# DENEYİMLİ PROGRAMCILAR İÇİN *DJANGO*'YA GİRİŞ

Serdar Dalgıç
Yazılım Geliştirme Uzmanı – Markafoni
23 Kasım 2013 – Yeditepe Üniversitesi

markafoni Her alışveriş bir aşk

# Ben Kimim?

- Markafoni'de Yazılım Geliştirme Uzmanı olarak çalışıyorum.
- ODTÜ Bilgisayar Mühendisliği Lisans, Boğaziçi Üniversitesi Yazılım Mühendisliği Yüksek Lisans mezunuyum.
- TÜBİTAK bünyesinde Pardus Projesi'nde çalıştım. (Python'la tanışmam)
- ~~Web Geliştirme ile tanışmam yakın bir döneme denk geliyor~~. 1 sene oldu :)
- Github: github.com/serdardalgic
- Web sitesi: serdardalgic.org
- Twitter: twitter.com/serdaroncode
- Python-Istanbul grubuna üyeyim. Freenode'da #pyistanbul IRC kanalı
- Uzun zamandır LKD üyesiyim.

# Neler Anlatacağım? - 1

- Neden Django?
  - Nerelerde Kullanılıyor?
- Python
  - Python'un özellikleri (özetin özeti)
- Django
  - Django Kurulumu
  - Django'da işler nasıl yürüyor?
  - Projeler ve Paketler
  - Proje ayar dosyaları: settings.py, manage.py
  - Paketlerdeki dosyalar: models.py, views.py, forms.py
  - MTV is our MVC

# Neler Anlatacağım? - 2

- Django *(devam)*
  - Django uygulama dosyaları: urls.py, views.py, models.py
  - İşinize yarayacak Django uygulamaları
    - django-extensions,
    - south,
    - ipython, ipdb,
    - v.s..
- Öneriler

# django

- Python?
- “For perfectionists with deadlines” - “Teslim tarihleri olan mükemmeliyetçiler için” Ne demek?
- __*Buraya “Django Unchained” esprisi gelecek.*__
- İsim babası müzisyen *Django Reinhardt* bu arada..



*Django ile ilgili (neredeyse) herşey:*
https://www.djangoproject.com/

# Neden Django? - 1

- Hızlı ve temiz bir geliştirme modeli sunması, pragmatik yapısı
- DRY (Kendini Tekrar Etme!) Prensibi
- Pluggable (Tak – Çıkar) Uygulamalar
- Geliştirme ve Test için hafif, tek başına çalışan sunucu
- Teste yatkınlık
- Topluluk Desteği (Açık Kaynaklı!)
  - https://www.djangoproject.com/community/
  - https://www.djangopackages.com/ ve http://pinaxproject.com/
  - https://groups.google.com/group/python-istanbul
- Lezzetli dökümantasyonu

# Neden Django? - 2

- Python!
- ORM (Nesne-veritabanı eşleştirmesi), farklı veritabanlarının kullanımına izin vermesi
- Otomatik Admin Arayüzü
- Regex temelli URL Dizaynı
- Template Desteği
- Caching Yapısı (veritabanı, local memory, redis v.s.)
- i18n, l10n
- Komut satırı arayüzü
- Geniş deployment olanakları

# Nerelerde kullanılıyor?

- Pinterest
- Disqus
- Instagram
- Mozilla
- Bitbucket

..

- Markafoni
- Morhipo
- Metglobal (tatil.com)

..

- Newyorktimes.com
- Guardian.co.uk

Kaynak: http://www.djangosites.org/

# PYTHON (özetin özeti)

- Dinamik bir programlama dili
- Yorumlanabilir (interpreted)
- Duck-typing
- Nesne Yönelimli (Object-Oriented)
- Fonksiyonel bir dil
- Güçlü namespace yapısı
- Exception desteği
- Obsesif-Kompülsifler için :) Pep8 ile düzenli, okunabilir kod yazımı
- Batteries Included! (İhtiyacın olan kitaplıkların beraberinde gelmesi)
- Topluluk Desteği!

