How to start project in Django:

Before you can use Django, you'll need to install it.

```
For Windows: ...\> py -m pip install Django
```

```
For Mac&Linux: $ python -m pip install Django
```

To verify that Django can be seen by Python, type `python` from your shell. Then at the Python prompt, try to import Django:

```
>>> import django
>>> print(django.get_version())
3.0
```

After that you realized that the Django was installed, you must CD to directory to make Django Webapp.

Then:

```
...\> django-admin startproject mysite
For Linux & mac: $ django-admin startproject mysite
```

(notes: …\> is your location of Python app, Mysite is your app name)

After that these are created:

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

The outer `mysite/` root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.

`manage.py`: A command-line utility that lets you interact with this Django project in various ways.

The inner `mysite/` directory is the actual Python package for your project.

`mysite/__init__.py`: An empty file that tells Python that this directory should be considered a Python package.

`mysite/settings.py`: Settings/configuration for this Django project.

`mysite/urls.py`: The URL declarations for this Django project; a “table of contents” of your Django-powered site.

`mysite/asgi.py`: An entry-point for ASGI-compatible web servers to serve your project.

`mysite/wsgi.py`: An entry-point for WSGI-compatible web servers to serve your project.

The development server

Let’s verify your Django project works. Change into the outer `mysite` directory, if you haven’t already, and run the following commands:

For linux & mac: `python manage.py runserver`
`For windows : ...\> py manage.py runserver`

You’ll see the following output on the command line:

```
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.

January 29, 2020 - 15:50:53
Django version 3.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

You can change the port too but its better don’t do it:

Commend:

```
...\> py manage.py runserver 8080
```

Or

```
...\> py manage.py runserver 0:8000
```

For linux & mac:

```
$ python manage.py runserver 0:8000
$ python manage.py runserver 8080
```

Can help You more:

Projects vs. apps

What's the difference between a project and an app? An app is a Web application that does something – e.g., a Weblog system, a database of public records or a small poll app. A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects.

Creating the Polls app:

Now that your environment – a "project" – is set up, you're set to start doing work.

Each application you write in Django consists of a Python package that follows a certain convention. Django comes with a utility that automatically generates the basic directory structure of an app, so you can focus on writing code rather than creating directories.

```
...\> py manage.py startapp polls
```

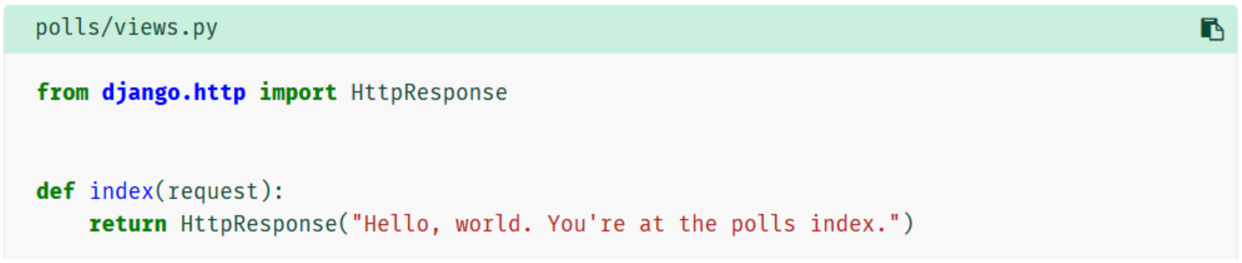
And for linux & mac:

```
$ python manage.py startapp polls
```

That'll create a directory `polls`, which is laid out like this:

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Write First view:

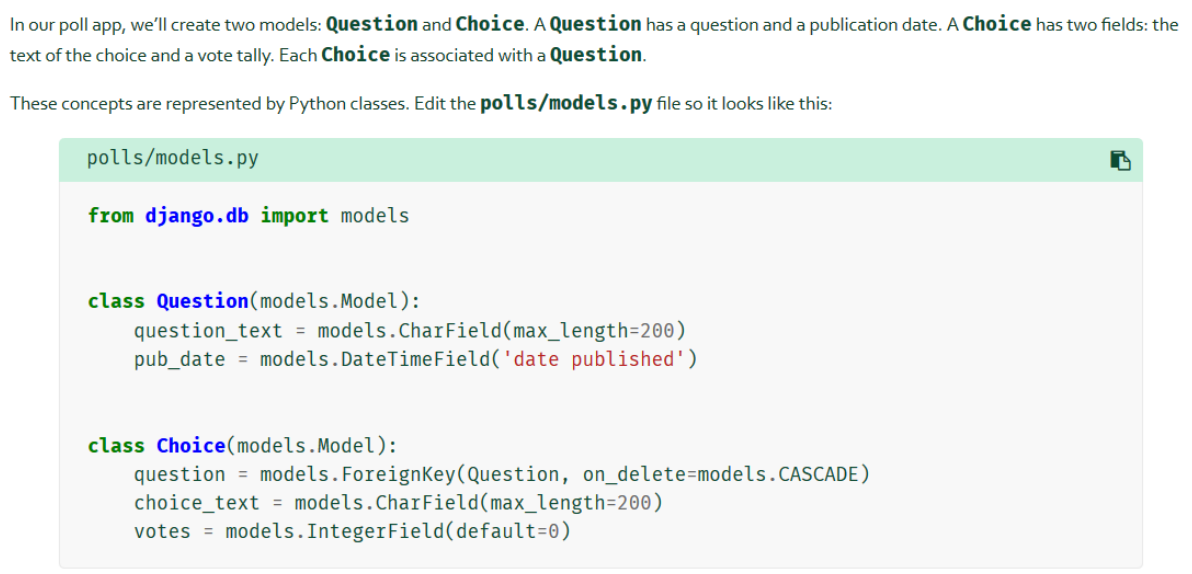


polls/views.py

