#pyconph

Django Workshop

PyCon PH 2016

Jon Danao Pythonista Wannabe
jondanao@gmail.com

Who is Jon Danao?

THE APP
FACTORY

http://theappfactory.io

Catholic Mass Media Awards

Boomerang Awards

Yahoo! Hack Day!

Spark Awards

Mobile Web Awards

Quill Awards

PANATA Awards

Anvil Awards

Mob-Ex Awards

Asia Pacific Tambuli Awards

Who are you?

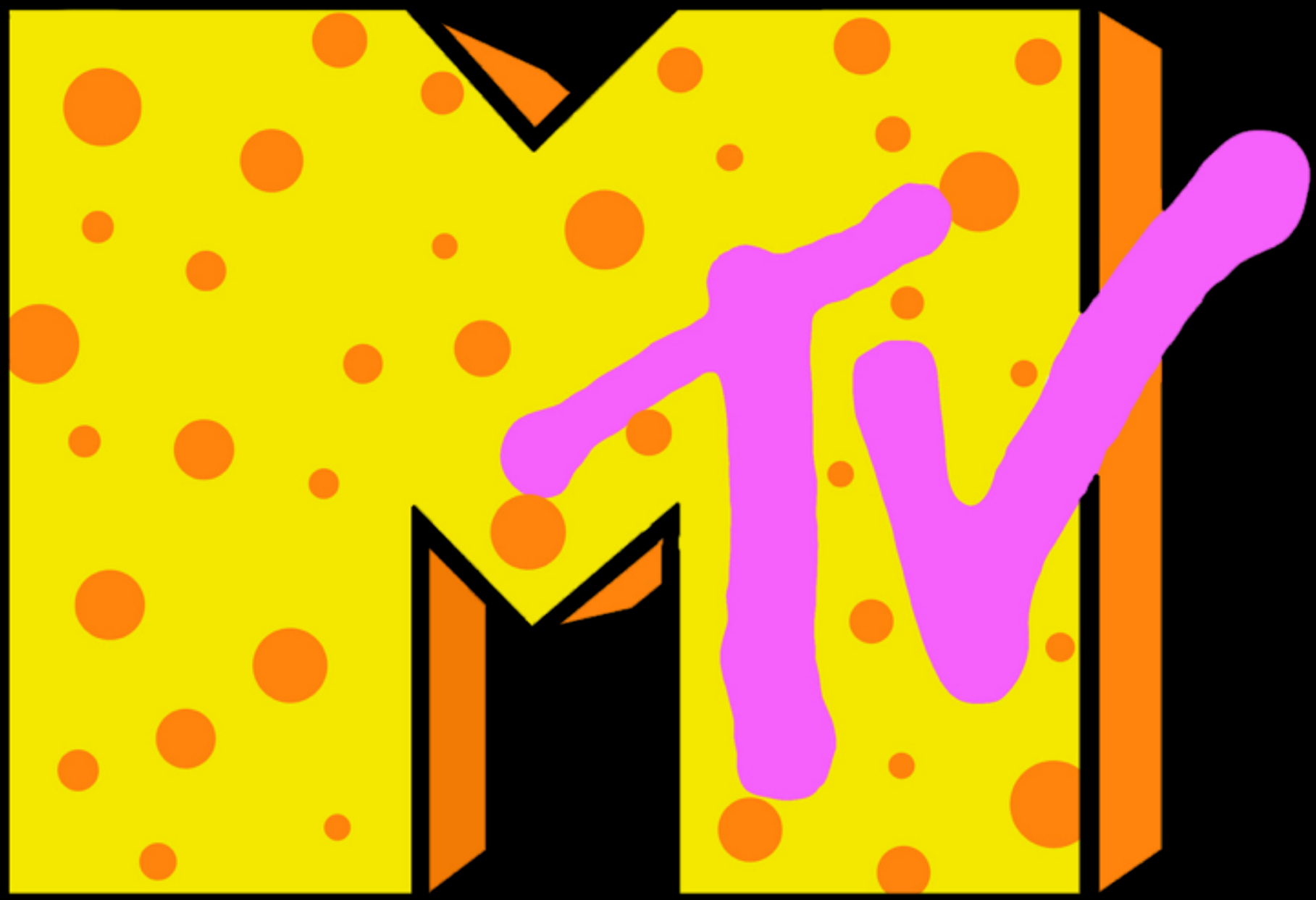
Topics

1. Setup: How to setup your machine for Django
2. Project: How to start your Django project
3. Models: How to define your data
4. URLs and Views: How to access your data
5. Templates: How to render your data

What is Django?