# PYTHON

- Interaktif Kabuk(Shell)
- Boolean ve Null değerler: *True, False, None*
- String, 'python stringi', *“python string!”, u“çişüğ”*
- String İşlemleri: *“foo” + “bar”, “foo”[:1], len(“foo”), “foo”.upper()*
- Nümerik Tipler: *42, 42.0, 42L*
- Listeler, Tuple'lar, Set'ler: *['a', 'b', 'c'], ('python', 'owtg'), set(['a', 'b', 'c'])*
- Liste İşlemleri: *[...](0), [...].append(4), [...].pop(), len([...])*
- Sözlükler: *{'key3': 'value3', 'konferans': 'owtgconf2013'}*
- Sözlük İşlemleri: *{...}['key3' ], {...}.keys, len({...}), {...}.items*

# PYTHON

- Atama ve Karşılaştırma: *foo = 'bar', foo == 'hede', foo!='hodo', foo is None, foo is not None*
- Akış Kontrolü: *if..elif..else, for, foreach, while..*
- Fonksiyonlar: *def foo(*args, **kwargs): return (args, kwargs)*
- Dekoratörler:

```
@dekoratorum('owtg2013')
def fonksiyon():
    return 42
```

# PYTHON

- Sınıflar:

```
class Foo(object):
    def __init__(self, bar):
        self.bar = bar
```

- Docstring yapısı: *sınıflara ve fonksiyonlara açıklama yazabilme.*
- Exceptionlar: *try..except..else..finally*
- Namespaceler:
  - ***import*** *logging*
  - ***from*** *datetime* ***import*** *tzinfo* ***as*** *taymzon*
- Introspection: *herhangi bir objenin hangi elemanlara sahip olduğunu görebilme*

# PYTHON

- Generator'ler: *yield anahtar kelimesi*
- List, Set, Dictionary Comprehension: *S = [x**2 for x in range(10)]*
- Bağlam Yöneticileri (Context Manager): *with ifadesi, __enter__ ve __exit__ metodları*
- Sınıf dekoratörleri
- Abstract Base Class'lar
- Meta Sınıflar (Metaclasses)
- Daha da fazlası için Stackoverflow Sorusu: *Python progression path - From apprentice to guru* http://stackoverflow.com/q/2573135/566715

# PYTHON - PEP8

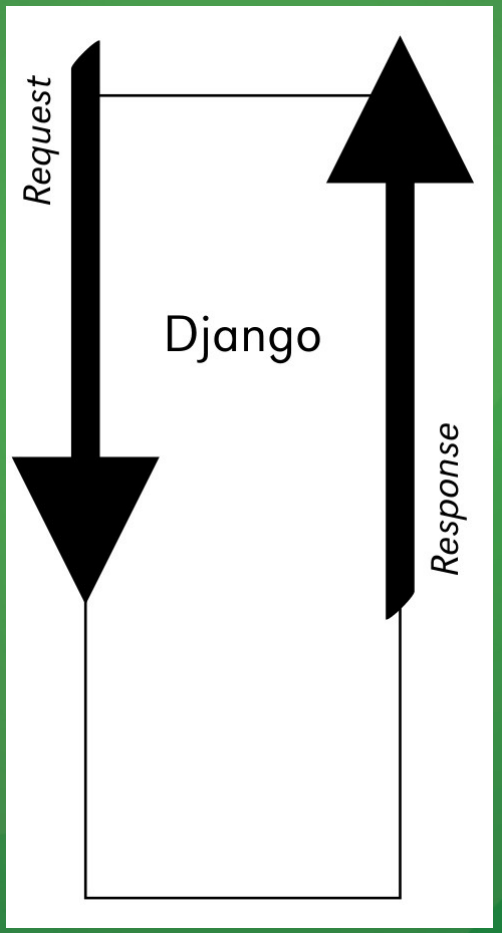
- Tab değil boşluk
- 4 boşluklu girintiler
- Tab ve boşlukları karıştırmamalı, birlikte kullanılmamalı!
- kucuk_harf_metodlar
- BuyukHarfSiniflar
- 80 karakteri geçmeme kuralı
- v.s. ..

