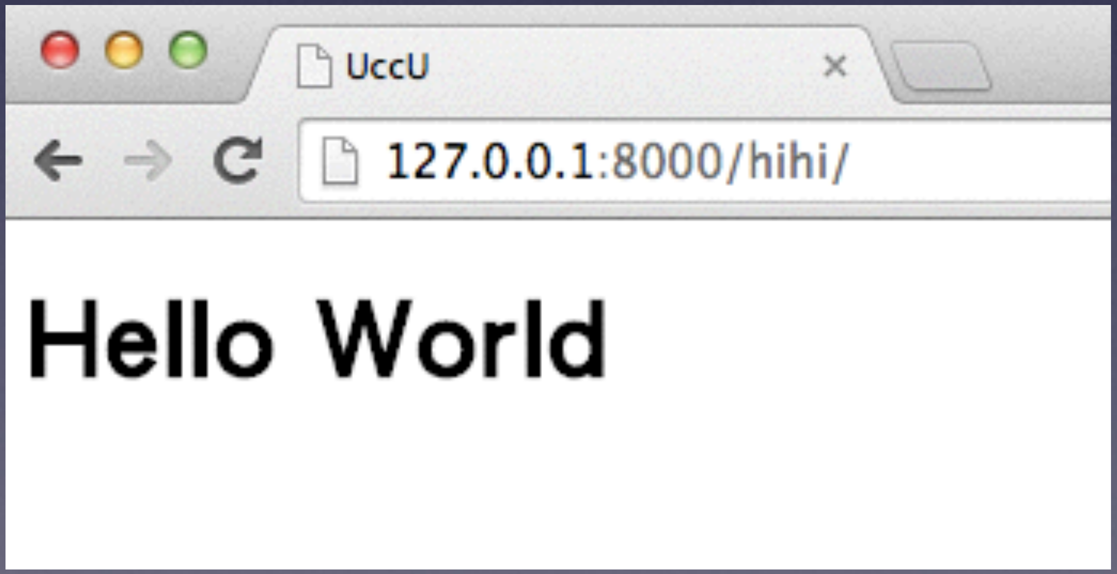
```
from django.shortcuts import render

def index(request):
    return render(request, 'index.html')
```

## shortcut

- render()
  - first argument: request
  - second argument: a template name
  - optional third argument: a dictionary
  - returns an **HttpResponse** object of the given template rendered with the given context

## Result

- $ python manage.py runserver



Hello World

## How about static files?

# conception

- static file: 相對於templates。靜止、不會變的。css, javascript, picture等。網頁元素即是。
- in development
  - Django: 我是好人我幫你
- in deployment
  - Django: 自己的static自己弄
  - setting by web server (apache, nginx, …etc)

## setting

- similar to templates setting
- create a static directory under your app
  - put file in “hihi/static/hihi”

## setting

- editing your **settings.py** to tell django your static directory

- 71行附近

```
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or
"C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not
relative paths.
    '/Users/mythnc/test/django_test/hello/hihi/
static/hihi',
)
```

# add a picture in index.html

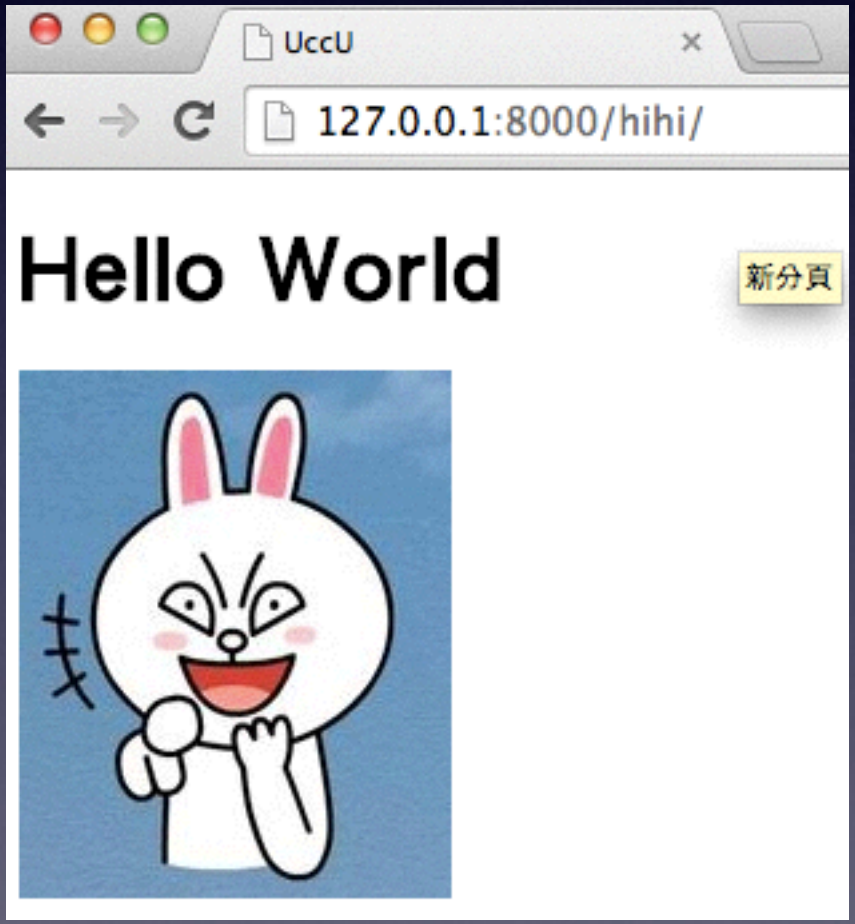
```
<!DOCTYPE html>
{% load staticfiles %}
<html>
    <head>
        <meta charset="UTF-8">
        <title>UccU</title>
    </head>
    <body>
        <h1>Hello World</h1>
        <img src="{% static ‘hihi/uccu.jpg' %}" alt="uccu"/>
    </body>
</html>
```

# Django template tags

- {% xxx %} template tag and filter
- {{ xxx }} 變數
- {% load staticfiles %}
  - Its loads the {% static %} template tag from the staticfiles template library.
  - The {% static %} template tag generates the absolute URL of the static file.

## Result

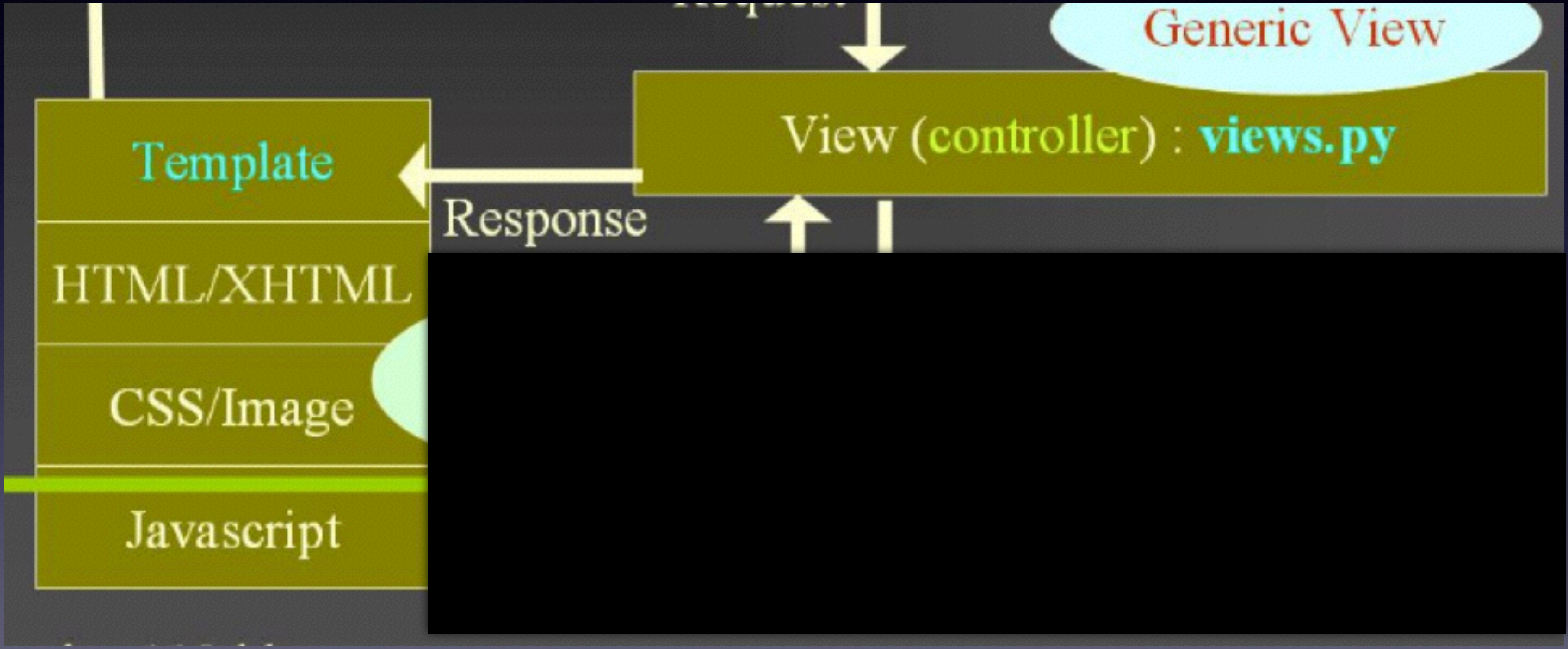
- python manage.py runserver



Hello World

# default static path

- STATIC_URL = ‘/static/‘ in settings.py
- http://127.0.0.1:8000/static/hihi/uccu.jpg


