

# Getting Started With Django — 1(Beginner series)



Ada Nduka Oyom · [Follow](#)

Published in Nur: The She Code Africa Blog

5 min read · Jul 12, 2017



New to Python? or already dancing some tango with it and you feel it's time to dabble into some framework magic for web?

## Meet Django:

DJANGO is a web framework built entirely on python, it's free, open-sourced and also follows the **Model View Controller** pattern, (in this case; **Model**, **Template**, **View** — Where the *View* relates to the *Controller* and The *Template* relates to the *View* in the MVC pattern. This part can be a bit confusing to newbies starting out, but along the line you'll get to understand the logic behind this more :). There are already lots of popular sites running on django, examples are:



popular social apps that make use of django

Django also provides some excellent documentation [here](#), along with features and tools, some of which include:

1. A nice templating language.
2. Security features like CSRF
3. Excellent lightweight server for development and testing e.t.c

In this tutorial, i will be showing you how to get your first Django website up and running. Before we start, we need to have Python downloaded and installed on our System, to download and install python, click [here](#) .

**Note:** *You need to already have a basic understanding of python, also I'll be running this tutorial on a Linux based system, so most commands would follow suite the linux way. But there will be little or no difference on most.*

To ensure it's fully downloaded, open up your terminal and type in

```
python
```

An interactive shell shows up:

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To exit, type in `ctrl + z`.

To have a more neater arrangement, it's always advisable to create a directory for your projects

```
mkdir folder_name
```

Then `cd` into the project with: `cd folder_name` (every other step will be carried out while inside this folder)

### Set Up Your Virtual Environment:

Next thing we need to do, is set up our virtual environment, a virtual environment helps you run several versions of python/django right on your machine (*e.g you could have two different python/django projects running on different versions, to avoid them clashing and to give you room to run them both without errors, the virtual environment comes to your rescue. One virtual environment = one python/django version*). It's strongly advised to always use a virtual environment.

To set up our virtual environment, we'll be using python's package manager `pip` to do the installation, type in:

```
pip install virtualenv
```

After installation, it's time to create a virtual environment that would enable us use a preferred django version of our choice:

```
virtualenv env_name
```

Note: `env_name` should be replaced with the preferred name of your environment. (I like to name my environments with the django version installed in it for easier recognition).

### Activating Virtual Environment:

To activate our virtual environment for linux/Mac OS:

```
source env_name/bin/activate
```

For windows:

```
env_name/script  
activate
```

## Install Django:

Now it's time to install django on to our machine:

```
pip install django==1.8
```

Using `==1.8` only gives a direction to django about the particular version you want to install, in this case version 1.8. To just go ahead and download the latest version, input

```
pip install django .
```

## Starting A project:

Now we have django up and running, it's time to start up our first project! Yaah!. Still in our command line, type in :

```
django-admin.py startproject project_name
```

**Note:** `project_name` = name of your project . In this case, we'll work with `mask_off` as our project name.

This creates a sub-folder with the name `mask_off` and a skeleton structure of

```
mask_off  
├─mask_off  
|   ├─ __init__.py  
|   ├─ settings.py  
|   ├─ urls.py  
|   └─ wsgi.py  
└─ manage.py
```

Open in app ↗

Sign up

Sign in



1) the `__init__.py` helps python treat the directories as containing packages; so as to prevent directories with a common name, from unintentionally hiding valid modules that occur later (deeper) on the module search path. In most cases, it's usually an empty file.

2) The `settings.py` file contains all settings your project requires, as we progress, we'll visit this file often.

3) The WSGI (Web Server Gateway Interface) acts as the interface our web server uses to interact with our web application. Read more about it [here](#).

### Run Server:

There's no fun thing as that of visiting your own webpage, so let's run our server which also generate a link for us to view our webpage

```
python manage.py runserver
```