# DJANGO - Kurulum

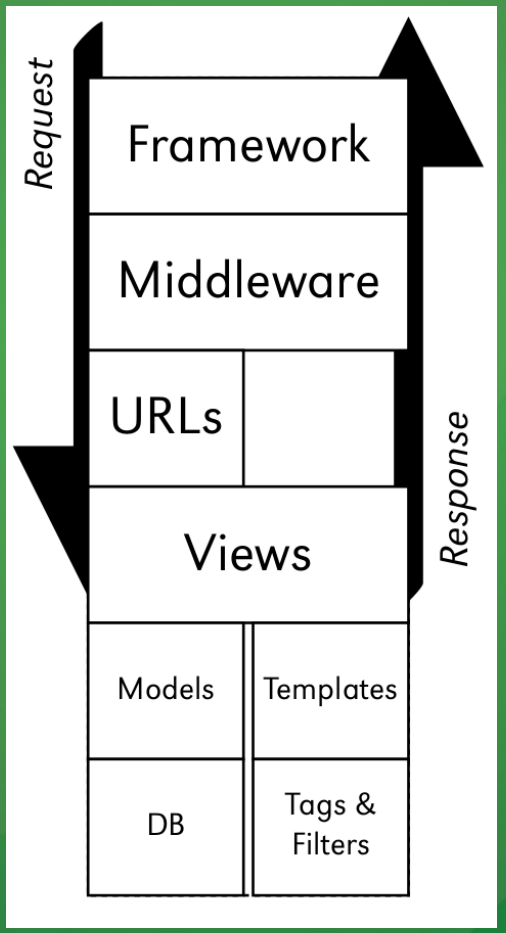
- Virtual Environment! *virtualenv, virtualenvwrapper*
  - İstediğiniz python paketlerini kurabildiğiniz, sisteminizde herhangi bir şeyi bozmadan kullanabileceğiniz bir sanal ortam sağlıyor
- Virtual Environment'ı kurduktan sonra:
  - *pip install django*

    ya da
  - *pip install django==1.5*
- requirements dosyası
  - *pip install -r requirements*
- Python 2? Python 3?

# Django'da İşler Nasıl Yürüyor?



# Django'da İşler Nasıl Yürüyor?



# Django - Proje

- Proje demek sizin websiteniz demektir.
- *django-admin.py startproject owtg*

owtg

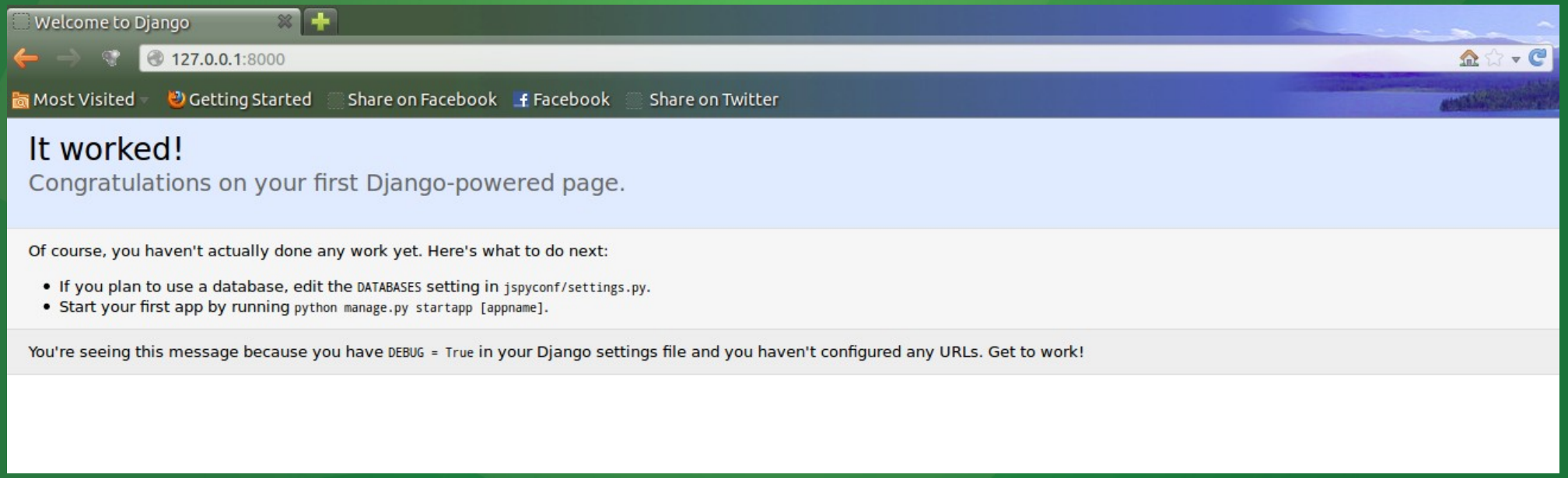
```
├── owtg
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py
```