Generic View
View (controller) : views.py
Template
Response
HTML/XHTML
CSS/Image
Javascript
[redacted]

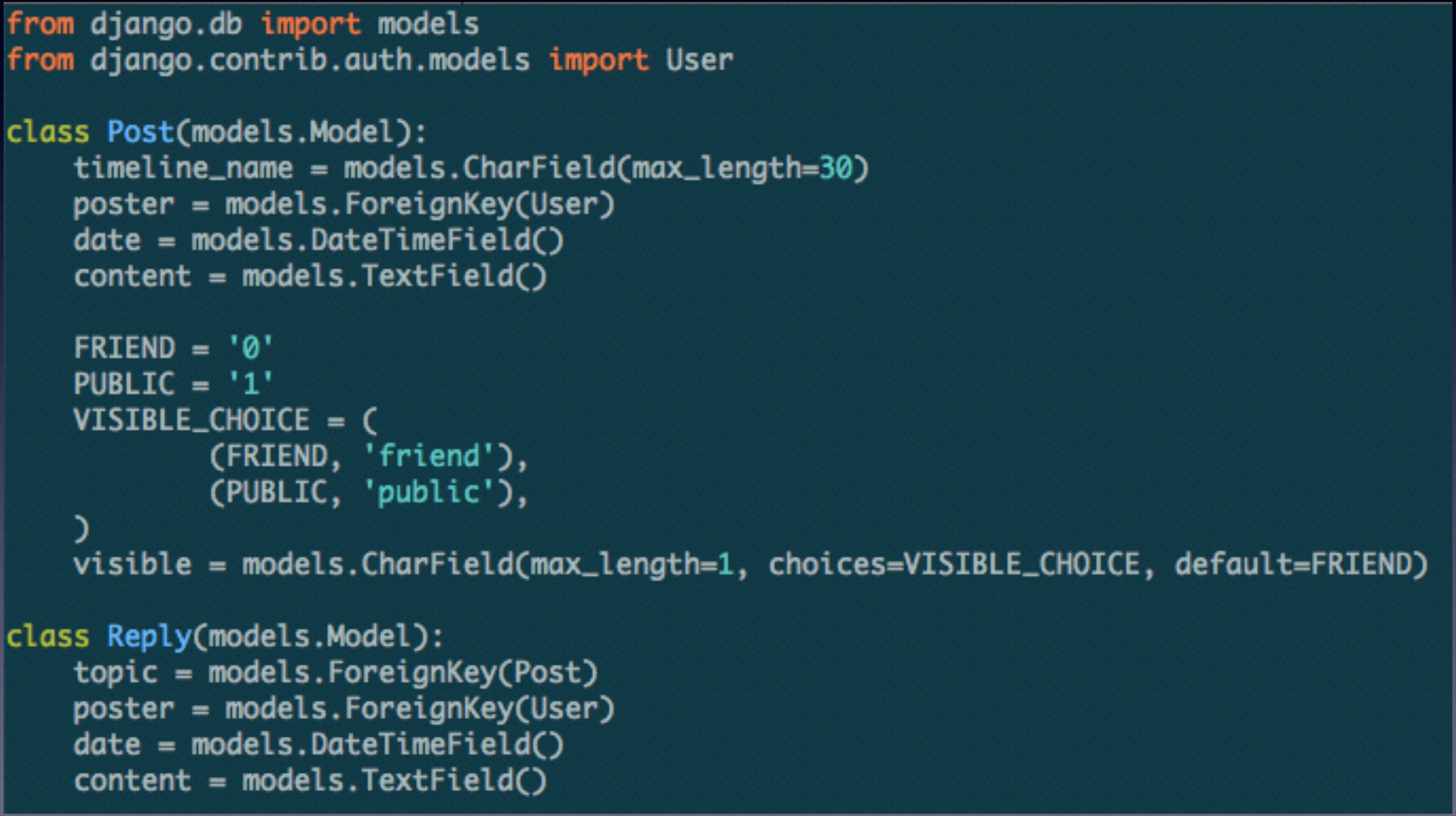
## Model

## Object Relational Mapping

- a programming technique for converting data between incompatible type systems in object-oriented programming languages.

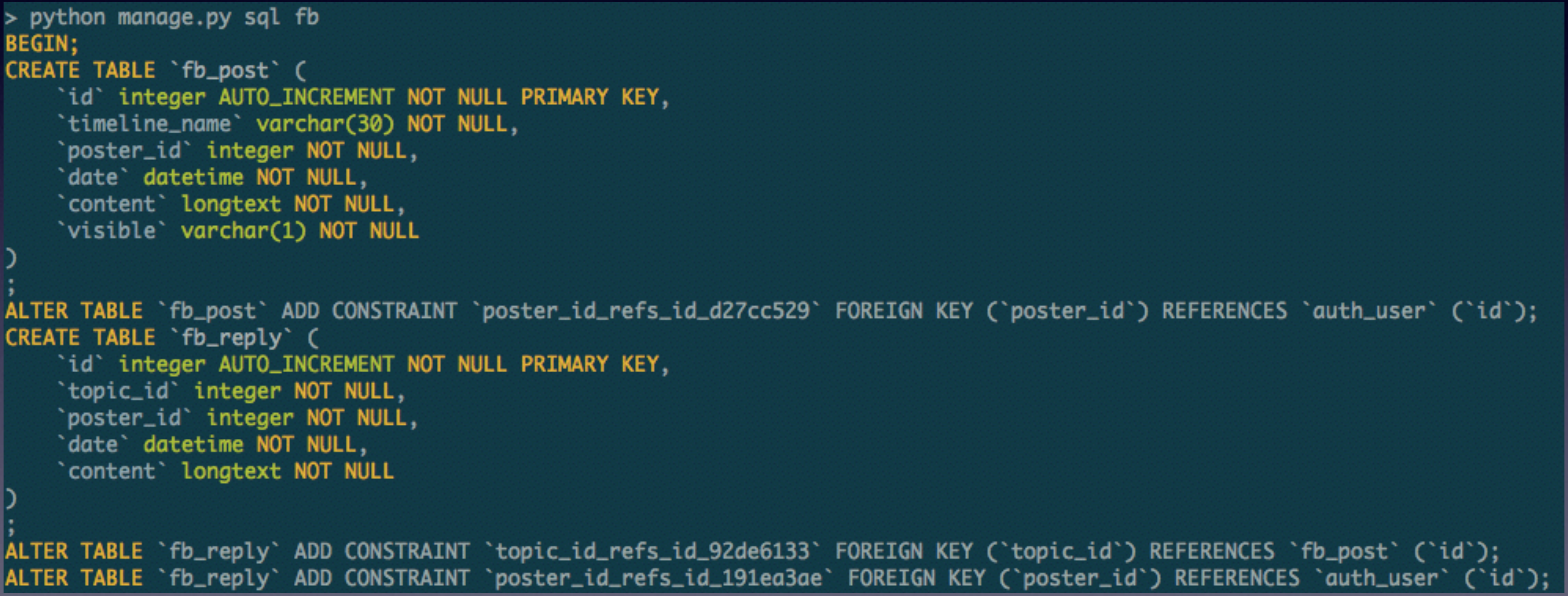
- This creates, in effect, a "virtual object database" that can be used from within the programming language.