What is Django?

- Batteries-included Python web framework
- Similar to Rails, CodeIgniter, Play Framework, Meteor
- Rapid application development (RAD)
- Shared-nothing architecture
- Model-View-Controller (MVC)



Python Philosophies People

Philosophies

Loose coupling	Flexibility
Consistency	Predictability
KISS	Clarity
RAD	Agility
DRY	Reusability

Who uses Django?

- All Mozilla sites migrated from CakePHP to Django
- Firefox add-ons site:
  - 250k add-ons
  - 150M views/month
  - 500M API hits/day

Source: http://www.slideshare.net/andymckay/anatomy-of-a-large-django-site-7590098

DISQUS

- 700k websites
- 30M profiles
- 170M comments
- 500M unique visitors
- 45k requests/sec
- 8B page views/day

Source: http://blog.disqus.com/post/62187806135/scaling-django-to-8-billion-page-views

- 70M users
- 18M visitors/month
- 80M objects in S3
- 500M pins/day
- 2.5B page views/month
- 1.6B follower rows

Source: http://expandedramblings.com/index.php/pinterest-stats/
http://highscalability.com/blog/2013/4/15/scaling-pinterest-from-0-to-10s-of-billions-of-page-views-a.html

- 150M monthly active users
- 55M photo uploads/day
- 13k likes/second
- 1k comments/second
- 16B photos shared
- 35M selfies

