*Quick Reference Guide to*

# DJANGO TERMINAL COMMANDS

GET YOUR DJANGO PROJECT UP AND RUNNING QUICKLY

CODEMY.COM

Hiking the Valley of Fire Outside Las Vegas:

Welcome to my Django Terminal Commands Quick Reference Guide!

In this guide I'll show you some of the most common Django Terminal Commands that you'll need to work with Django. Sure, you probably already know all these commands, but it's really nice to have them all in one place where you can grab them at a quick glance if you need them!

This is not meant to be a comprehensive guide, but rather a quick glance sheet to get you up and running with commands quickly as you need them.  I'll also talk about some of the regular things you need to do every time you create a new Django Project.

Enjoy!

*-John Elder*
Codemy.com

## TABLE OF CONTENTS

*Section One*

## QUICK REFERENCE GUIDE
## DJANGO COMMANDS

# QUICK REFERENCE GUIDE TO DJANGO COMMANDS

*This is not meant to be a comprehensive list, just some of the most common Django commands...*

**INSTALL VIRTUALENV:** pip install virtualenv

**START VIRTUALENV POWERSHELL:** virtualev .
(first be sure to Set-ExecutionPolicy Unrestricted in powershell)

**START VIRTUALENV GITBASH:** python -m venv nameofVirtualenv

**ACTIVATE VIRTUALENV POWERSHELL:** ./Scripts/activate
(sometimes ../Scripts/activate)

**ACTIVATE VIRTUALENV GITBASH:** source virtualenvDIR/Scripts/activate

**ACTIVATE VIRTUALENV MAC:** source bin/activate

**DEACTIVATE VIRTUALENV:** deactivate

**INSTALL DJANGO INSIDE VIRTUALENV:** pip install django
(pip install django==2.04 to install a specific version of django)

**START A NEW DJANGO PROJECT:** django-admin.py startproject projectname

**RUN THE SERVER:** python manage.py runserver

**CREATE DATABASE MIGRATION:** python manage.py makemigrations

**MIGRATE THE DATABASE:** python manage.py migrate

**CREATE ADMIN USER:** python manage.py createsuperuser

**CREATE ADMIN USER GIT BASH:** winpty python manage.py createsuperuser

**CREATE NEW APP IN PROJECT:** python manage.py startapp APPNAME

*Section Two*

## STARTING NEW PROJECT THINGS TO DO

# STARTING A NEW PROJECT: THINGS TO DO

*Starting a new project? There's always a few things to do right off the bat. Let's talk about them!*

So you're creating a new Django project. You have your virtual environment set up and initiated, you've installed Django and started a new project, and added an app to it. Now what?

## SETTINGS.PY

The first thing to do is add your newly created app (let's call it "posts") to your settings.py file.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'posts',
]
```

Notice the 'posts', there at the bottom. That's what we want to add. Save/exit.

## ORIGINAL URLS.PY

When you start your project, a URLS.PY file will be generated. But it needs to be modified and you need to create a second URLS.PY file in the new app you generated.

Let’s take a look at the original URL.PY file and how you need to modify it…

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
   path('admin/', admin.site.urls),
   path('', include(‘posts.urls')),
]
```

What we’ve added is the “include” module to the second line, as well as the new path(“”, include(‘posts.urls’)) line.

That line points to the new URLS.py file we will create next in the posts directory (assuming the new ‘app’ you created is called posts. If you called it something else, modify that line with whatever you called it.

## NEW URLS.PY

After you've modified your original URLS.PY file to point to your new URLS.py file, you need to actually create the new URLS.py file! Add that to the directory of your new app. We're calling our new app "posts" so you'll add it to the posts directory.

Once you create the file, add this code to it:

```
from django.urls import path
from . import views

urlpatterns = [
	path('', views.home, name="home"),
	path('about.html', views.about, name="about"),
	path('add_post.html', views.add_post, name="add_post"),
	path('delete/<post_id>', views.delete, name="delete"),
	path('delete_post.html', views.delete_post, name="delete_post"),
]
```

I've added some dummy urlpatterns in the code just to give you an idea of the type of things you'll eventually add there.

It's important to import your views (line 2 of the code) because you need them to create pages for your website.

The rest is boilerplate Django…

## CREATE A TEMPLATES DIRECTORY

The next thing you'll need to create is a templates directory.

The templates directory is where you'll keep your base.html file and all of the .html webpage files that go with your project.

We're talking:

> base.html
> index.html
> home.html
> about.html
> whatever.html
> and on and on…

The templates directory should go in the posts directory (or whatever you named your app).

A common error is to place the templates directory in your main project directory. Don't do that! It goes in your app directory.

# CREATE A STATIC DIRECTORY

The next thing you'll need to create is a Static directory.

The Static directory is where you'll keep your images, css, and any javascript you might need.