# create your models.py



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

class Post(models.Model):
    timeline_name = models.CharField(max_length=30)
    poster = models.ForeignKey(User)
    date = models.DateTimeField()
    content = models.TextField()

    FRIEND = '0'
    PUBLIC = '1'
    VISIBLE_CHOICE = (
            (FRIEND, 'friend'),
            (PUBLIC, 'public'),
    )
    visible = models.CharField(max_length=1, choices=VISIBLE_CHOICE, default=FRIEND)

class Reply(models.Model):
    topic = models.ForeignKey(Post)
    poster = models.ForeignKey(User)
    date = models.DateTimeField()
    content = models.TextField()
```

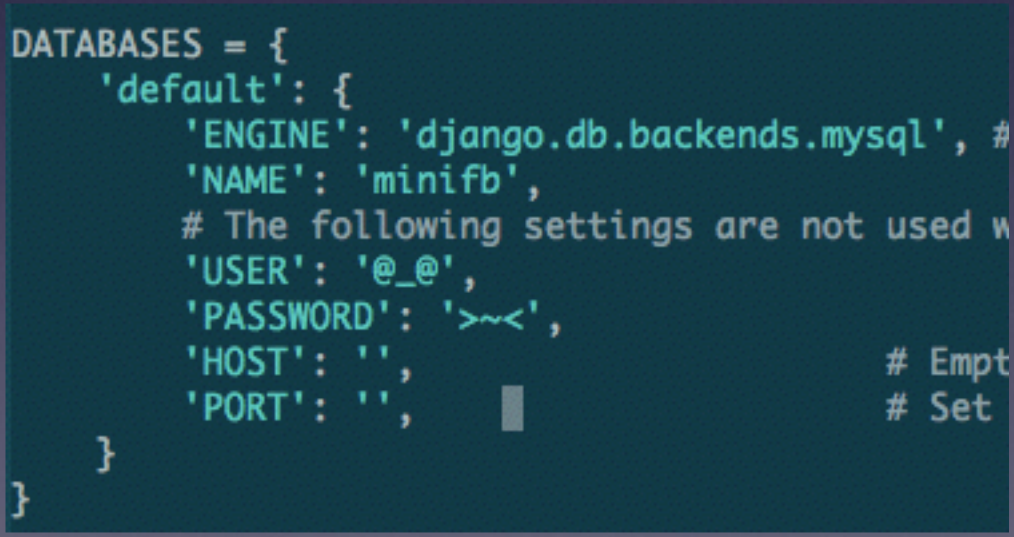
## sql app_name



```
> python manage.py sql fb
BEGIN;
CREATE TABLE `fb_post` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `timeline_name` varchar(30) NOT NULL,
    `poster_id` integer NOT NULL,
    `date` datetime NOT NULL,
    `content` longtext NOT NULL,
    `visible` varchar(1) NOT NULL
)
;
ALTER TABLE `fb_post` ADD CONSTRAINT `poster_id_refs_id_d27cc529` FOREIGN KEY (`poster_id`) REFERENCES `auth_user` (`id`);
CREATE TABLE `fb_reply` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `topic_id` integer NOT NULL,
    `poster_id` integer NOT NULL,
    `date` datetime NOT NULL,
    `content` longtext NOT NULL
)
;
ALTER TABLE `fb_reply` ADD CONSTRAINT `topic_id_refs_id_92de6133` FOREIGN KEY (`topic_id`) REFERENCES `fb_post` (`id`);
ALTER TABLE `fb_reply` ADD CONSTRAINT `poster_id_refs_id_191ea3ae` FOREIGN KEY (`poster_id`) REFERENCES `auth_user` (`id`);
```

# setting before syncdb

- Database setup
  - edit DATABASES in **settings.py**
    - ENGINE: django.db.backends.mysql
    - NAME: The name of your database
    - USER: root account of mysql
    - PASSWORD: root pwd of mysql
    - HOST and PORT don’t matter
- Use “python manage.py syncdb” to sync database.

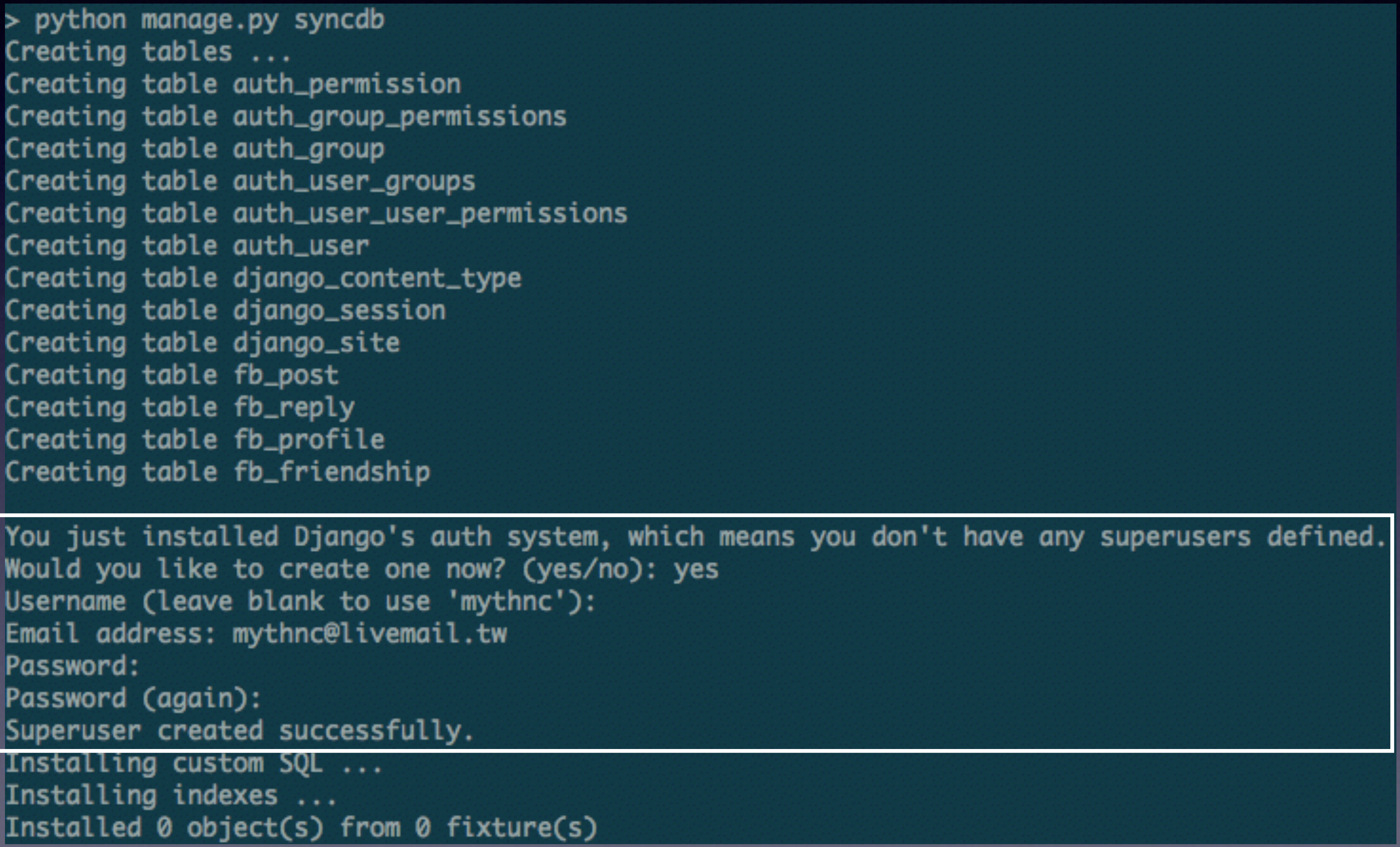


```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql', #
        'NAME': 'minifb',
        # The following settings are not used w
        'USER': '@_@',
        'PASSWORD': '>~<',
        'HOST': '',                          # Empt
        'PORT': '',                          # Set
    }
}
```

## setting before syncdb

1. $ mysql.server start
2. create a corresponding name in mysql
    - mysql> create database minifb;
3. $ python manage.py syncdb

# syncdb



```
> python manage.py syncdb
Creating tables ...
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
Creating table django_site
Creating table fb_post
Creating table fb_reply
Creating table fb_profile
Creating table fb_friendship

