

# Django Tutorial for Beginners

This is the first post in the Django tutorials series. Today we'll be learning about the Django environment and will set up our first Django web application on a localhost. This Django tutorial is for beginners who don't have any idea about Django framework. Let's get set and GO!.

## Table of Contents[\[hide\]](#)

- [1 Django](#)
  - [1.1 Why is Django so popular?](#)
- [2 Django Hello World Project](#)
  - [2.1 Installing Python](#)
  - [2.2 Installing Virtual Environment](#)
  - [2.3 Installing Django](#)
  - [2.4 Create New Django Project](#)
  - [2.5 Creating a Django Web Application](#)
- [3 URL Mapping](#)
  - [3.1 URL Routing](#)

## Django

Django is a high-level web framework that's written in Python. It's free and open sourced and is used to develop web applications easily.

### Why is Django so popular?

Django takes care of the common stuff of web development so that you can concentrate on writing the stuff that's unique to your application. The framework provides you with pre-defined modules and tools so that you don't reinvent the wheel every time.

Few features that make Django stand high in the web application development:

- **Authentication:** The Django system handles both authentication and authorization.
- **URL Routing:** Django lets you design easy URL routing.
- **Security:** Django takes care of the security of your web application. It enables protection against many vulnerabilities such as SQL Injection, cross scripting by default.
- **Database Integration:** Django takes care of database integration for us. The major databases that are primarily used with Django are PostgreSQL, MySQL, SQLite, and Oracle. SQLite is the default database. Though other third-party databases work well too.
- **Database Migration:** Django uses ORM to map objects to databases. Hence database migration is easy.

Besides Django is well documented and has a large community support. So no doubt that Django is so popular to develop web applications quickly and easily.

In the following section, we'll develop a web application that displays "Hello World". We'll deploy it on our localhost.

## Django Hello World Project

**Python**, **Virtualenv** and **Django** are the three major things that need to be installed for setting up our web application.

### Installing Python

Make sure you've installed Python3 on your system. You can check that by running the following command on your system.

```
python3
```

```
anupamchugh@Anupams-MacBook-Air ~$ python3
Python 3.6.4 (v3.6.4:d48eeced5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("a")
a
>>>
```

If you don't have Python installed you can download it from their [website](#).

On Mac OSX you can install it quickly using [Homebrew](#):

```
brew install python3
```

Since Mac OS already comes with python. Installing python would keep both the Python2 and Python3 versions.