This shows up...

```
python manage.py runserver
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
You have unapplied migrations; your app may not work properly until they
are applied.
```

```
Run 'python manage.py migrate' to apply them.
```

```
July 12, 2017 - 15:19:01
```

```
Django version 1.8, using settings 'mask_off.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```

Notice the warning message about having unapplied migrations? Now let's do a small but very important talk about migrations;

### Making Migrations:

Migrations helps us make changes to our database schema without losing any data, each time we create a new model or make changes to a current one and run migrations, it helps update our database tables with the schemas without having to go through all the stress of dragging and recreating the database ourselves.

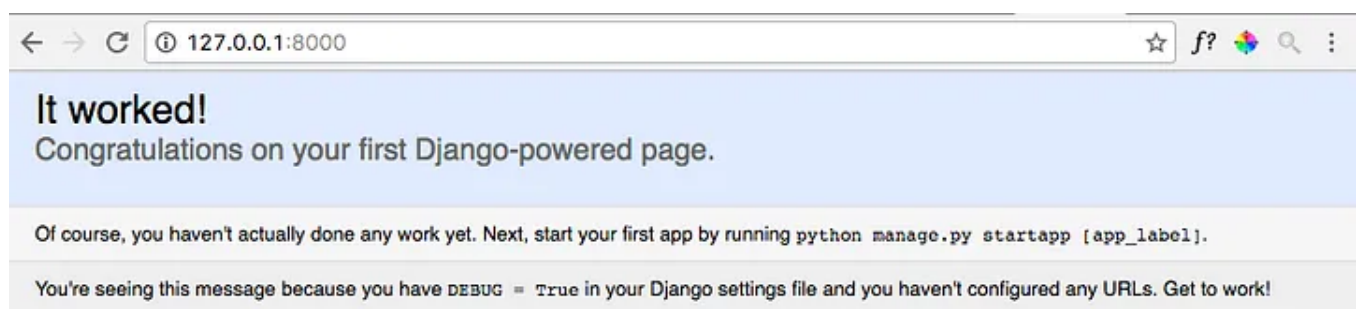
To make our migration:

```
python manage.py migrate
```

An output of this sort should show up:

```
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: admin, contenttypes, auth, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying sessions.0001_initial... OK
```

This implies a successful migration, Now we can successfully run our server with no issues `python manage.py runserver` . We get a success message on our webpage like the one below



Yaah! our very own webpage.

Now we have our server running and a 'webpage' , but django takes pleasure in reminding us we still have a lot of work to do before we can proudly call this a webpage.

The second part of this tutorial would contain; Creating a new app in our project, working with Urls, Templates, Creating views and Linking pages. For now, take out more time to go through the steps again and again to become more familiar with them. Only practice makes perfect!

Things learnt From this Tutorial:

- Setting up a virtual environment(Downloading, Activiating Virtual env)
- Installing Django
- Creating a project
- Basic project components
- Migrations
- Running Server

Encountered any issue along the line? Let me know! I'll be glad to help

Python

Django

Django Girls

Beginner

Women In Tech



Follow

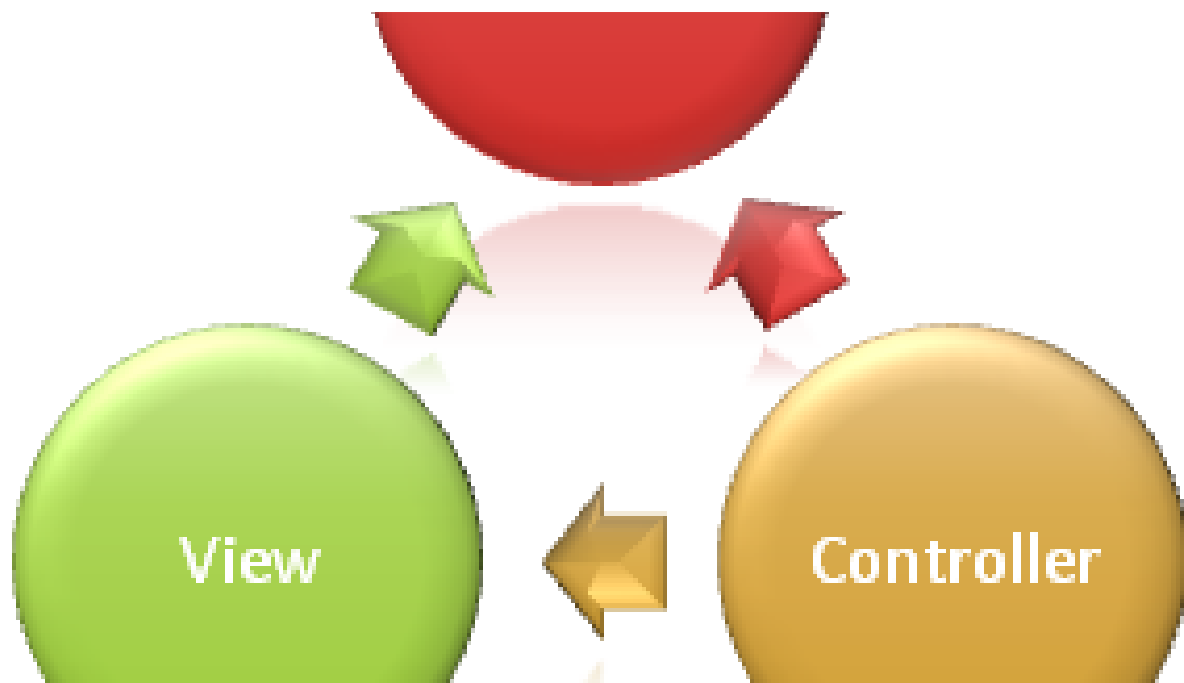
## Written by Ada Nduka Oyom


1.3K Followers · Editor for Nur: The She Code Africa Blog