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 'mythnc'):
Email address: mythnc@livemail.tw
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
```

## Caution about syncdb

- When adding a new model to models.py then do syncdb is OK

- updating an existing model to models.py then do syncdb will **ERROR**

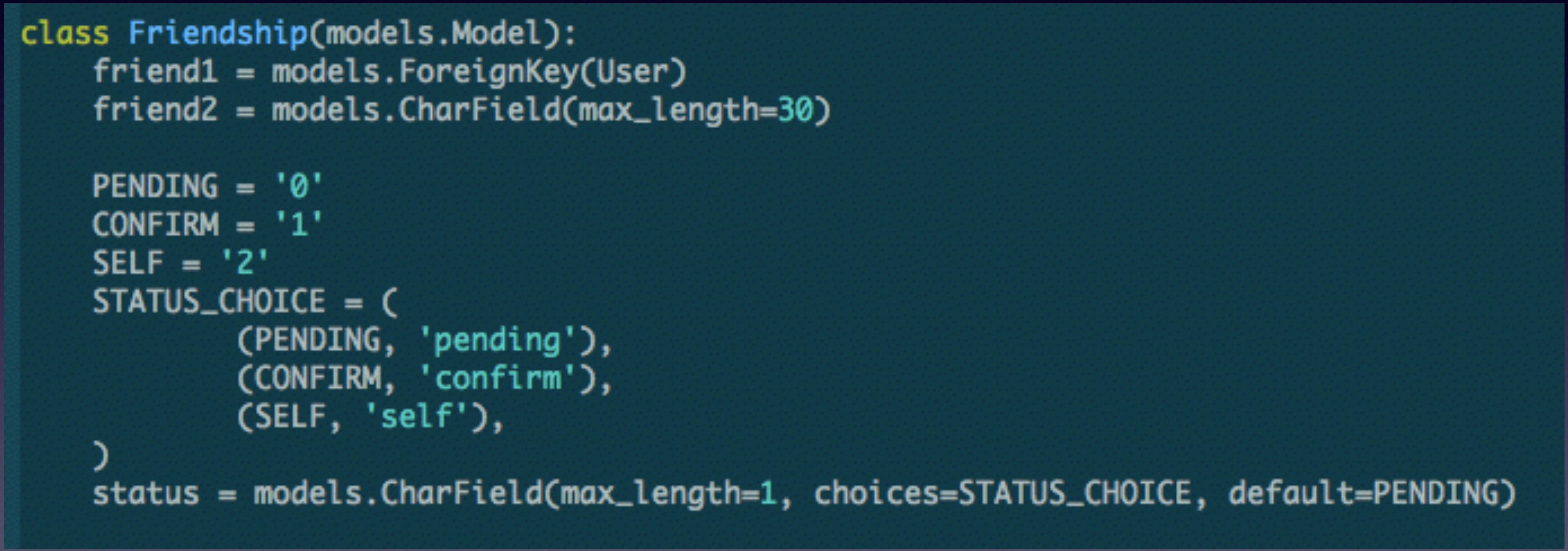
  - delete database then recreate it

  - $ python manage.py syncdb

- Use South

# Django database API

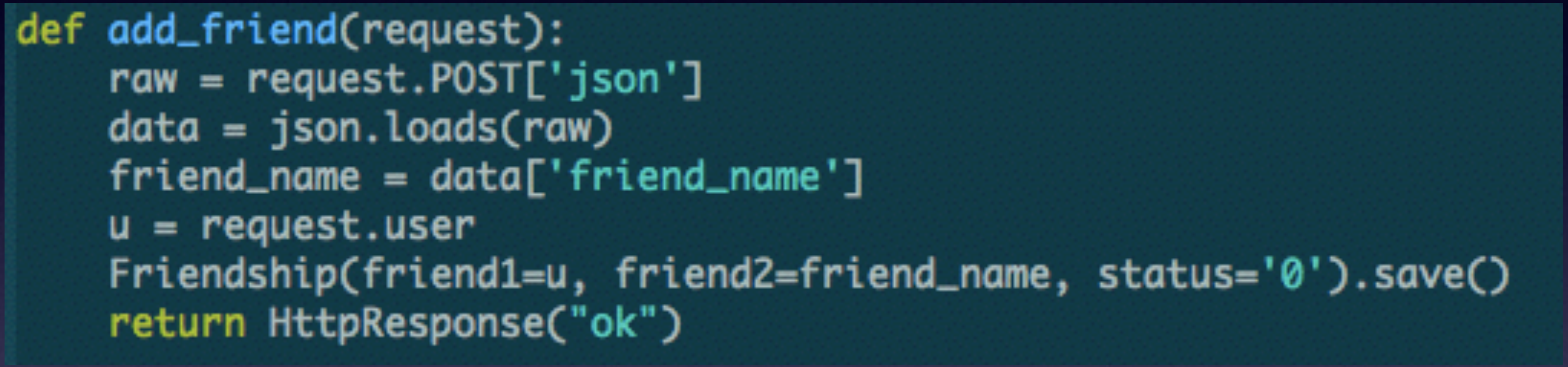
- 以Friendship為例，展示CRUD



```
class Friendship(models.Model):
    friend1 = models.ForeignKey(User)
    friend2 = models.CharField(max_length=30)

    PENDING = '0'
    CONFIRM = '1'
    SELF = '2'
    STATUS_CHOICE = (
            (PENDING, 'pending'),
            (CONFIRM, 'confirm'),
            (SELF, 'self'),
    )
    status = models.CharField(max_length=1, choices=STATUS_CHOICE, default=PENDING)
```

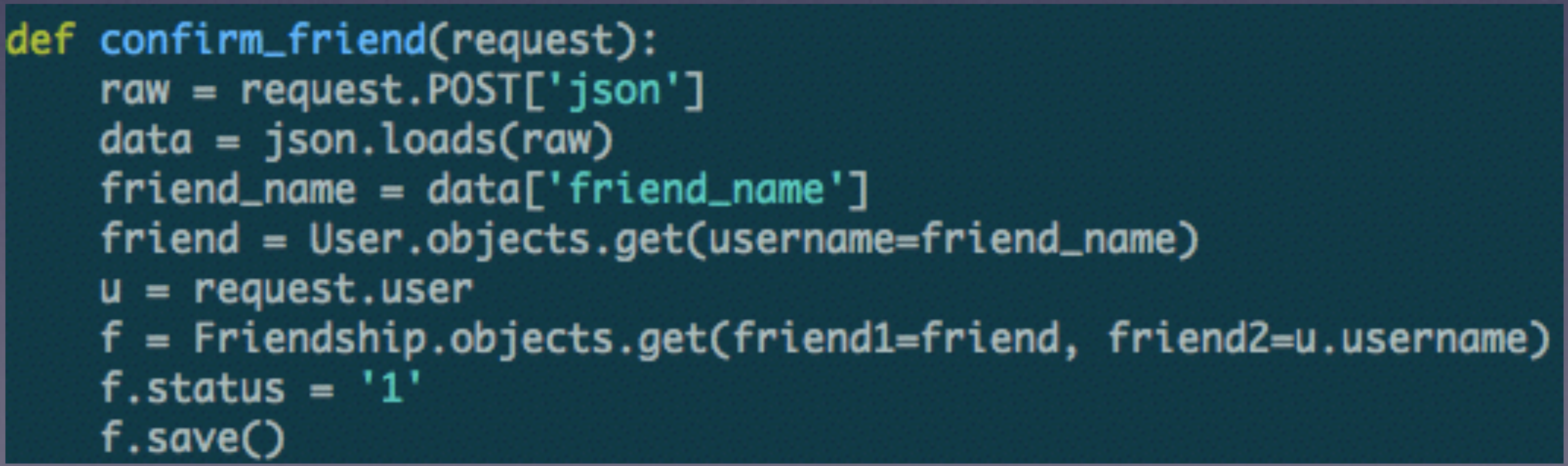
## Django database API

- Create



```python
def add_friend(request):
    raw = request.POST['json']
    data = json.loads(raw)
    friend_name = data['friend_name']
    u = request.user
    Friendship(friend1=u, friend2=friend_name, status='0').save()
    return HttpResponse("ok")
```

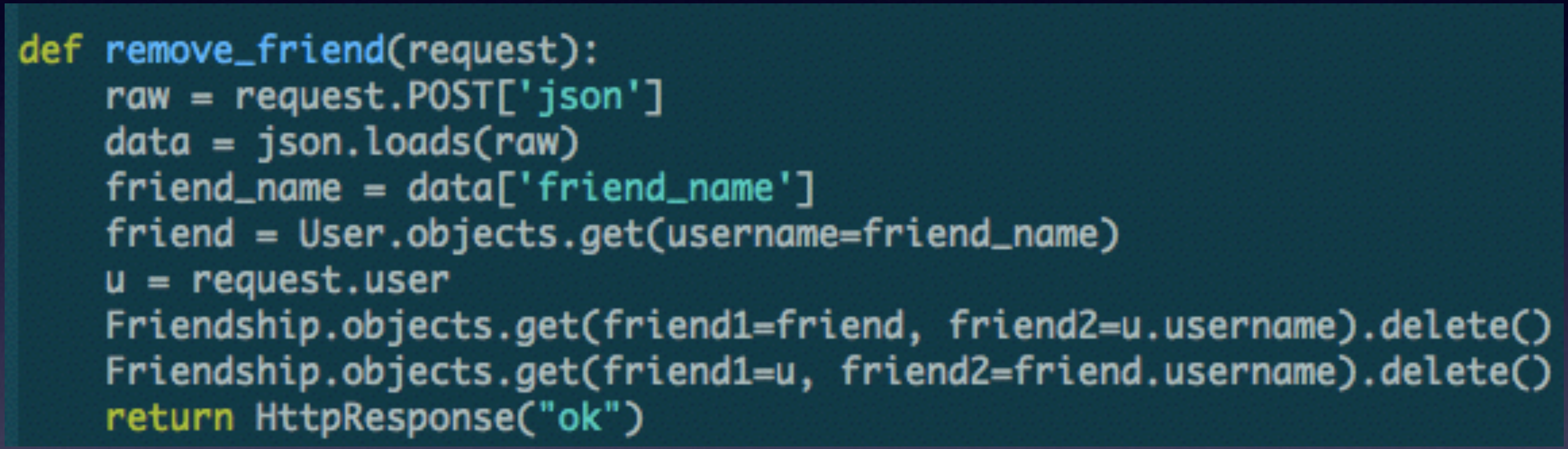
- Retrive, Update