# Django - Proje

- *__init__.py:* Boş bir dosya, dizinin python paketi olarak görülmesini sağlıyor.
- *manage.py:* Django ile iletişiminizi sağlayan komut satırı uygulaması
- *settings.py:* Ayarların durduğu dosya
- *urls.py:* URL tanımlarının yapıldığı dosya, bir nevi Django web sitenizin "İçindekiler" bölümü
- *wsgi.py:* WSGI uyumlu sunucularda sitenizi çalıştırmak istiyorsanız giriş noktanız.

# Django - Proje

- Proje dizinine girip: *python manage.py runserver* komutunu verdiğinizde, 127.0.0.1:8000'de:



# settings.py

- Temel ayarlarınızı belirttiğiniz dosya
- Genel yaklaşım:

```
try:
    # Import local settings to override existing values
    from settings_local import *
except ImportError:
    from settings_default import *
```

# settings.py

- *settings_local.py* 'de önce

  *from settings_default import **

  Sonra da istediğiniz ayarları verebilirsiniz. Böylelikle, settings_local.py'ı sürüm kontrolüne koymayıp geliştiricilerin kendi ayarlarını tutmalarını sağlayabilirsiniz.

- Herkesin kullandığı ayarlar *settings_default.py* içinde durur:
  - Veritabanı ayarları
  - Eposta ayarları
  - Statik dosyaların duracağı yerler
  - Kullanılan Uygulamalar
  - v.s. ..

# urls.py

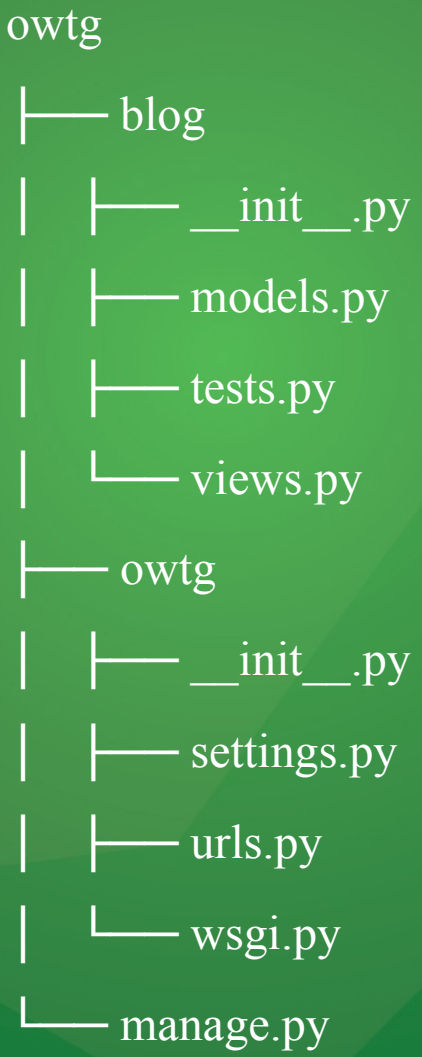
- Sitede gidilen adresin hangi view tarafından işleneceğini burada belirtiyoruz.
- Düzenli İfadeler! (Regular Expressions!)
- Farklı urls.py'ler tanımlayabilir, aşağıdaki gibi include edebilirsiniz.

```
from django.conf.urls import patterns, include, url

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

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', '{{ project_name }}.views.home', name='home'),
    # url(r'^{{ project_name }}/', include('{{ project_name }}.foo.urls')),

    # Uncomment the admin/doc line below to enable admin documentation:
    # url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

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

# Python Uygulamaları - Paketler

- Bir Django sitesi bir çok uygulamadan oluşur. Bunların bir kısmını siz yazarsınız, bir kısmı ise 3rd Party uygulamalardır.
- *python manage.py startapp blog*

# Python Uygulamaları - Paketler



# Python Uygulamaları - Paketler