Source: http://expandedramblings.com/index.php/important-instagram-stats/

DEAL yipit

Prezi

Discovery CHANNEL™

rdio™

Atlassian Bitbucket

NASA

openstack™ CLOUD SOFTWARE

NATIONAL GEOGRAPHIC

Learn Django the Easy Way

Concept

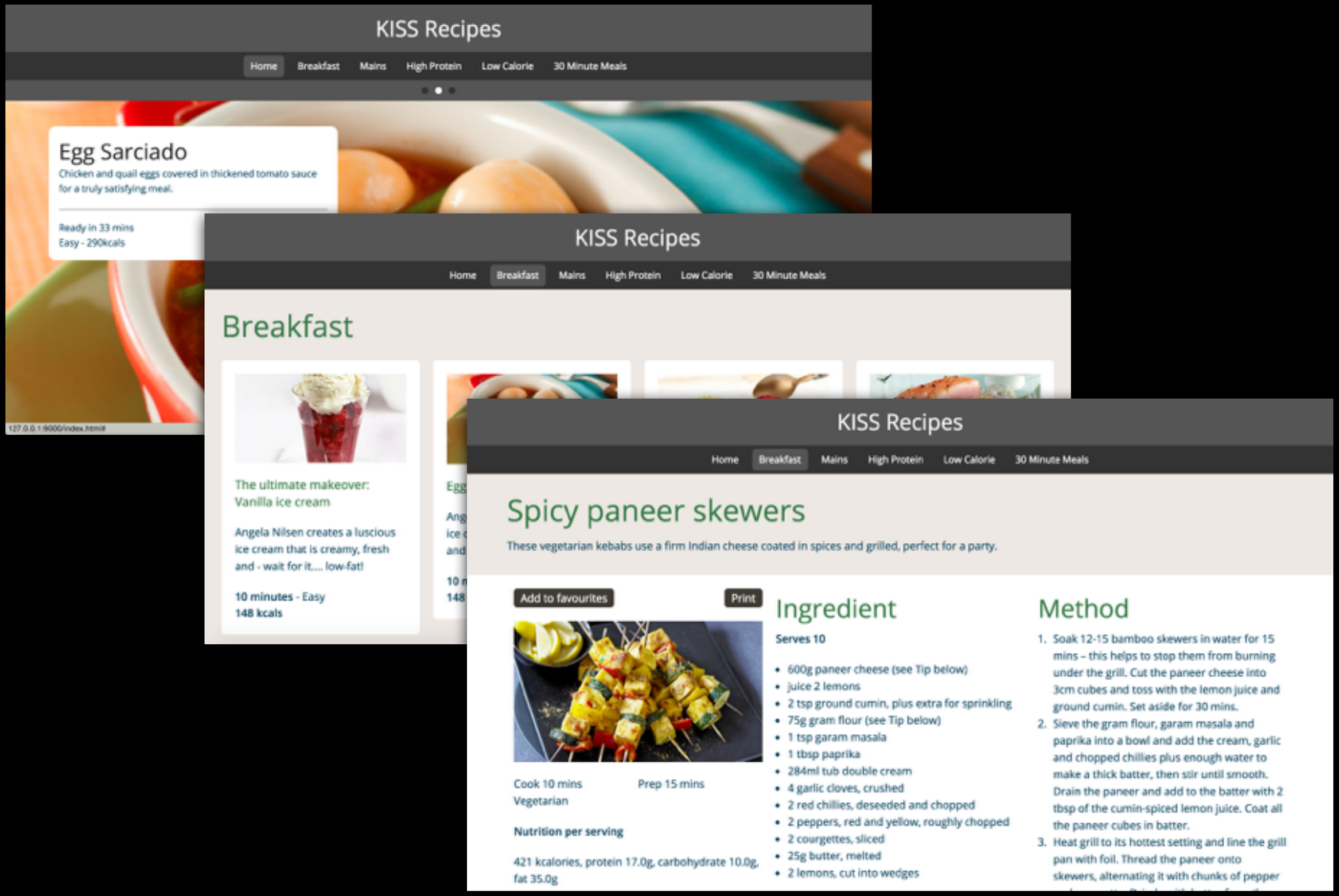
Summary

Walkthrough

Hands-on

The Project

KISS Recipes



http://localhost:8000/breakfast/6/spicy-paneer-skewers/

KISS Recipes

Home | Breakfast | Mains | High Protein | Low Calorie | 30 Minute Meals

Spicy paneer skewers

These vegetarian kebabs use a firm Indian cheese coated in spices and grilled, perfect for a party.

Add to favourites

Print

Cook 10 mins

Prep 15 mins

Vegetarian

Nutrition per serving

421 kcalories, protein 17.0g, carbohydrate 10.0g, fat 35.0g

Ingredient

Serves 10

- 600g paneer cheese (see Tip below)
- juice 2 lemons
- 2 tsp ground cumin, plus extra for sprinkling
- 75g gram flour (see Tip below)
- 1 tsp garam masala
- 1 tbsp paprika
- 284ml tub double cream
- 4 garlic cloves, crushed
- 2 red chillies, deseeded and chopped
- 2 peppers, red and yellow, roughly chopped
- 2 courgettes, sliced
- 25g butter, melted
- 2 lemons, cut into wedges

Method

1. Soak 12-15 bamboo skewers in water for 15 mins – this helps to stop them from burning under the grill. Cut the paneer cheese into 3cm cubes and toss with the lemon juice and ground cumin. Set aside for 30 mins.
2. Sieve the gram flour, garam masala and paprika into a bowl and add the cream, garlic and chopped chillies plus enough water to make a thick batter, then stir until smooth. Drain the paneer and add to the batter with 2 tbsp of the cumin-spiced lemon juice. Coat all the paneer cubes in batter.
3. Heat grill to its hottest setting and line the grill pan with foil. Thread the paneer onto skewers, alternating it with chunks of pepper

Topics

1. Setup: How to setup your machine for Django
2. Project: How to start your Django project
3. Models: How to define your data
4. URLs and Views: How to access your data
5. Templates: How to render your data

1. Setup

Pip

- Python CLI package management system
  - `$ pip install [package-name]`
  - `$ pip uninstall [package-name]`
  - `$ pip freeze > requirements.txt`
  - `$ pip install -r requirements.txt`

VirtualEnv

- Creates isolated Python environments
- Dependencies between projects don't mix
- Like having multiple XAMPP installations

Git

- Distributed Version Control System
- Fast, simple, non-linear, work offline
- Practice: http://www.atlassian.com/git/
- `$ git init`
- `$ git status`
- `$ git add`
- `$ git commit`