```python
def confirm_friend(request):
    raw = request.POST['json']
    data = json.loads(raw)
    friend_name = data['friend_name']
    friend = User.objects.get(username=friend_name)
    u = request.user
    f = Friendship.objects.get(friend1=friend, friend2=u.username)
    f.status = '1'
    f.save()
```

## Django database API

- Delete


```
def remove_friend(request):
    raw = request.POST['json']
    data = json.loads(raw)
    friend_name = data['friend_name']
    friend = User.objects.get(username=friend_name)
    u = request.user
    Friendship.objects.get(friend1=friend, friend2=u.username).delete()
    Friendship.objects.get(friend1=u, friend2=friend.username).delete()
    return HttpResponse("ok")
```

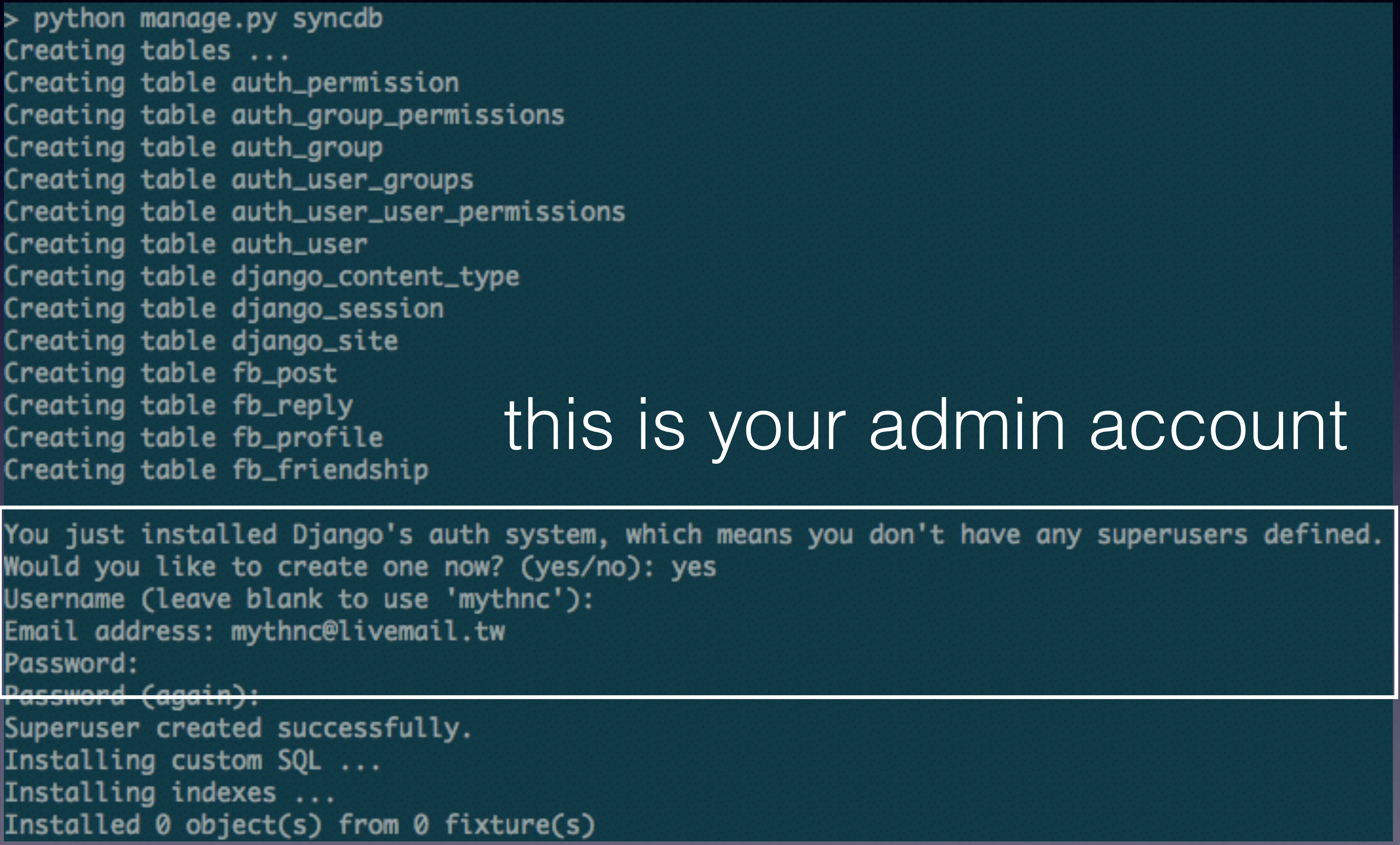
# Reference

- models
- model relations
- Database API reference
- Underscore用法

## Admin

- similar to phpMyAdmin
- visualization data in model
- 好不好用？見仁見智。
- 青菜蘿蔔各有所好

## create your admin account

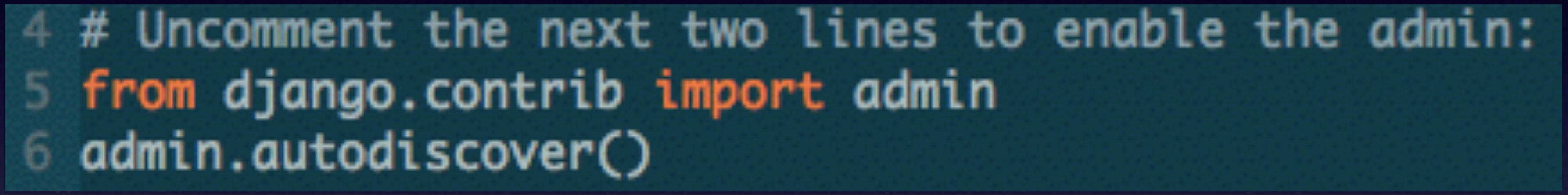


```
> python manage.py syncdb
Creating tables ...
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table fb_post
Creating table fb_reply
Creating table fb_profile
Creating table fb_friendship