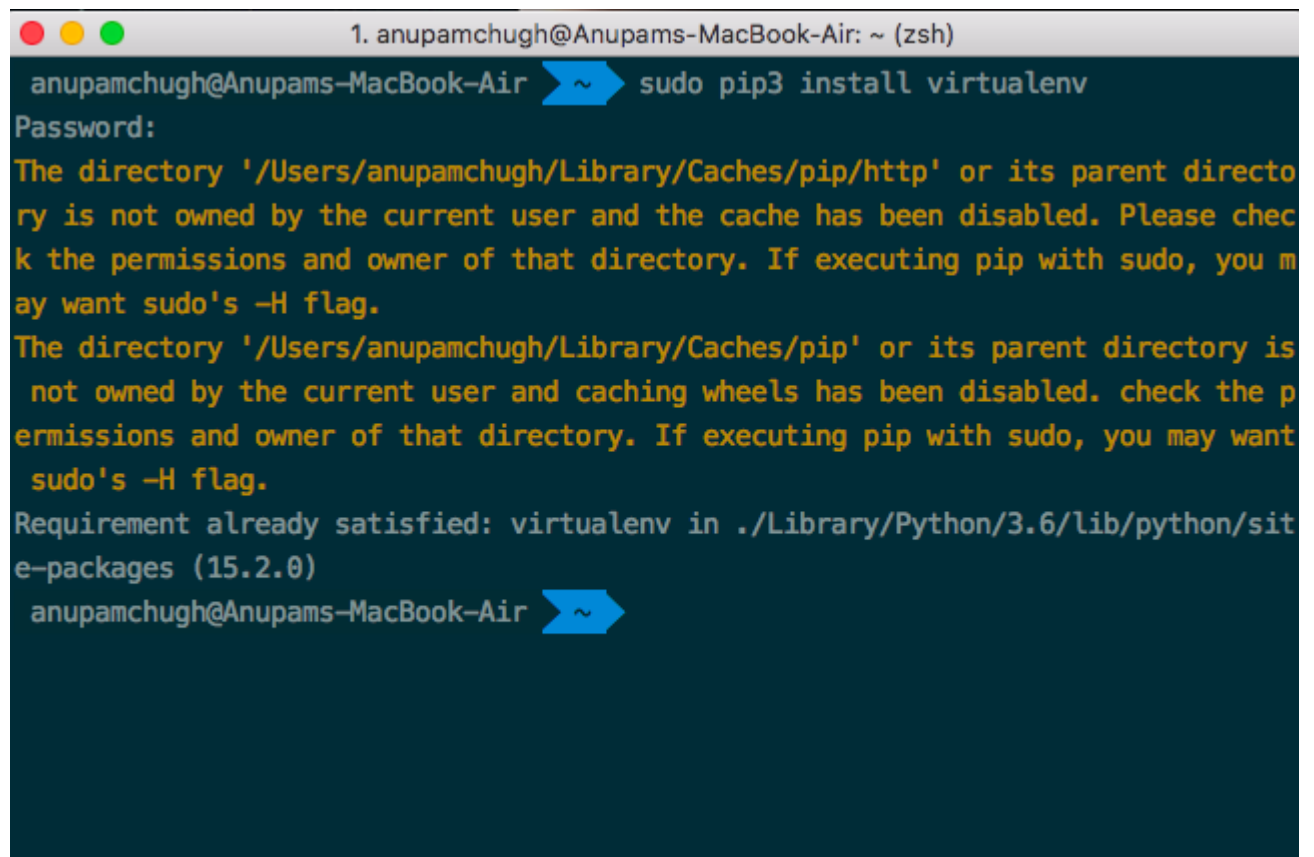
### Installing Virtual Environment

Virtual Environment is used to isolate our Python and Django project from the rest of the system. To install `virtualenv`, we'll use the [pip](#) tool that comes with Python.

Run the following command in your terminal.

```
sudo pip3 install virtualenv
```

The following is the screenshot from my terminal:

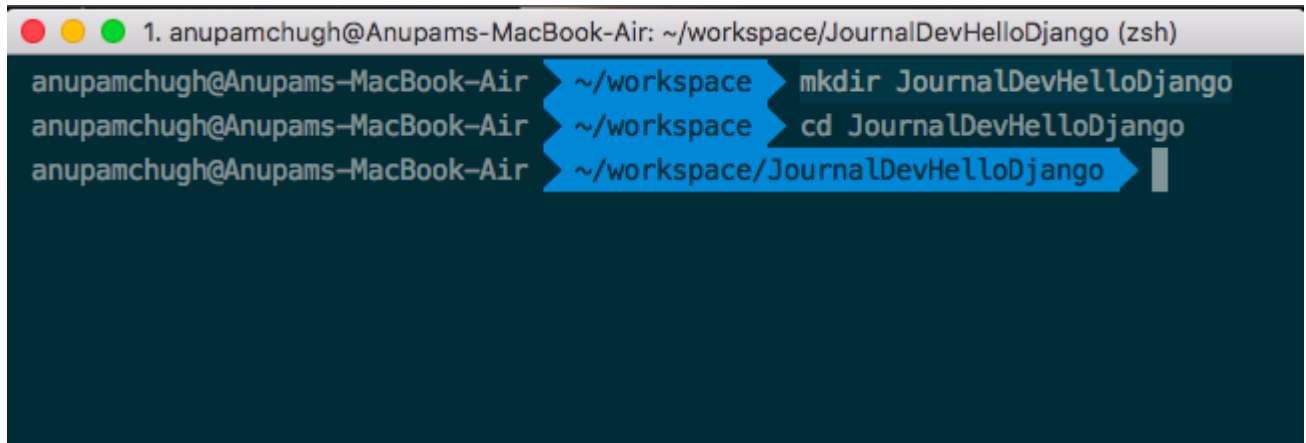


```
1. anupamchugh@Anupams-MacBook-Air: ~ (zsh)
anupamchugh@Anupams-MacBook-Air ~$ sudo pip3 install virtualenv
Password:
The directory '/Users/anupamchugh/Library/Caches/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/Users/anupamchugh/Library/Caches/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Requirement already satisfied: virtualenv in ./Library/Python/3.6/lib/python/site-packages (15.2.0)
anupamchugh@Anupams-MacBook-Air ~$
```

Since I'd already installed it before, it won't do it again for me. Ignore the warning.