Exercise 1.x

Setup for OSX/Linux/Windows

Topics

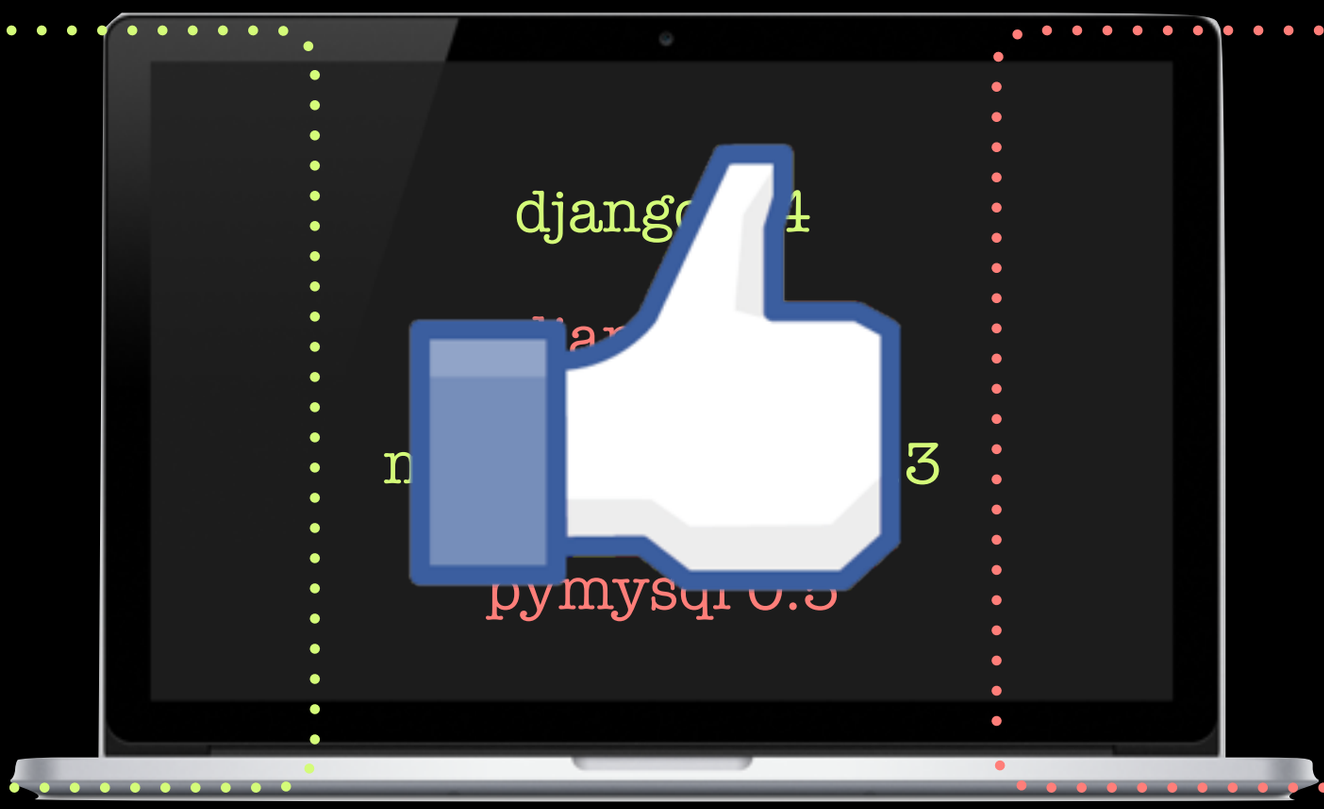
1. ✓ Setup: How to setup your machine for Django
2. Project: How to start your Django project
3. Models: How to define your data
4. URLs and Views: How to access your data
5. Templates: How to render your data

2. Project

VirtualEnv

python.ph

pycon.ph



(python-env)

(pycon-env)

Exercise 2.1

Setting Up the Development Environment

Project

1. In Django, a "project" is a collection of apps and settings
2. Can be created via command line
3. Auto-generates code - init, settings, urls, wsgi

Exercise 2.2

Creating the Django Project

Settings

- List of configurations and default values
- Where you define type of database to use
- Or the path to static files

Database

- No point of using Django if site is static!
- Support for PostgreSQL, SQLite3, MySQL, Oracle
- NoSQL is not fully supported

Migrations

- Way to safely make changes to the database schema
- Better replacement for syncdb
- Great for tracking versions of database schema
- `$ python manage.py makemigrations`
- `$ python manage.py migrate`

Exercise 2.3

Configuring the Site Settings

Summary

- Project - collection of apps
- Database - supports SQLite, MySQL, PostgreSQL and Oracle
- Migrations - make changes to database schema via the model

Topics

✓1. Setup: How to setup your machine for Django

✓2. Project: How to start your Django project

3. Models: How to define your data

4. URLs and Views: How to access your data

5. Templates: How to render your data

3. Models

App

- Piece of functionality - articles, comments, ratings

- A project can have multiple apps; An app can be in multiple projects

- Just a Python package

Write programs that do one thing and do it well.