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 'mythnc'):
Email address: mythnc@livemail.tw
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
```

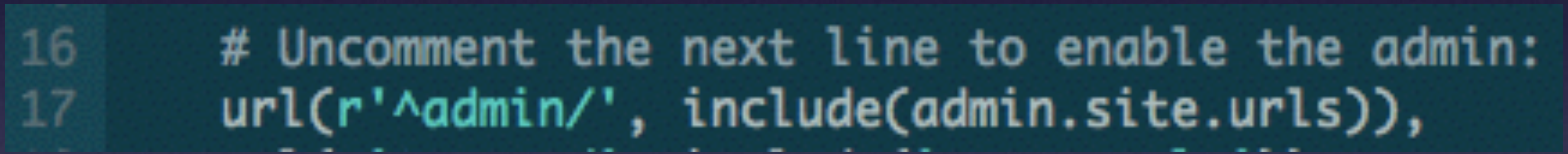
this is your admin account

# modify project/urls.py

- uncomment admin (共3行)



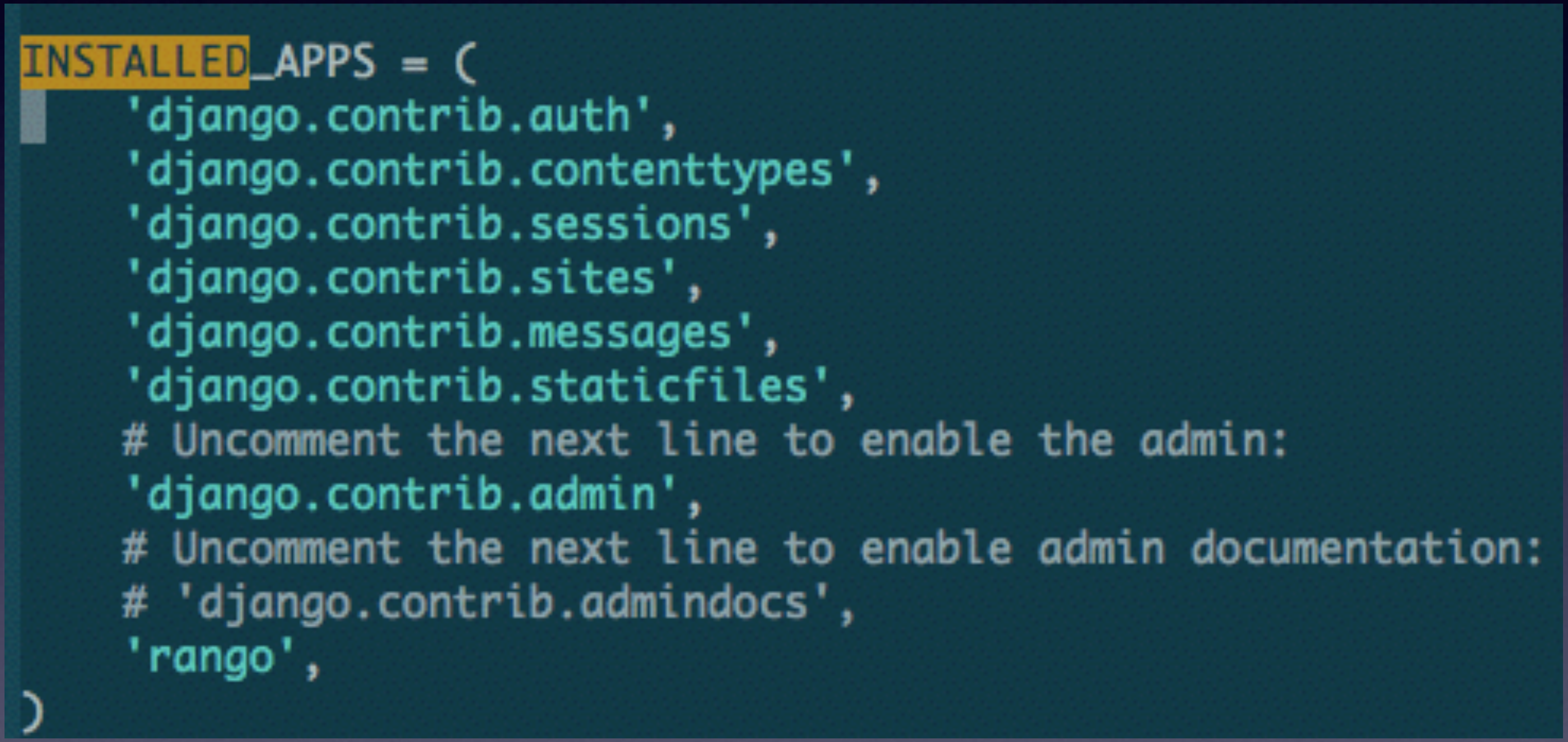
```
# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()
```



```
    # Uncomment the next line to enable the admin:
    url(r'^admin/', include(admin.site.urls)),
```

## modify settings.py

- uncomment admin in INSTALLED_APPS



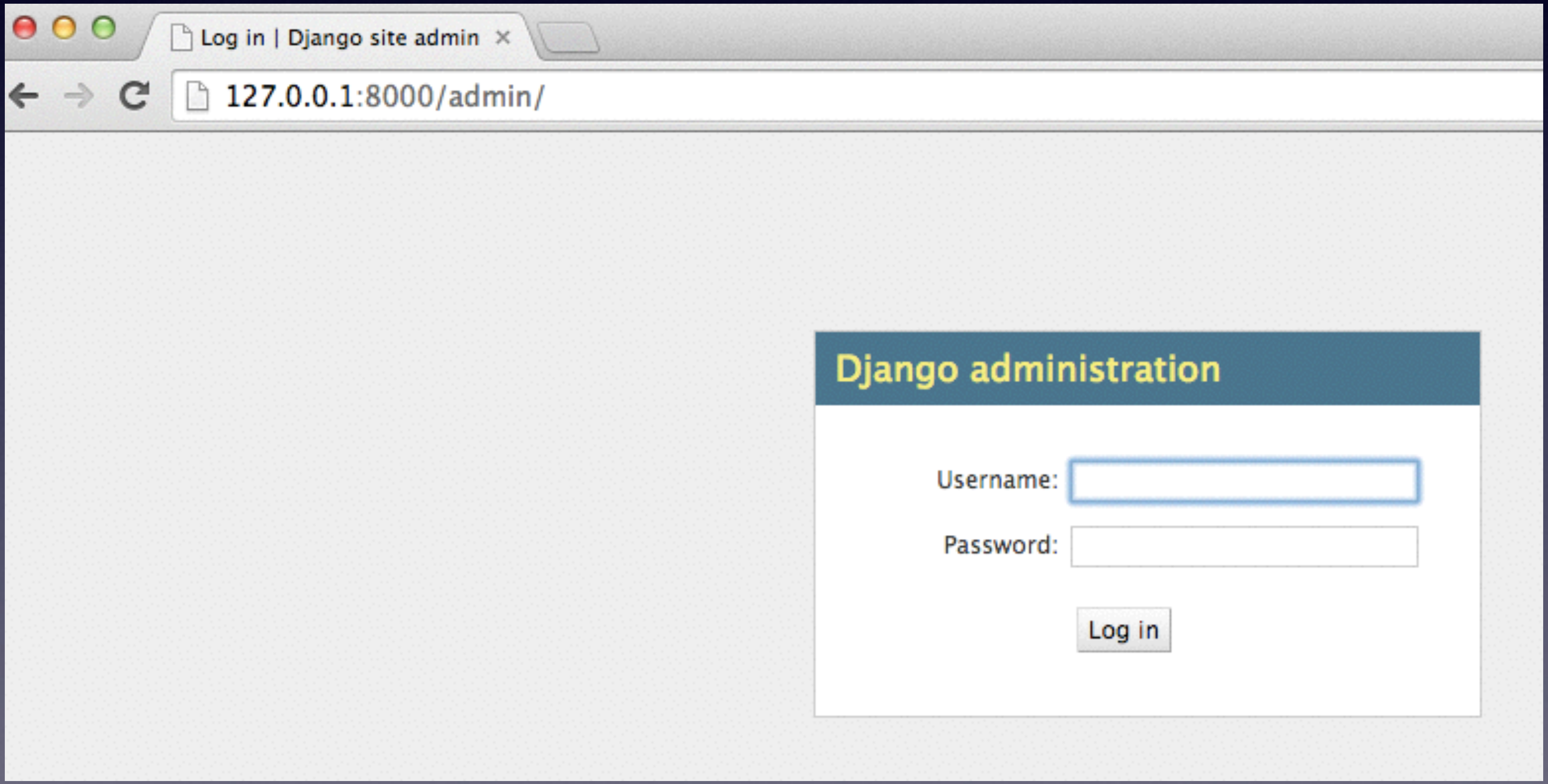
```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'rango',
)
```

- Don’t forget to syncdb again
- $ python manage.py syncdb

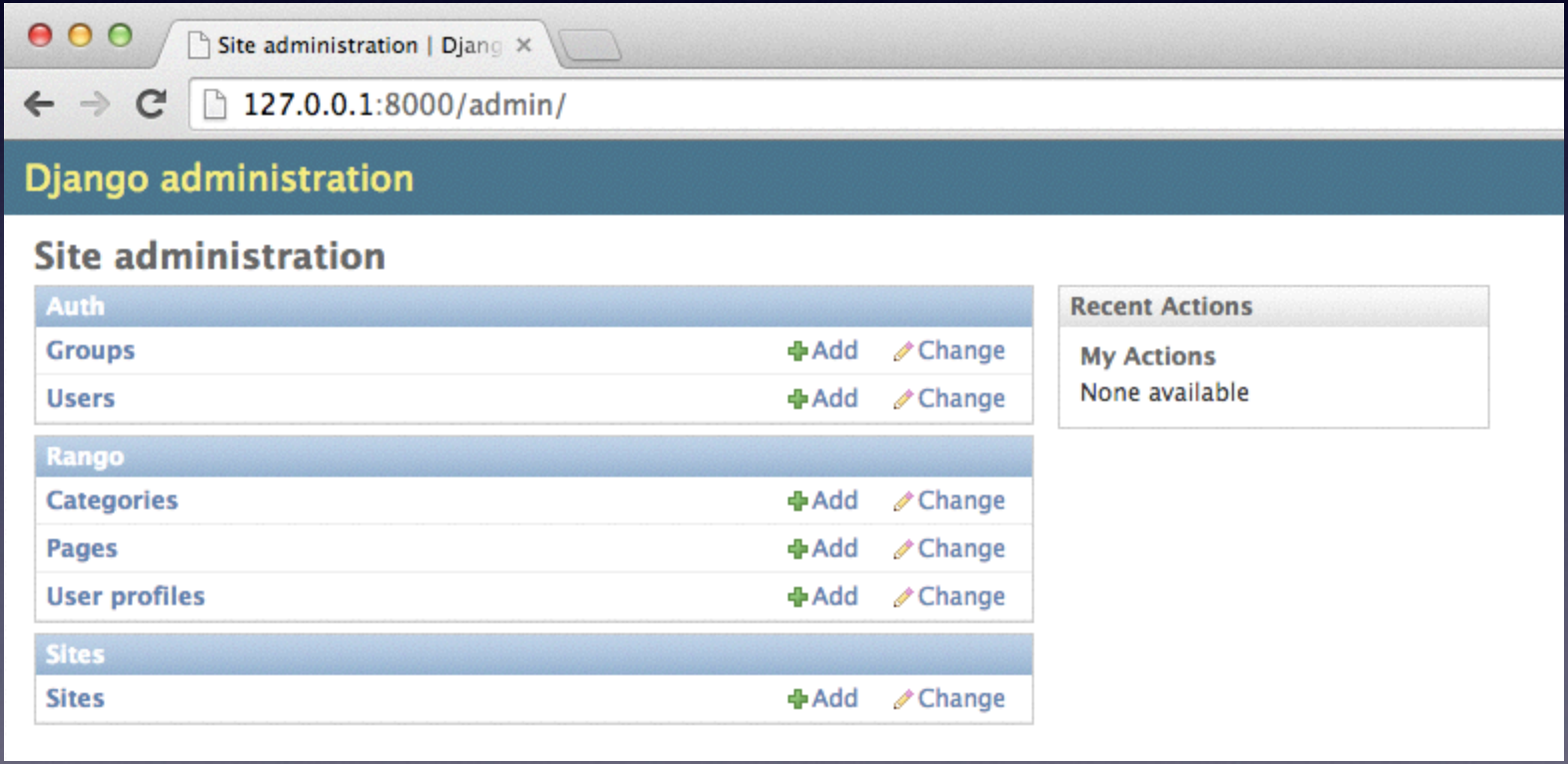
## create admin.py in app