```
owtg
├── blog
│   ├── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
...
```

- Uygulamanızın içinde bunların dışında:
  - *urls.py*
  - *forms.py*

  tanımlamak isteyebilirsiniz. Detayları sonraki slaytlarda.

# MTV is our MVC

- Model ----------> Model
- View ----------> Templates
- Controller ----------> Views

# Model: models.py

- Kısaca veritabanı yapınızı burada kuruyorsunuz. ORM sayesinde bu model yapısına uygun veritabanı tablolarını oluşturabiliyorsunuz.
- Örnek:

```
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)
```

- Veritabanında modellere uygun tabloları oluşturmak için:

  *python manage.py syncdb*

# Model: models.py

- OneToOne, ManyToMany ve one to many(ForeignKey) ilişkilerini tanımlayabiliyorsunuz.
- Meta alt sınıfı sayesinde admin tarafında hangi alana göre sıralama yapılacağını belirlemek gibi tanımlamalar yapabiliyorsunuz.
- __*unicode*__ metodu ile ilgili sınıfın bir instance'ının print edildiğinde ne döneceğini belirtebiliyorsunuz.

# View: views.py

- (en heyecanlı kısmı burası :) )
- urls.py'de hangi *“view”*a gidileceğinin tanımlandığını söylemiştik.
- Gelen request'e göre arka taraf (“backend”) işlerinin yapıldığı ve bir response döndürüldüğü yer bu dosyanın içindeki fonksiyonlar ve sınıflardır.
- Örnek:

```
def my_view(response):
    return render(request, “list.html”)
```

# Template: templates dizini

- Projenin içinde bir *templates* dizini yaratılıp, *settings_py* içinde *TEMPLATE_DIRS* değişkeni içerinde belirtilir. Birden fazla dizin de belirtebilirsiniz.

```
import os

PROJECT_DIR = os.path.abspath(os.path.dirname(__file__))

...

TEMPLATE_DIRS = (

    os.path.join(PROJECT_ROOT, "templates"),

)
```

- templates dizini altına Django'nun template desteği içeren html dosyalarınızı koyabilirsiniz.

# Template: list.html ve Django Template Dili

***list.html:***

```
<div class=“post”>
{% for post in posts.object_list %}
    <div class=“title”>{{ post.title }}</div>
    <p>{{ post.body | truncatewords:50 }}</p>
    <a href=“{% url post post.pk%}”>More</a>
    {% if user.is_staff or user.is_authenticated and user == post.author %}
    <a href="{% url edit post.pk %}">Edit</a>
    {% endif %}
</div>
```

# forms.py

- Tanımladığınız modellere giriş yapılmasını kolaylaştıran bir yapı. Ayrıca daha veritabanına herhangi bir şey yazmadan, gelen veri üzerinde kontroller yapabilirsiniz. Burada anahtar kelime: *Form Validation.*
- Hazır formları kullanabilir, ya da kendiniz yazabilirsiniz.

```
from django import forms

class MyForm(forms.Form):

    name = forms.CharField(max_length=30)

    email = forms.EmailField()
```

# forms.py

- **Formları bir view içerisinde kullanmak:**

