**Steps to Create a Django project:**



**Cheat Sheet**

1. **Activate virtual environment:** source /Users/catherinebose/Desktop/CodingDojo/PythonDojo/myEnvironments/djangoEnv/bin/activate
2. django-admin startproject myFirstDjangoTemplates
3. cd myFirstDjangoTemplates
4. mkdir apps
5. cd apps
6. touch \_\_init\_\_.py
7. python ../manage.py startapp first\_app
8. cd first\_app/
9. touch urls.py
10. Go to settings.py INSTALLED\_APPS = [  **'apps.first\_app', ### added this line!**
11. Now open main/urls.py in your text editor. We're going to modify this document in order to use routes from our first\_app:

**Modify that file, so it looks like this:**

from django.conf.urls import url, **include** # Notice we added include

from django.contrib import admin

urlpatterns = [

**url(r'^', include('apps.first\_app.urls')) #** And now we use the include function to pull in our first\_app.urls...

]

1. Open urls.py in your text editor. Add the code below:

***apps/first\_app/urls.py***

from django.conf.urls import url

from . import views # This line is new!

urlpatterns = [

url(r'^$', views.index) # This line has changed!

]

13.Let’s build our first views function, **index**, inside of **views**:

*apps/first\_app/views.py*

from django.shortcuts import render, HttpResponse, redirect

# the index function is called when root is visited

def index(request):

response = "Hello, I am your first request!"

return HttpResponse(response)

14.Go to your main folder and run python **manage.py runserver**from the terminal. **Then proceed to localhost:8000.** You should see a simple page congratulating you on a Django-powered page. Stop the server now (ctrl-C) before moving on to the next step.

**15.Examples of Regular Expression**

* url(r'^articles/(?P\d+)$', views.show)
* url(r'^articles/(?P\w+)$', views.show\_word)
* url(r'^articles/2003/$', views.special\_case\_2003)
* url(r'^articles/(?P[0-9]{4})$', views.year\_archive)
* url(r'^articles/(?P[0-9]{4})/(?P[0-9]{2}$', views.month\_archive

Django installation is smooth sailing:

**Step 1**

Create a virtual environment in your virtual environment folder.

**1.If you're using Mac:**

**> virtualenv djangoEnv**

**> source djangoEnv/bin/activate**

**(djangoEnv)>pip install Django==1.10**

**If you're using a PC:**

> python -m virtualenv djangoEnv

> call djangoEnv/scripts/activate

(djangoEnv)>pip install django

1. **Activate virtual environment:** source /Users/catherinebose/Desktop/CodingDojo/PythonDojo/myEnvironments/djangoEnv/bin/activate

**Step 2**

Now whenever you want to do a Django project, you will be activating this same virtualenv.

**Creating a project**

Let's make a project! Navigate to a location where you want to create your project (e.g. either your desktop folder or your user folder). We need to do a little set-up work to get our Django project up and running:

1 **> django-admin startproject main**

This is going to create the main project for us, from which we’re going to have to do a few small set-up things before we can really get going:

NOTE: All our videos are using bash, but to use the same commands on Windows Command Prompt, the analogous command for **touch** is **nul>**

Navigate into the main directory that you just created. At this point you could run "python manage.py runserver" and take a look at what your development server delivers to the browser.

2 **> cd main**

**# Make a new apps directory**

**3 > mkdir apps**

**# Navigate into apps**

**4 > cd apps**

Next you'll create an \_\_init\_\_.py file. This lets the Python interpreter know that this folder is present and may be accessed. Now, other Python modules can look in this directory and access the files and therefore the code they contain. Don't forget this step!

**# Bash: (Mac, Linux)**

**5 > touch \_\_init\_\_.py**

**# PCs: (Command Prompt)**

**> nul> \_\_init\_\_.py**

Notice that when you started a project a manage.py file was automatically created. That file can be called using one of several commands to request a specific operation. Let's do that from within the apps directory to start a new app.

6 **> python ../manage.py startapp first\_app**

1. By running python **../manage.py startapp** first\_app you've created a new app. In your apps folder – let's make sure we can use our new app. We have to do a couple more things before we can get started.