```
from django.contrib import admin
from rango.models import Category, Page, UserProfile

class PageAdmin(admin.ModelAdmin):
    list_display = ('category', 'title', 'url')

admin.site.register(Category)
admin.site.register(Page, PageAdmin)
admin.site.register(UserProfile)
```

# Result


Log in | Django site admin
127.0.0.1:8000/admin/
Django administration
Username:
Password:
Log in

# Result



Site administration | Djang

127.0.0.1:8000/admin/

Django administration

Site administration

| Auth | | |
|---|---|---|
| Groups | Add | Change |
| Users | Add | Change |

| Rango | | |
|---|---|---|
| Categories | Add | Change |
| Pages | Add | Change |
| User profiles | Add | Change |

| Sites | | |
|---|---|---|
| Sites | Add | Change |

Recent Actions

My Actions

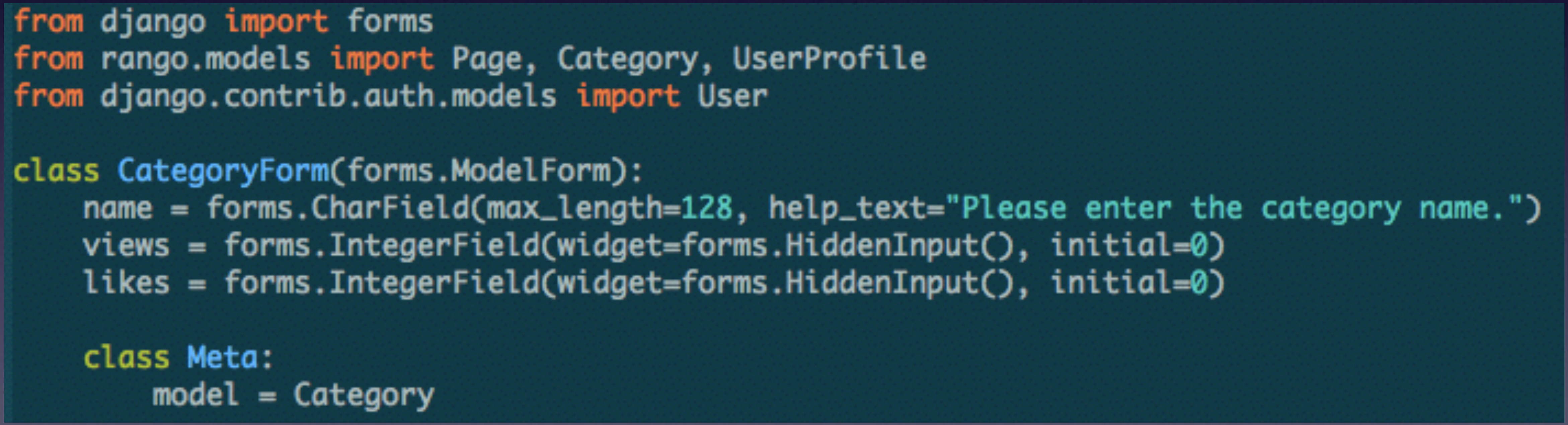
None available

## Form

## step

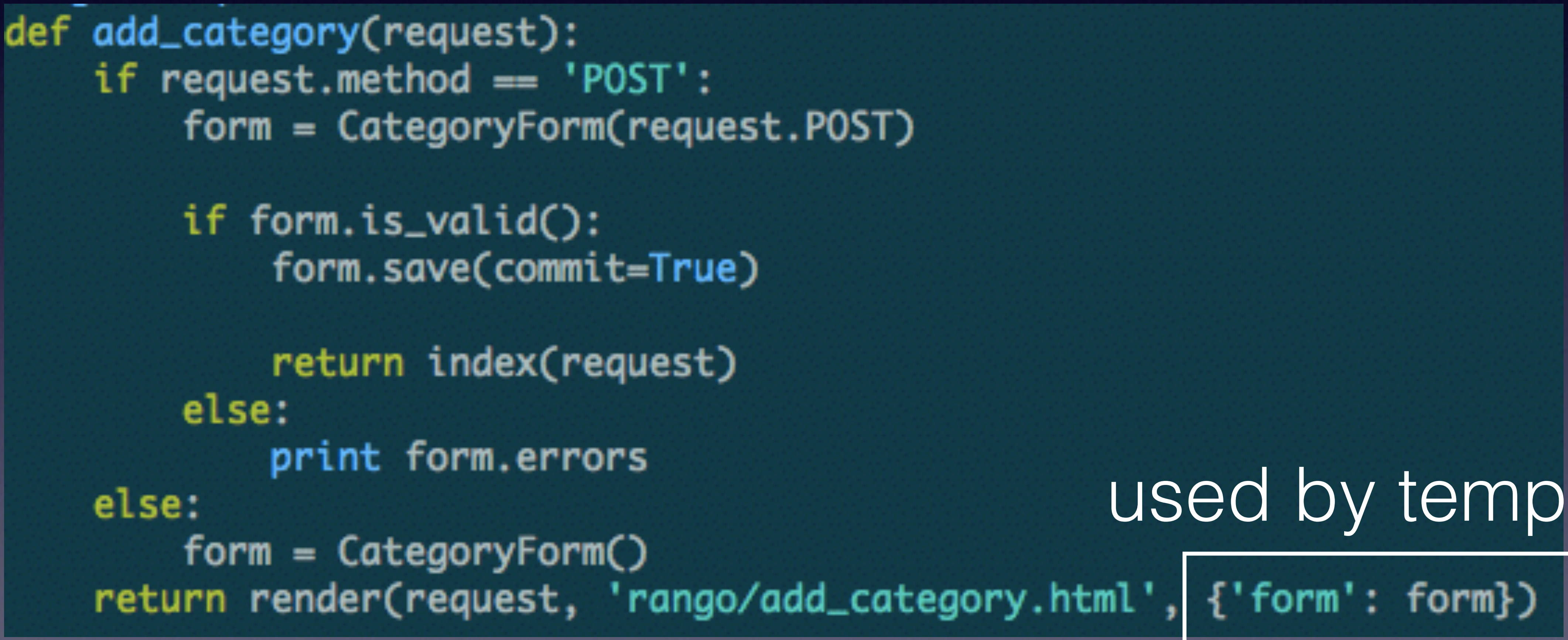
1. create forms.py in app (只做一次)
2. create form class
3. create or update a view to handle form
4. create or update a template to display form
5. add a urlpattern to map to the new view

## create forms.py