Now we need to create our Project. Run the following commands in the terminal to create a new folder for your Python Django Project.

```
mkdir JournalDevHelloDjango
cd JournalDevHelloDjango
```

A terminal window titled "1. anupamchugh@Anupams-MacBook-Air: ~/workspace/JournalDevHelloDjango (zsh)". It shows three lines of commands and their outputs: 1. Command: `mkdir JournalDevHelloDjango`, Output: `~/workspace`. 2. Command: `cd JournalDevHelloDjango`, Output: `~/workspace`. 3. Command: (blank), Output: `~/workspace/JournalDevHelloDjango`. The outputs are highlighted with blue arrows pointing from the command line to the output text.

```
anupamchugh@Anupams-MacBook-Air: ~/workspace/JournalDevHelloDjango (zsh)
anupamchugh@Anupams-MacBook-Air: ~/workspace$ mkdir JournalDevHelloDjango
anupamchugh@Anupams-MacBook-Air: ~/workspace$ cd JournalDevHelloDjango
anupamchugh@Anupams-MacBook-Air: ~/workspace/JournalDevHelloDjango$
```

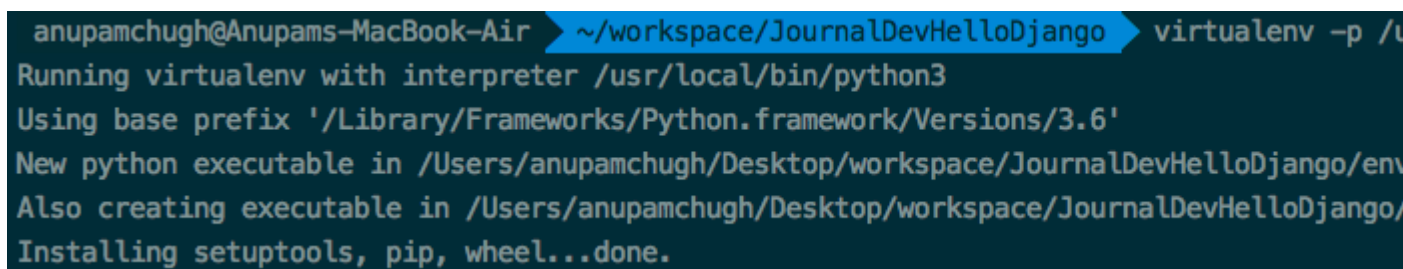
Now add the virtual environment in this project using the following command:

```
virtualenv -p /usr/local/bin/python3 env
```

`-p` tells the virtual environment the path of the python we wish to use. We're using python3 in this tutorial.

`env` is the environment name we've set. You can set any name in it's place.

The terminal displays the following:

A terminal window showing the output of the `virtualenv -p /usr/local/bin/python3 env` command. The output text is: "Running virtualenv with interpreter /usr/local/bin/python3", "Using base prefix '/Library/Frameworks/Python.framework/Versions/3.6'", "New python executable in /Users/anupamchugh/Desktop/workspace/JournalDevHelloDjango/env", "Also creating executable in /Users/anupamchugh/Desktop/workspace/JournalDevHelloDjango/", and "Installing setuptools, pip, wheel...done.".

```
anupamchugh@Anupams-MacBook-Air: ~/workspace/JournalDevHelloDjango$ virtualenv -p /usr/local/bin/python3 env
Running virtualenv with interpreter /usr/local/bin/python3
Using base prefix '/Library/Frameworks/Python.framework/Versions/3.6'
New python executable in /Users/anupamchugh/Desktop/workspace/JournalDevHelloDjango/env
Also creating executable in /Users/anupamchugh/Desktop/workspace/JournalDevHelloDjango/
Installing setuptools, pip, wheel...done.
```

Once it's done, we're ready to start our Virtual Environment:

For mac osx we run the following command:

```
source env/bin/activate
```

For windows:

```
env/script/activate
```

The terminal shows the following:

```
anupamchugh@Anupams-MacBook-Air ~/workspace/JournalDevHelloDjango source env/bin/
(env) anupamchugh@Anupams-MacBook-Air ~/workspace/JournalDevHelloDjango
```

Note: For mac osx terminal the activate virtual environment name would be displayed in each command from now. `(env)`

To stop the virtual environment, we need to run `deactivate` in the terminal.

Don't do this for now since we need to create our first application quickly.

Let's install Django next.