```
from blog.forms import MyForm
def my_view(request):
    form = MyForm(request.POST or None)
    if request.method == "POST" and form.is_valid():
        name = form.cleaned_data['name']
        email = form.cleaned_data['email']
        # veriyle bir şeyler yapabilirsiniz, örn: modelin save() metodunu
        # çağırıp veritabanına kaydedebilirsiniz v.s.
    return render(request, 'myform.html', {'form': form})
```

# İşinize yarayacak Django uygulamaları

- ***django-extensions:*** Django içinde olmayan, ama yararlı bir çok manage.py eklentileri içermektedir. Hayatınızı kolaylaştıracaktır.

  https://github.com/django-extensions/django-extensions

- ***south:*** Django projeleri için veritabanı şeması ve veri göçü işlerini sağlayan bir uygulama.

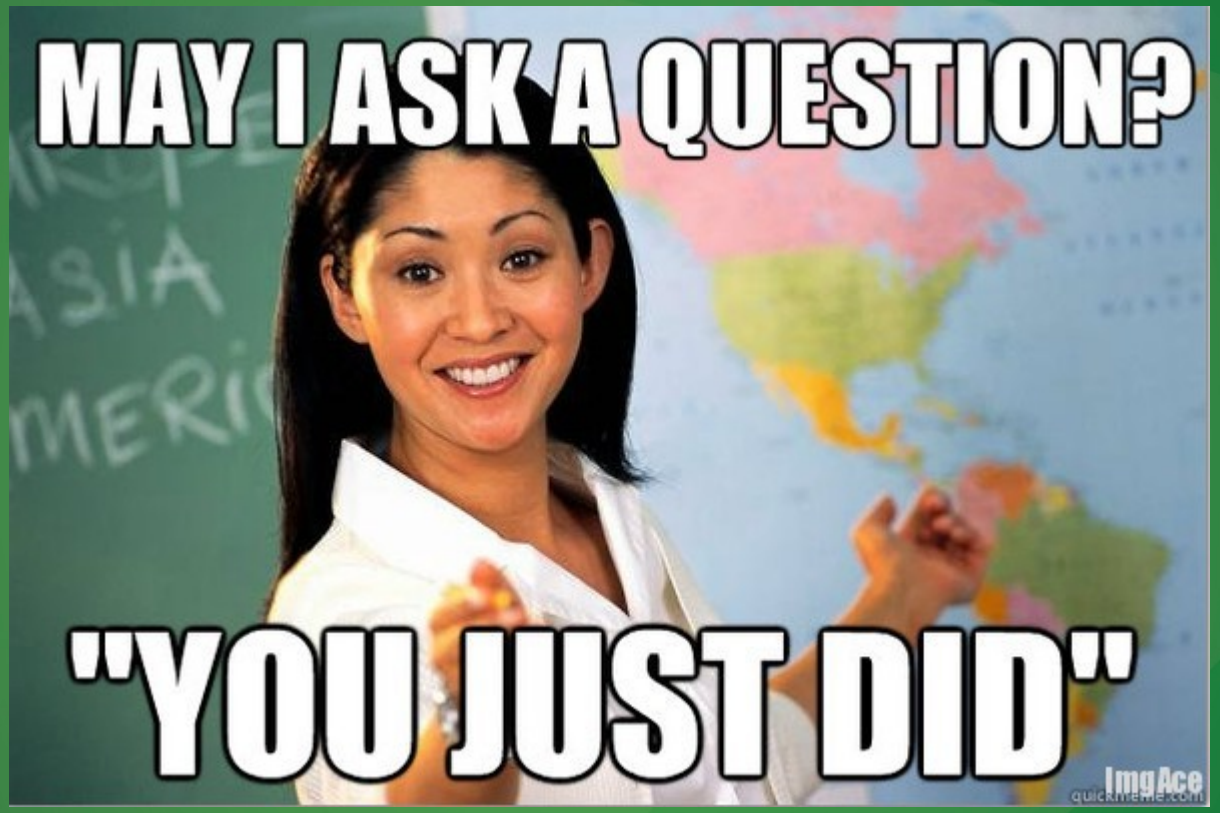
  http://south.aeracode.org/

- ***ipython ve ipdb:*** Tab ile tamamlama desteği, renkli kod gösterimi gibi özelliklere sahip python kabuğu ve hata ayıklama konsolu. Python konsolu ve pdb hata ayıklama aracını aramayacaksınız.

# Öneriler

- Django dökümanlarını bol bol gezin, tutorial'ı https://docs.djangoproject.com/en/1.5/intro/ mutlaka anlayarak bitirin!
- Kendiniz mutlaka, en az bir uygulama yazın.
- Topluluğun bir parçası olun, yeni çıkan, trendde olan Django uygulamalarını takip edin.
- Django-Users https://groups.google.com/group/django-users e-posta listesine üye olun, sorunuz olursa bu listeye ya da freenode üzerinde #django kanalına sorun.
- Python-Istanbul https://groups.google.com/group/python-istanbul grubuna üye olun, toplantılara katılın.
- Daniel Greenfield ve Audrey Roy'un yazdığı “Two Scoops of Django” kitabını https://django.2scoops.org/ edinin.

# SORULAR?




Teşekkürler.

# Kaynaklar

- Mike Crute, Mike Pirnat, David Stanek: Web Development with Python and Django - http://slidesha.re/16n7uiN
- Django başlangıç dökümantasyonu (Tutorial) : https://docs.djangoproject.com/en/dev/
- Two Scoops of Django: https://django.2scoops.org/