```
from django import forms
from rango.models import Page, Category, UserProfile
from django.contrib.auth.models import User

class CategoryForm(forms.ModelForm):
    name = forms.CharField(max_length=128, help_text="Please enter the category name.")
    views = forms.IntegerField(widget=forms.HiddenInput(), initial=0)
    likes = forms.IntegerField(widget=forms.HiddenInput(), initial=0)

    class Meta:
        model = Category
```

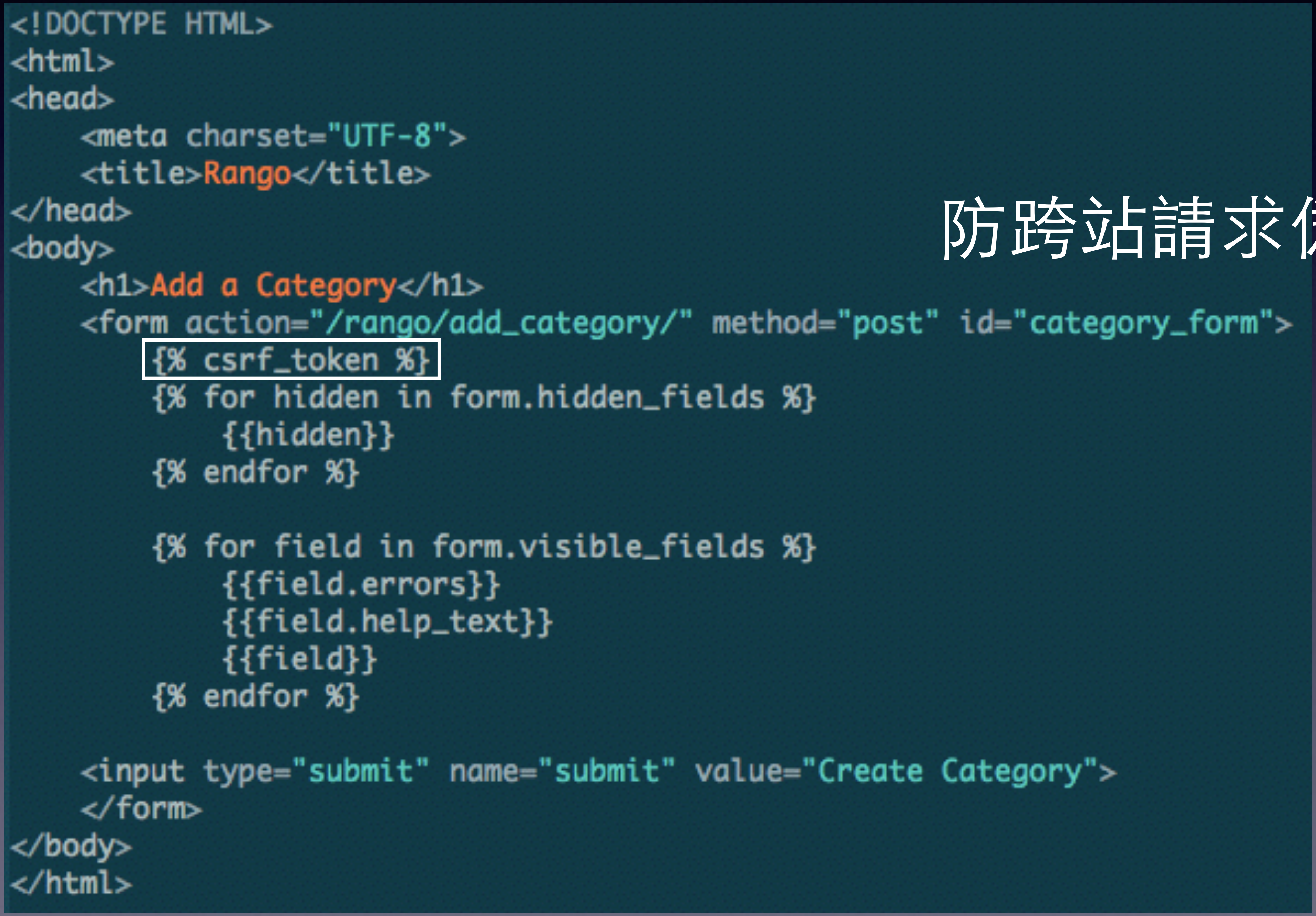
# modify views.py



```
def add_category(request):
    if request.method == 'POST':
        form = CategoryForm(request.POST)

        if form.is_valid():
            form.save(commit=True)

            return index(request)
        else:
            print form.errors
    else:
        form = CategoryForm()
    return render(request, 'rango/add_category.html', {'form': form})
```

used by template

## create new template



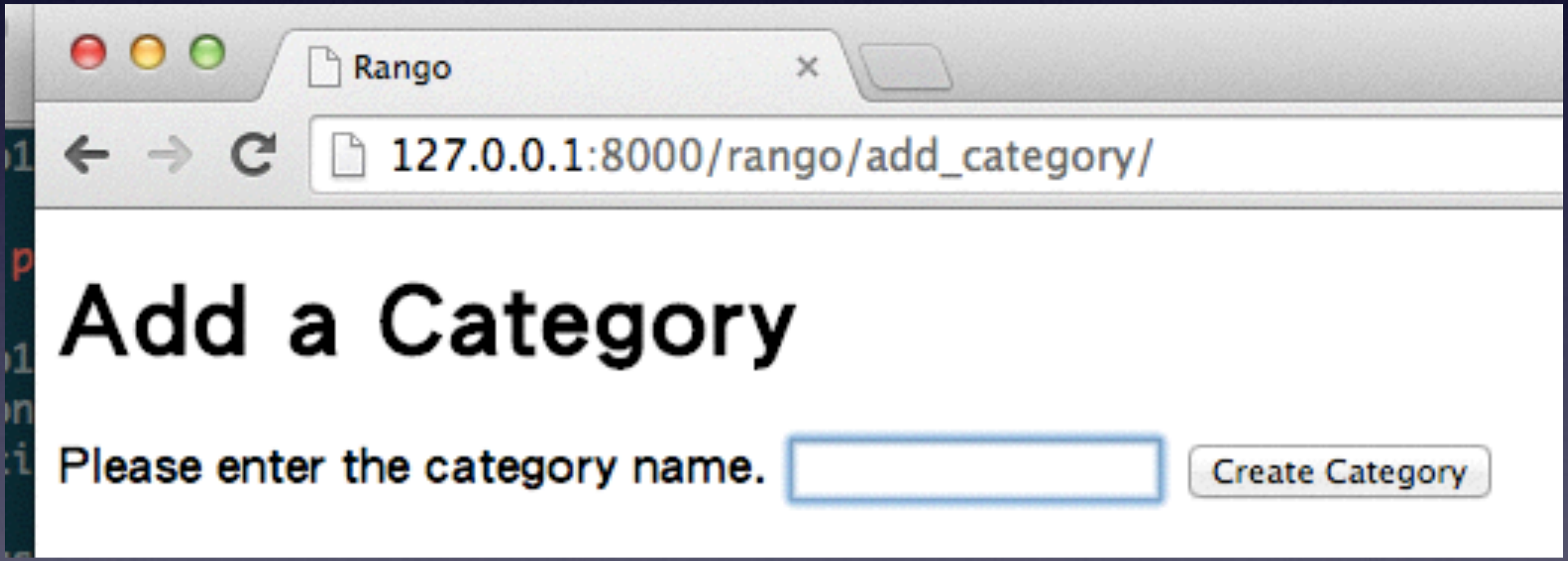
```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8">
    <title>Rango</title>
</head>
<body>
    <h1>Add a Category</h1>
    <form action="/rango/add_category/" method="post" id="category_form">
        {% csrf_token %}
        {% for hidden in form.hidden_fields %}
            {{hidden}}
        {% endfor %}

        {% for field in form.visible_fields %}
            {{field.errors}}
            {{field.help_text}}
            {{field}}
        {% endfor %}

    <input type="submit" name="submit" value="Create Category">
    </form>
</body>
</html>
```

防跨站請求偽造攻擊

## update urls.py

```
url(r'^add_category/$', views.add_category,
name='add_category'),
```

## Result



Add a Category

Please enter the category name.

## User Authentication

- Django有自己的使用者認證機制。不用在自己寫，只要學會怎麼使用。包含註冊表單、修改表單、登入、登出，都不用自己刻！OP!

## Reference

# If you want to learn more…

- virtualenv
- Django documentation (超級硬)
- Django tutorial part1 - part6
- Tango with Django (部分寫法老舊，小心服用)

- Learning programming language and framework are two different thing

# Q&A