Create the static directory in the main directory of your project (not in the app directory where your views.py file is for instance). Inside of that directory, create an images directory, a css directory, and a javascript directory.

In each of those directories place your images, css, and javascript respectively.

## MODIFY YOUR SETTINGS.PY FILE

In your settings.py file, at the bottom of the file, under the line
STATIC_URL = '/static/'

Add this bit of code…

```
STATICFILES_DIRS = [
	os.path.join(BASE_DIR, 'static'),
]
```

Finally, to access things in your static directory, for instance an image called me.png in the images directory in the static directory, place this code on your web page under the {% extends 'base.html %} line:

```
{% load static %}
```

And then to place the actual thing on the page (ie our me.png image):

```
<img src="{% static 'images/me.png' %}">
```

## ADD A DATABASE MODEL

It may be a little early to start talking about database stuff. But if you're planning on using a database in your project, you need to create a model class in your models.py file inside of your app directory.

Since this isn't really an in depth book on Django, I'm not going to go into this, well…in depth.  I'll just give you a very basic  model with one field.

**MODELS.PY**

```
from django.db import models

class Post(models.Model):
	title = models.CharField(max_length=200)

	def __str__(self):
		return self.ticker
```

As you can see, this just creates a basic class called Post, that has one field called title.

To  migrate this model and shove it into the database for use, issue these two commands from the terminal:

```
python manage.py makemigrations
python manage.py migrate
```

The first command creates the migration, the second one pushes it (migrates) into the database.

## ADD YOUR MODEL TO ADMIN SECTION

Once you've created a model class in your models.py file, created a migration, and then pushed the migration into the database…you can add your model to the **ADMIN** section of the site (localhost:8000/admin).

To do this you need to register you model in the admin.py file, located in the app directory of your project.