Software Developer | Developer Relations and community Expert | DEI Consultant

---

More from Ada Nduka Oyom and Nur: The She Code Africa Blog



 Ada Nduka Oyom in Nur: The She Code Africa Blog

## Understanding the MVC pattern in Django


In my previous post, i put up a tutorial for beginners on how to get started with Django, here. And in between i talked about how Django...

Jul 19, 2017  1K  13





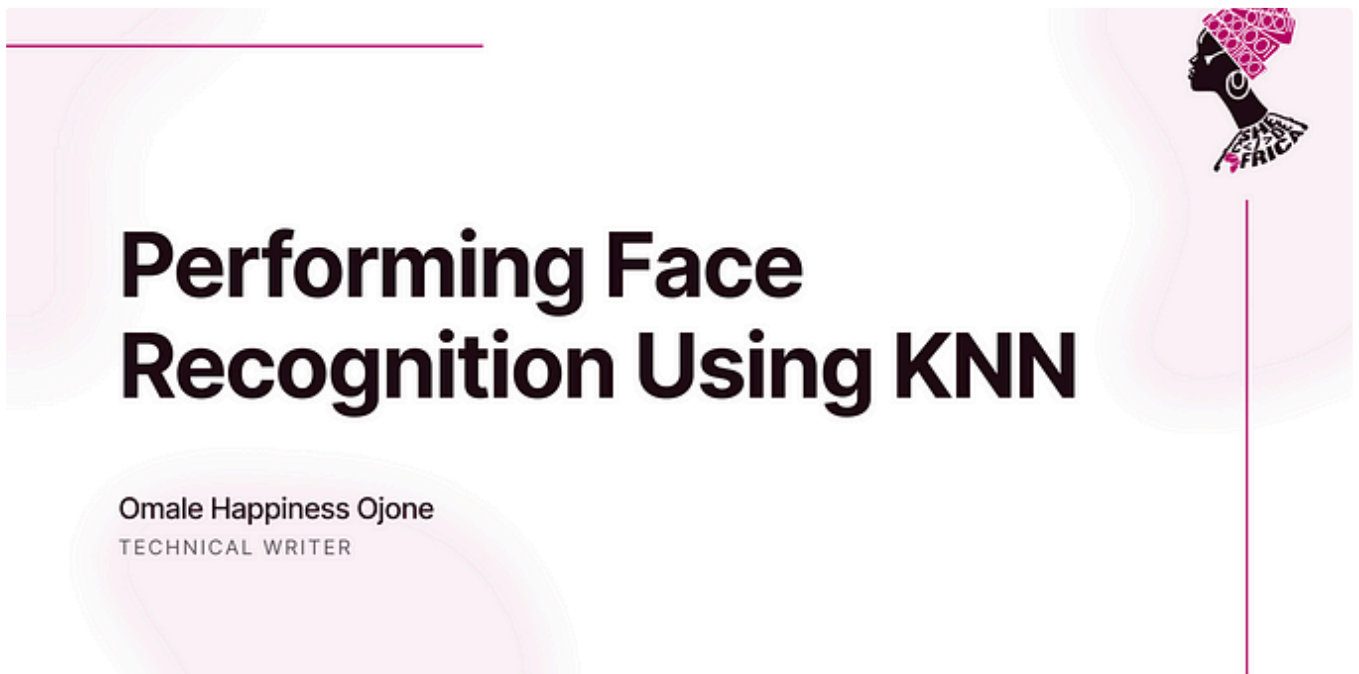



 Olamide 'Pearl' Makinde in Nur: The She Code Africa Blog

## A Guide to Technical Writing

A Guide To Technical Writing: Dos & Don'ts

Apr 9, 2021  706  5



 Omale Happiness in Nur: The She Code Africa Blog

## Performing Face Recognition using KNN


May 8, 2023  209





# CAREER FIELDS

## IN DEVELOPER RELATIONS

 Ada Nduka Oyom

### Career Fields in Developer Relations

Recently I broke down what Developer relations was about via my new page (@DevRelLite—focused on helping newbies interested in Developer...

Jun 24, 2021  142



See all from Ada Nduka Oyom

See all from Nur: The She Code Africa Blog

Recommended from Medium

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

---

## Projects

---

### NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

### HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

## The resume that got a software engineer a \$300,000 job at Google.

1-page. Well-formatted.