### Installing Django

Just execute the following command in your terminal to install Django.

```
pip3 install django
```

```
(env) anupamchugh@Anupams-MacBook-Air ~/workspace/JournalDevHelloDjango pip3 insta
Collecting django
  Using cached https://files.pythonhosted.org/packages/23/91/2245462e57798e9251de87c88b
61ef7bd3/Django-2.0.5-py3-none-any.whl
Collecting pytz (from django)
  Using cached https://files.pythonhosted.org/packages/dc/83/15f7833b70d3e067ca91467ca2
06b120f2/pytz-2018.4-py2.py3-none-any.whl
Installing collected packages: pytz, django
Successfully installed django-2.0.5 pytz-2018.4
(env) anupamchugh@Anupams-MacBook-Air ~/workspace/JournalDevHelloDjango
```

Congratulations if you've come so far! You're all set to create your first Django project.

### Create New Django Project

To create a new project in the virtual environment execute the following command:

```
django-admin startproject HelloDjango
django-admin is a built-in command.
```

Our project structure looks like this right now:

```
(env) anupamchugh@Anupams-MacBook-Air > ~/workspace/JournalDevHelloDjango/HelloDjango
.
├── HelloDjango
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

1 directory, 5 files
```

It consists of an inner folder with the same name i.e. `HelloDjango` in this case.

Some of the pre-defined python files that are created are:

- `__init__.py`: This indicates that this directory should be considered as a Python package.
- `settings.py`: This file contains all the project's configuration.
- `urls.py`: This file is used for URL routing. It's here where you'll define the URL paths to different directories of your application.
- `wsgi.py`: An entry-point for WSGI-compatible web servers to serve your project. It's just a gateway. We'll look into this in a later tutorial.
- `manage.py`: This is used to manage our Django project. Over this file, we'll run commands to deploy, test our application as well as do database migrations if needed. It's a command line utility. This file is present in the outer HelloDjango folder.

Run the following command to deploy the project onto the server.

```
python3 manage.py runserver
```

Ignore the migration errors.

By default, the server has started on port 8000.

Open the URL: `http://127.0.0.1:8000/` in your browser and you should see this:



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



**Django Documentation**  
Topics, references, & how-to's



**Tutorial: A Polling App**  
Get started with Django



**Django Community**  
Connect, get help, or contribute

To run the server on a different port simply put the port number after the command as `python3 manage.py runserver 8080`.

To run the server on all IPs present in the network do: `python3 manage.py runserver 0:8000`

To stop the server do a `Ctrl + C`.

Now let's create a Hello World Django Web Application.

### Creating a Django Web Application

A single Django project can contain one or more Django Apps.

Our `settings.py` file already consists of a few pre-defined installed Django Apps:

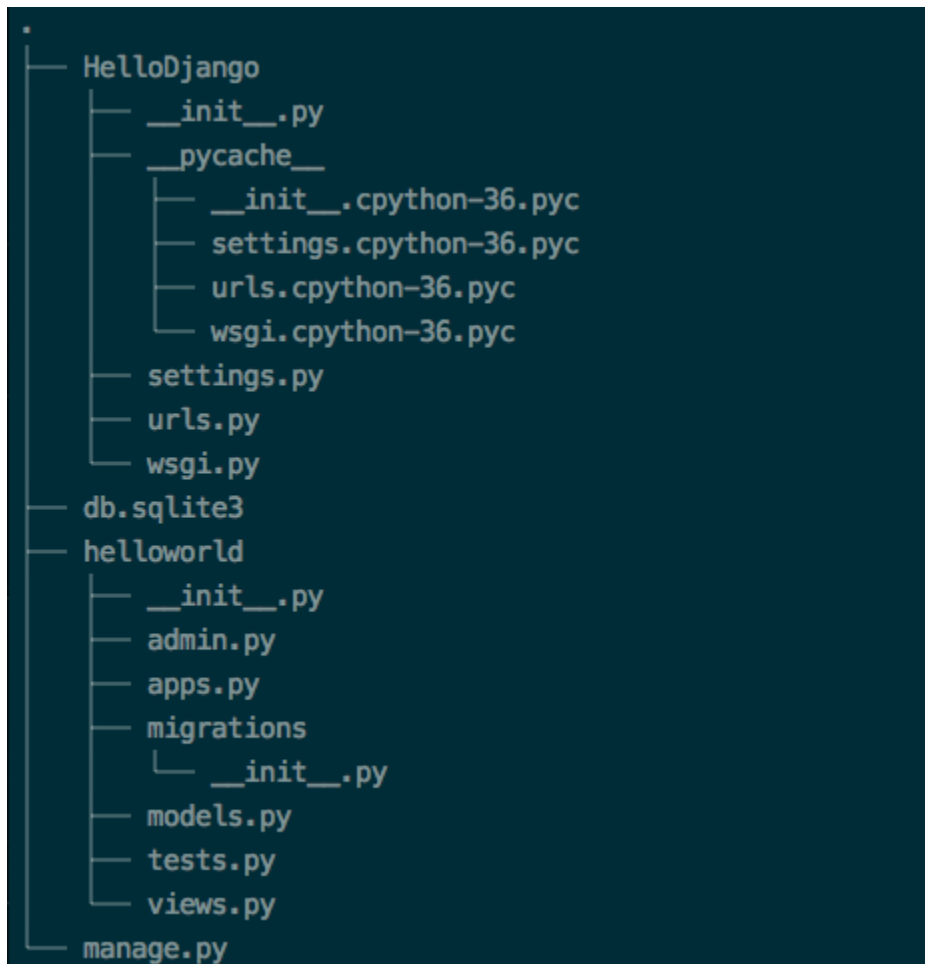
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