Model

- Class that represents objects in the database
- Class : Table : : Property : Field
- Model + ORM eliminates writing of SQL
- Run migrations to make tables out of these models

Exercise 3.1

Creating the Recipes App and its Models

Admin Site

- Pseudo CMS for CRUD operations
- Automatic creation of interfaces for models
- Register the models you want to appear in admin

Writing CRUD interfaces is boring and repetitive.

Exercise 3.2

Activating Models in the Admin

Summary

- App - piece of functionality
- Model - a class representation of db table
- Admin Site - pseudo CMS

Topics

✓1. Setup: How to setup your machine for Django

✓2. Project: How to start your Django project

✓3. Models: How to define your data

4. URLs and Views: How to access your data

5. Templates: How to render your data

4. URLs and Views

URLConf / urls.py

- Routes a URL pattern to a view
- Uses regex for maximum flexibility
- Table of contents for site

```
website.com/tech/12345/apple-launches-new-iphone/
urlpatterns = ['',
    url(r'^nation/$', views.nation),
    url(r'^sports/$', views.sports),
    url(r'^tech/$', views.tech),
]
```

Views

- Again, views DO NOT render HTML!
- Takes a request and returns a response
- Queries data from model, asks template to render
- Determines WHAT data the user should see, not HOW data should be rendered

Exercise 4.1

Naming the URL for Index Page

Querying

- Retrieve all objects (aka SELECT *)

```
all_recipes = Recipe.objects.all()
```

- Retrieve objects with filters (aka WHERE clause)

```
ten_minute_recipes = Recipe.objects.filter(cooktime=10)
recipes_with_photos = Recipe.objects.exclude(photo='')
```

- Chaining filters (aka AND operator in WHERE)

```
hi_pro_low_carb = Recipe.objects
     .filter(protein__gte=36)
     .filter(carbs__lte=50)
```

Querying

- Limiting and offsetting results (aka Pagination)

```
recipes = Recipe.objects.all()[0:5]
recipes = Recipe.objects.all()[5:10]
```

- Sorting objects

```
recipes = Recipe.objects.all().order_by('title')
recipes = Recipe.objects.all().order_by('-title')
```

- Retrieving single object

```
recipe = Recipe.objects.get(id=1)
```

Dunder

- `__init__`, `__file__`, `__unicode__`
- Pythonic, common and idiomatic methods
- Used as field lookups in `filter()`, `exclude()`, `get()`
- Extremely powerful and intuitive

Field Lookups

- Basic model lookups

  `SYNTAX: field__lookuptype=value`

  - `Recipe.objects.filter(title__contains='Adobo')`
    `SELECT ... WHERE title LIKE '%Adobo%'`

  - `Recipe.objects.filter(title__startswith='Chicken')`
    `SELECT ... WHERE title LIKE 'Chicken%'`

  - `Recipe.objects.filter(calories__gte='200')`
    `SELECT ... WHERE calories >= '200'`

Field Lookups

- Model relationship lookups

```
SYNTAX: field__foreignfield__lookuptype=value
```

```
Recipe.objects.filter(category__slug__contains='fast')
```

```
SELECT * FROM recipe
   INNER JOIN category
   ON recipe.category_id = category.id
       WHERE category.slug LIKE ‘%fast%’
```

Exercise 4.2

Querying Data for Index Page

Summary

- URLConf - routes URL patterns to views
- View - takes requests and returns a response
- Dunder - used for filtering queries