```python
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

For example: (creating models)



In our poll app, we'll create two models: **Question** and **Choice**. A **Question** has a question and a publication date. A **Choice** has two fields: the text of the choice and a vote tally. Each **Choice** is associated with a **Question**.

These concepts are represented by Python classes. Edit the **polls/models.py** file so it looks like this:

polls/models.py

```python
from django.db import models


class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Database setup:

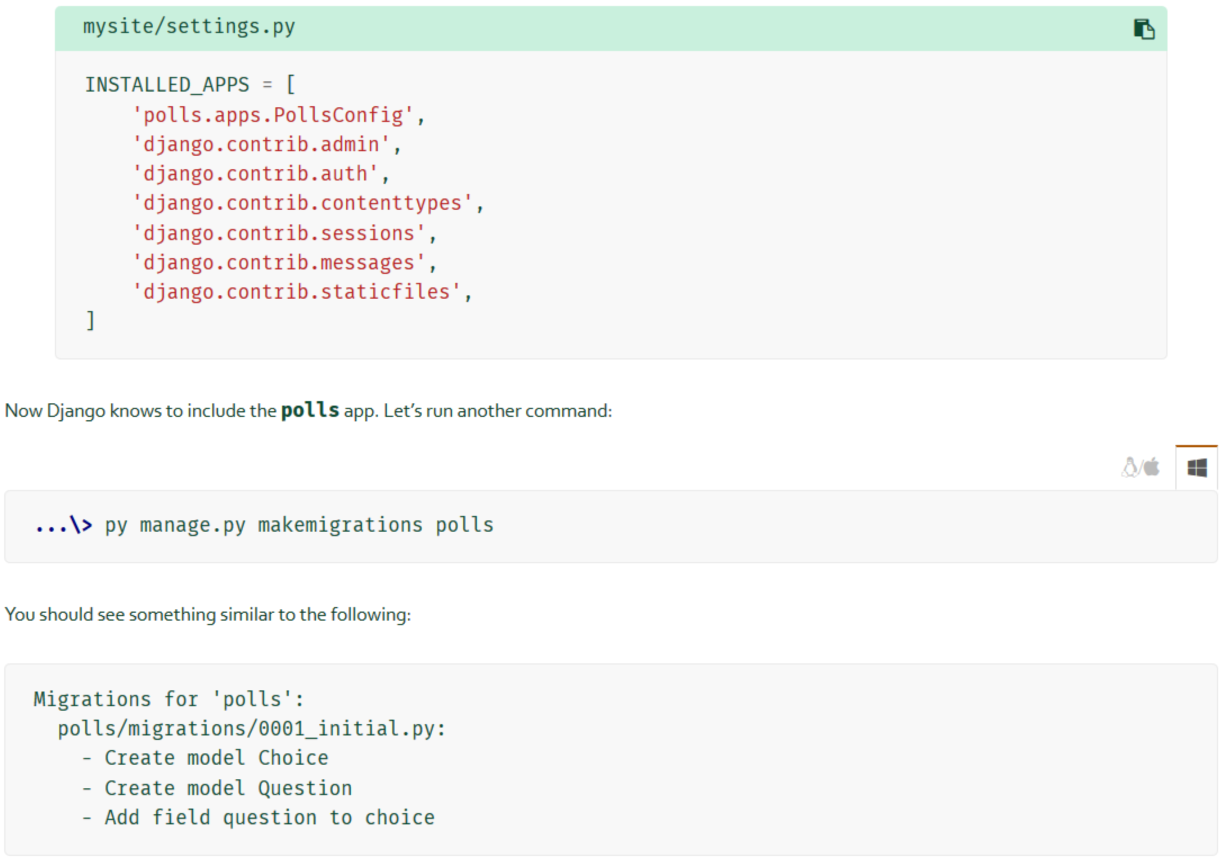
```
$ python manage.py migrate
...\> py manage.py migrate
```

Activating models:

That small bit of model code gives Django a lot of information. With it, Django is able to:

- Create a database schema (`CREATE TABLE` statements) for this app.
- Create a Python database-access API for accessing `Question` and `Choice` objects.

But first we need to tell our project that the `polls` app is installed.



```
mysite/settings.py
```

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

Now Django knows to include the **`polls`** app. Let's run another command:

```
...\> py manage.py makemigrations polls
```

You should see something similar to the following:

```
Migrations for 'polls':
  polls/migrations/0001_initial.py:
    - Create model Choice
    - Create model Question
    - Add field question to choice
```

I think that could you help you so much ,

If you want learn more its better watch Videos

This is just little guides.

Thank you for Reading.