To create your own Django App goto the outer HelloDjango folder and run the following command in the terminal.

```
django-admin startapp helloworld
```

This creates a `helloworld` app in our Django project.

Following is the structure of our project now:



Let's look at the files in the Django App:

- `models.py`: Entities of the Django App are defined here. These are translated into the database table.
- `migrations/`: This folder stores file to keep track of database migration.
- `admin.py`: This is the config file for the Django Admin App that's auto-generated.
- `apps.py`: This is a config file of the current application.
- `tests.py`: Unit Tests of the Django App are written here.
- `views.py`: This is used to display the view in our web app. It consists of a function that does a network request and gets back the response.



- `urls.py`: This does the URL routing in the current app only. It's different from the `urls.py` file present in the project which can do URL dispatches at the project level. That means it can contain URL routing list of different apps.

`pycache` gets generated when the project is deployed on the server.

Now add the `helloworld` app in the `settings.py` file of our HelloDjango Project.

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

Let's do something in our `views.py` file now:

```
from django.shortcuts import render
# Create your views here.
```

```
from django.http import HttpResponse
```

```
def hello(request):
    return HttpResponse('Hello, World!, Django')
```

- `from django.shortcuts import render` is used to load the Django view template.
- `hello` function returns a response that prints the string onto the screen.

## URL Mapping

URL Mapping refers to mapping URLs to Django Views. We use regular expressions and tuples for URL Mapping.

URL Mapping is done in the `url.py` Python file.

An example `url.py` file is given below:

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
```

```
url(r'^hello/', 'myapp.views.hello', name = 'hello'),
url(r'^$', include('example.urls')) # tell django to read
urls.py in example app
]
```

- First Param: Regular Expression – The first parameter passed is typically a regular expression. `r` is followed by a `^`. A `^$` refers to an empty path. That means root path.
- Second Param: View Path – Here you pass the path to the View that's triggered from that regular expression url. You can also include a new url router here which refers to a different Django app/module. For this `include()` is used.
- Third param: url name – In order to do a reverse lookup, the name of the url path is passed.

The second url pattern in the previous code invokes the hello function from the views.py file when the url `127.0.0.1/hello` is run.

`re_path` and `path` are the new keywords introduced since Django 2.0.

from django.urls import include, path, re\_path

`re_path()` is similar to `url()`

`path()` is devoid of regular expressions.

`url()` would be deprecated in future.

## URL Routing

Let's do the URL routing so that this view is loaded instead of the previous Django documentation view.

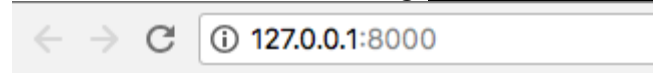
Edit the `urls.py` in the inner HelloDjango project folder with the following code:

```
from django.contrib import admin
from django.urls import urls
from helloworld import views

urlpatterns = [
    url(r'^$', views.hello, name='hello'),
    url(r'^admin/', admin.site.urls),
]
```

`^$` matches with an empty path. So the views.py would open on the homepage itself.

Let's run the server using `python3 manage.py runserver`



Hello, World!, Django

Woah! Finally we've created our first Django Project and a Hello World Web App.

Now you can kill the server using `ctrl + c` command. Then run `deactivate` command to shutdown the virtual environment.