Topics

✓1. Setup: How to setup your machine for Django

✓2. Project: How to start your Django project

✓3. Models: How to define your data

✓4. URLs and Views: How to access your data

5. Templates: How to render your data

5. Templates

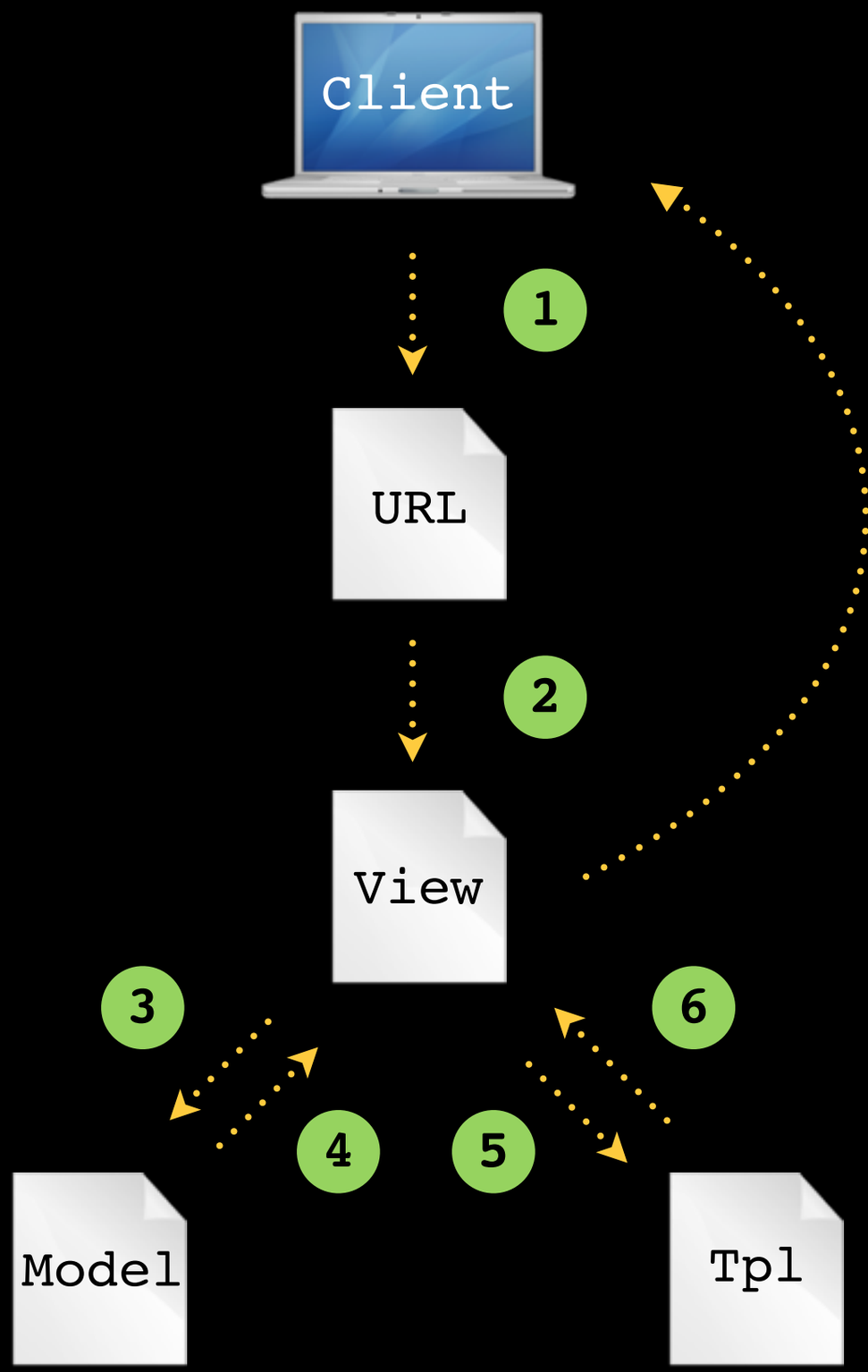
Template

- Renders HTML and other formats like XML, CSV and JSON
- Uses custom pseudo programming syntax
- Call function or do logic: `{% ... %}`
- Print variable: `{{ variable }}`
- Filters: `{{ variable|filter }}`

Exercise 5.1

Rendering the Index Page

Request-Response Cycle



1. Client requests a page via a URL
2. URLConf checks for URL patterns and assigns view to handle request
3. View requests data from model
4. Model queries the database and returns data to the view
5. View passes the data to template
6. Template renders data and returns HTML to the view
7. View returns the HTML response to the browser

Exercise 5.2

Rendering the Category and Recipe Pages

Summary

- Templates - render HTML and other formats
- `{{ var }}` - prints the value
- `{{ var|filter }}` - applies formatting to the variable before printing

Topics

✓1. Setup: How to setup your machine for Django

✓2. Project: How to start your Django project

✓3. Models: How to define your data

✓4. URLs and Views: How to access your data

✓5. Templates: How to render your data

That's it pancit!