**ADMIN.PY**

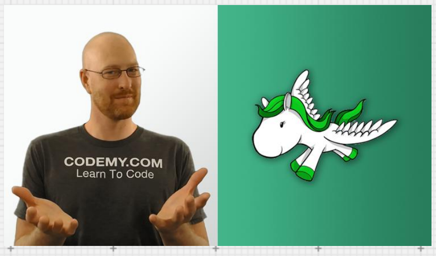
```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

That's pretty much all there is to it. Notice the Post in the second line and the Post in the third line reference the name of the class we just created in the models.py file. If you called your model "Stocks" for instance, you would replace the word Post With Stocks above.

Once you add those three lines of code to your ADMIN.PY file, your database model will appear in the Admin section of your site, located at localhost:8000/admin

*Section Three*

## ONLINE TUTORIALS
## WHERE TO LEARN MORE



## Intro To Django For Web Development

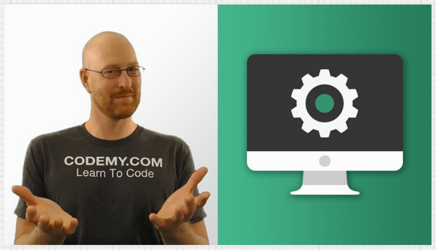
Learn the absolute basics of Django from the ground up in this course. Perfect for beginners.

**Take Course:** Udemy | Codemy

## Django & Python: To-Do List App

We'll build a cool To-Do List app! Learn how to use Databases with Django

**Take Course:** Udemy | Codemy



## Build A Stock Market App With Django

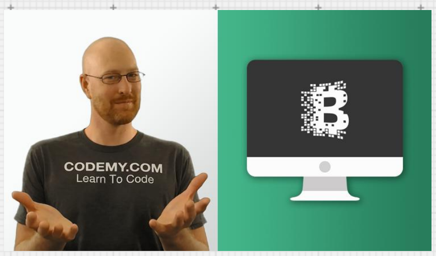
We'll build an even cooler Stock Market App to get real time data and databases

**Take Course:** Udemy | Codemy

## Build a User Authentication App

Allow people to sign up for your web app, log in, log out, edit their profiles, and more…

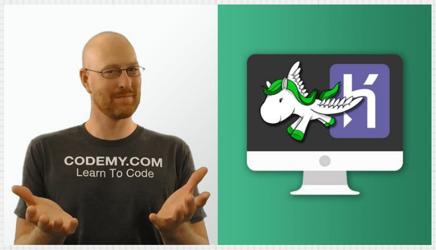
**Take Course:** Udemy | Codemy



## Crypto Currency News With Django

Build a Crypto currency News App! Learn how to connect to a 3rd party Crypto API and more

**Take Course:** Udemy | Codemy



## Push Django To Heroku for Webhosting

Learn to host your Django apps on Heroku for professional webhosting.

Take Course: Udemy | Codemy

## Intro To Django with Python For Web Development

In this course you'll learn how to build simple websites with Django and Python!

Django overwhelms a lot of people, and it doesn't have to! If you understand just a few basic concepts, you'll see that Django is a breeze to use!

In this course I'll be developing on a Windows machine, but you should be able to follow along if you're on a Mac or Linux. I'll show you how to download and install Python and Django, and create a basic website.

We won't be doing ANY database work in this course. I feel like databases confuse a lot of newbies, so instead of getting bogged down in all of that, we're just going to skip it and focus on building basic static websites.

You'll be able to build simple personal websites, or simple business websites when you're finished with this course.

**TAKE COURSE:** **UDEMY** | **CODEMY**

## Django & Python: To-Do List App

In this course I'll walk you through it step by step and you'll be building your first web app in MINUTES. You'll be amazed how quick and easy it is to create very professional looking websites, even if you have no programming or web design experience at all.

Watch over my shoulder as I build a cool To-Do List app step by step right in front of you. You'll follow along and build your own copy. By the time we're finished, you'll have a solid understanding of Django and how to use it to build awesome web apps.

The course contains 28 videos – and is just over 2 hours long. Watch the videos at your own pace, and post questions along the way if you get stuck. You don't need any special knowledge or software to take this course, though any experience with HTML or CSS is a plus. You don't even need to know the Python programming language. I'll walk you through EVERYTHING.

Django is a great web development tool and learning it has never been this easy.

**TAKE COURSE:** **UDEMY** | **CODEMY**

## Build a Stock Market Web App With Python and Django

Watch over my shoulder as I build a cool Stock Market app step by step right in front of you. You'll follow along and build your own copy. By the time we're finished, you'll have a solid understanding of Django and how to use it to build awesome web apps.

The course contains 39 videos – and is just over 2 hours long. Watch the videos at your own pace, and post questions along the way if you get stuck.

You don't need any special knowledge or software to take this course, though any experience with HTML or CSS is a plus. You don't even need to know the Python programming language. I'll walk you through EVERYTHING.

We'll connect to a third party API to get our stock market data. Once you learn to do this, you can use literally any API online in your Django Apps!

There's lots of moving parts to this app, and it's a lot of fun!

**TAKE COURSE:** UDEMY | CODEMY

## User Authenticate With Python And Django

We'll build a cool User Authorization app that let's users sign up (register) for your site, log in, log out, edit their profile, and change their passwords. These days just about every website let's people sign up and log in, and you really need this skill if you want to build modern web apps.

We'll style the website using the popular Bootstrap CSS framework (I'll show you how to use it!)

The course contains 30 videos – and is just over 2 hours long. Watch the videos at your own pace, and post questions along the way if you get stuck. You don't need any special knowledge or software to take this course, though any experience with HTML or CSS is a plus. You don't even need to know the Python programming language. I'll walk you through EVERYTHING.

This is one of my best-selling Django courses because it's such an important skill to learn, and it's surprisingly easy to pick it up!

**TAKE COURSE:** **UDEMY** | **CODEMY**

## Crypto Currency News With Python & Django

Crypto currencies are all the rage right now. Wouldn't it be cool to build a website that shows Crypto news automatically? That's what we'll learn in this course!

We'll build a website using Django and Python and Bootstrap that connects to a free third party crypto API.

We'll be able to pull news stories, crypto price data, and all kinds of cool stuff, and output it onto the screen of our website automatically.

Who Should Take This Course?

This course is aimed at the beginner. You don't need to know Python, or Django, or Bootstrap...or anything at all...to take this course. I'll walk you through it all; step by step.  If you already know the basics of any of those things, you'll be fine too. You'll still learn some cool things along the way!

**TAKE COURSE:** UDEMY | CODEMY

## Push Django Apps To Heroku for Web Hosting

So you know how to build Django apps with Python, but aren't sure how to get your app up onto the Internet?  This course if for you!

With many other programming languages and web frameworks, pushing your code up to the Internet for web hosting is relatively easy.

With Django it's a bit harder.  Well, harder may not be the right word…complicated may be a better way to put it!  It's actually fairly easy to push your Django code online to a web host, but you have to tinker with quite a few settings and do a few things to your app that aren't necessarily intuitive.

In this course I'll show you how to push your code up to Heroku, the massively popular web hosting service.  Heroku has a free tier that we'll be using and is great for learning how to do all this stuff.

See you inside!

**TAKE COURSE:** **UDEMY** | **CODEMY**

## Thanks For Reading!

I hope you enjoyed this short ebook on Django Commands and setting up new Django projects!

If you'd like to learn more about Django, I'd love to see you in one of my courses either at Udemy or on my own website, Codemy.com  Check out a special discount offer for ALL my courses on the next page.

*-John Elder*
Codemy.com

"

*As we look ahead into the next century, leaders will be those who empower others.*

**-Bill Gates**