Jun 1

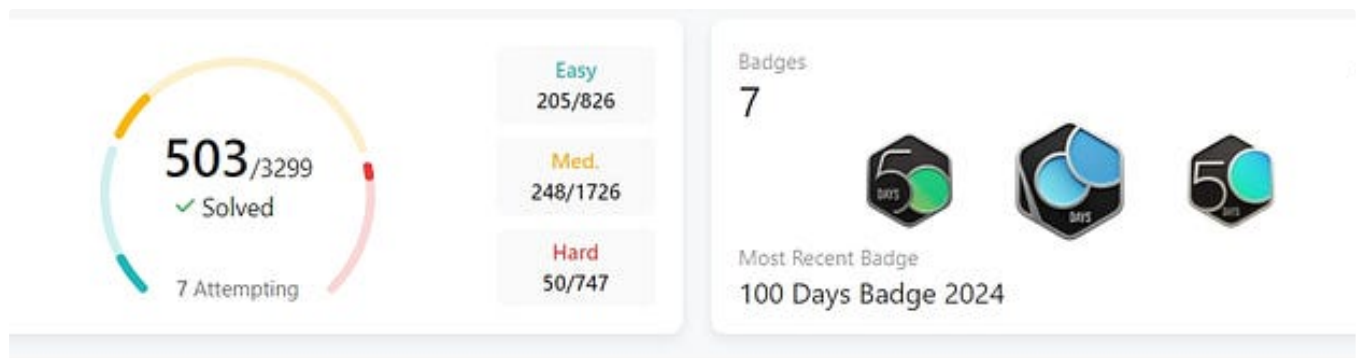


25K



502





Surabhi Gupta in Code Like A Girl

## Why 500 LeetCode Problems Changed My Life

How I Prepared for DSA and Secured a Role at Microsoft

★ Sep 26 🤝 3.3K 💬 69

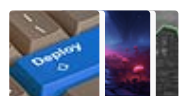


### Lists



#### Coding & Development

11 stories · 895 saves



#### Predictive Modeling w/ Python

20 stories · 1659 saves



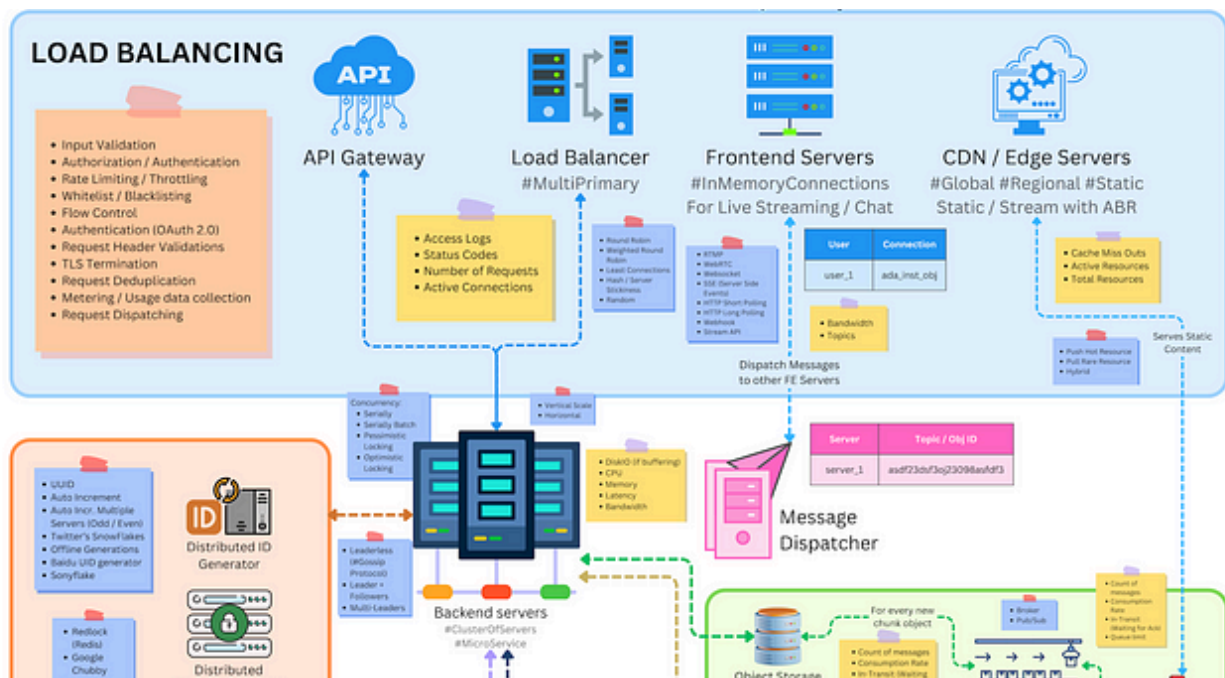
#### Practical Guides to Machine Learning

10 stories · 2017 saves



#### ChatGPT

21 stories · 864 saves



 Love Sharma in ByteByteGo System Design Alliance

## System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the...

★ Sep 17, 2023 🖱 8.8K 💬 63







Axel Casas, PhD Candidate in Python in Plain English

## Stop Doing Tutorials. Learn Programming Like This

Learn programming faster and better



Jul 13



4.3K



67



# django REST framework

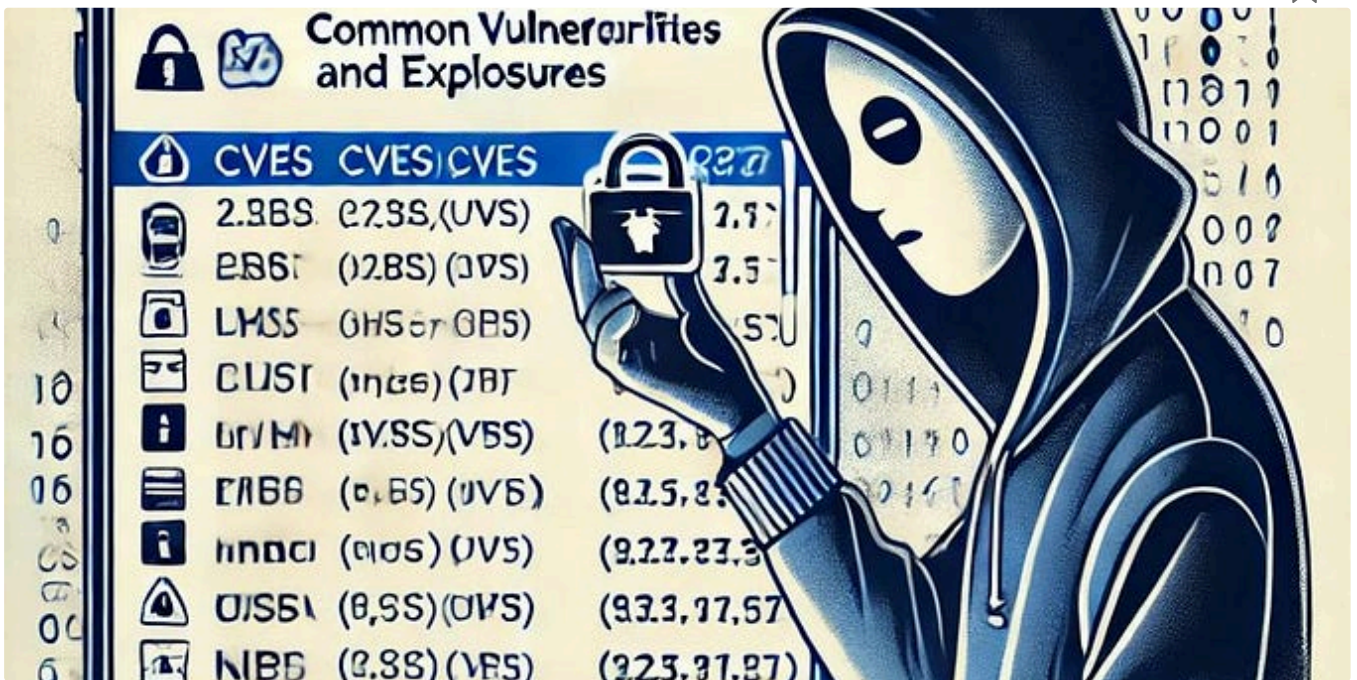


Dhruv Ahuja

## Stop Sending Messy Responses! The Ultimate Django DRF Response Structure You Need 🚀



Heads Up! Click here to unlock this article for free if you're not a Medium member!



Jonathan Mondaut

## How ChatGPT Turned Me into a Hacker

Discover how ChatGPT helped me become a hacker, from gathering resources to tackling CTF challenges, all with the power of AI.

See more recommendations