In settings.py (this file lives inside the main directory. The main directory is alongside apps and manage.py) we are going to have to add our application to the project. That means adding apps.first\_app to our INSTALLED\_APPS list:

8.  **# Inside main/settings.py**

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

# BECOMES:

INSTALLED\_APPS = [

8.  **'apps.first\_app', ### added this line!**

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

1. Go to your main folder and run python **manage.py runserver**from the terminal. Then proceed to localhost:8000. You should see a simple page congratulating you on a Django-powered page. Stop the server now (ctrl-C) before moving on to the next step.
2. **URLs**

**Django's urls are the routing part of Django we talked about earlier. To make sure we're sending the right routes to our first app, we'll set up our main project's urls to make requests to our app's routing file. This file, urls.py, will be inside of our directory 'apps/first\_app'.**

**Now open main/urls.py in your text editor. We're going to modify this document in order to use routes from our first\_app:**

**Modify that file, so it looks like this:**

from django.conf.urls import url, include # Notice we added include

from django.contrib import admin

urlpatterns = [

**url(r'^', include('apps.first\_app.urls')) #** And now we use the include function to pull in our first\_app.urls...

]

11.Our app's urls file doesn't exist yet. Let's make it. From the command line:

**> cd apps/first\_app**

**> touch urls.py**

1. Open urls.py in your text editor. Add the code below:

***apps/first\_app/urls.py***

from django.conf.urls import url

from . import views # This line is new!

urlpatterns = [

url(r'^$', views.index) # This line has changed!

]

Let's take a closer look at urls.py:

* from django.conf.urls import url
  + This gives us access to the function url
* from . import views
  + This explicitly imports views.py in the current folder.
* url(r'^$', views.index)
* ^ stands for start and $ stands for end
  + Uses the url method in a way that’s very similar to the @app.route method in flask. The r after the ( tells Python to interpret the following as a raw string, so it won't escape any special characters -- useful when dealing with regex strings! We'll talk about regex in a bit more depth in the next tab.
  + In this case, our regex will exactly match an empty string. That means if you were to go to localhost:8000/, Django (after removing the '/' automatically) will check if your url matches any of the patterns in the urlpattern list.
  + In this case, it does! An empty string is what r'^$' looks for. Since the pattern matches, we run the views.index method.
  + url() will eventually take another parameter, something like name='index', which we’ll introduce when we talk about **named routes**.
* Note that, unlike a Flask route where there is an HTTP method (e.g. “GET” and/or “POST”), Django doesn’t care. We (the developer) figure that out in the method by accessing request.method in our function.

1. **Views**

Let’s build our first views function, **index**, inside of **views**:

*apps/first\_app/views.py*

from django.shortcuts import render, HttpResponse, redirect

# the index function is called when root is visited

def index(request):

response = "Hello, I am your first request!"

return HttpResponse(response)

Look back at your code in urls.py. You should see that we have called url, passing in a regex that will be used to attempt to find a match for an incoming HTTP request. Next, we pass in views.index. We now have access to all the functions in our views file. When the given route is visited, the code in that function executes.

In the above code we respond to an HTTP request with a simple string. It's passed to the client and displayed there. That's not going to help us much with creating a complex website. We'll need to render some HTML from some templates. Let's take a look:

1. **Regular Expressions**

Regular expressions, commonly known as regex, are a set of rules for identifying or matching strings. Many programming languages, including Python, support use of regular expressions for matching and searching strings.

Common uses include searching string inputs from users. Search engines and other form input, like user registration and login, are great examples of correct uses for regular expressions. As you've seen, Django uses regex to match url patterns in routing. You won't need to do anything too complex when working with Django, but you can use the following reference for some of the most common expressions you'll use.

A good resource for you to get familiar with Regex is regexr.com

Here is a cheat sheet for expressions you'll be using regularly:

* **'^'** Matches the following characters at the beginning of a string. Example: '^a' matches 'anna' but not 'banana'.
* **'$'** Matches the previous characters at the end of a string. Example: 'a$' matches 'anna' and 'banana' but not 'fan'.
* **'[]'** Matches any value in a range. Example: '[0-9]' matches '9' and '9s'.
* **'{n}'** Matches n number or more repetitions of the preceding pattern. Example: '[0-9]{2}' matches '91' and '9834' but not '9'
* **\d** Matches digits.  Example: "\d" matches "8" and "877x"
  + \d+ matches a string with one or more digits
* **\w** Matches characters.
  + \w+ matches a string with one or more character/word

All of the above examples are pretty simple. You can combine any of the above patterns to match complex strings. You'll see lots of examples of regex in Django urls, but they follow a simple pattern.

Use Python's [regex documentation](https://docs.python.org/2/library/re.html) as a reference if you need anything more complex than the above. You most likely won't. If you're curious and you have some spare time try [Regex One](https://regexone.com/) or [Regex 101](https://regex101.com/) to practice your patterns.

**Examples of Regular Expression**

* url(r'^articles/(?P\d+)$', views.show)
* url(r'^articles/(?P\w+)$', views.show\_word)
* url(r'^articles/2003/$', views.special\_case\_2003)
* url(r'^articles/(?P[0-9]{4})$', views.year\_archive)
* url(r'^articles/(?P[0-9]{4})/(?P[0-9]{2}$', views.month\_archive)