Boxes in green are required commands that everyone will need to run to get the environment ready on their machine.
Boxes in white are outputs, optional commands, or instructions to enter in to get the project started from scratch (when the Git repository is still empty).

1. **Download Python Version 3.7.2** from https://www.python.org/downloads/
    a. In cmd.exe or any other terminal type the following to verify the correct version of python is installed:

    ```
    python -V
    ```

    It should output

    ```
    Python 3.7.2
    ```

    If it doesn't, you can switch which versions you are using by entering:

    ```
    python -3.7.2
    ```

    b. If for whatever reason you get the error: 'python' is not recognized as an internal or external command, operable program or batch file. This is due to the fact that the python.exe was not added to your PATH environment variable. If you do not know how to add a path onto your existing paths in windows follow this tutorial: https://www.architectryan.com/2018/03/17/add-to-the-path-on-windows-10/
        i. The two paths you want to add for python in windows 10 are:
            1. C:\Users\USERNAMEHERE\AppData\Local\Programs\Python\Python37-32
            2. C:\Users\USERNAMEHERE\AppData\Local\Programs\Python\Python37-32\Scripts

        Make sure to replace the USERNAMEHERE with your own account's name.
2. **Download Pip**, a package installer for python which will allow us to get Django on our machine easily and any packages our Django project requires. To do so type in the commands:

    ```
    curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
    ```

    ```
    python get-pip.py
    ```

3. Next we will **install virtualenv** with pip. Virtualenv provides us with an isolated Python environment for which to install all the Django packages and dependencies we might need. This is so we don't clutter up our systems with those packages.

    ```
    pip install virtualenv
    ```

4. **Make the project directory/folder** you plan to start Django in. Inside that directory in your terminal run this command to setup the new virtual environment:

    ```
    virtualenv -p
    ```

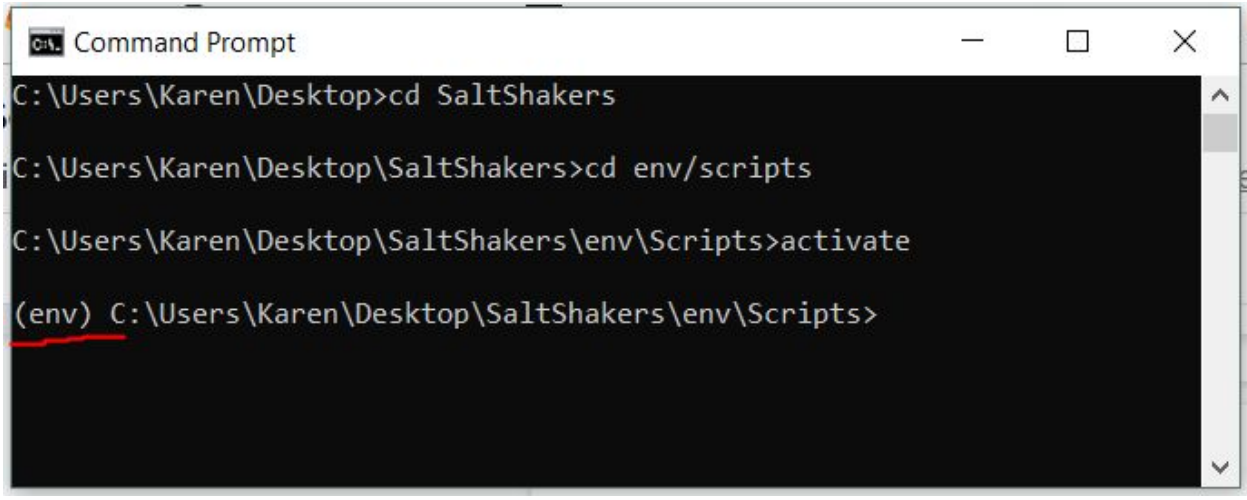
```
C:/Users/USERNAMEHERE/AppData/Local/Programs/Python/Python37-32/python env
```

   a. Inside that directory you should now see a new folder labeled env. It will hold all our packages installed through pip.

5. To **activate the virtualenv** we created:
   a. If using Windows: inside the terminal move into the env/scripts folder inside the project directory we created. In the terminal type:

   ```
   activate
   ```

   When the virtualenv is active you will have (env) to the left of your command prompt like this:



   b. If using Linux: In the terminal type:

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

   c. For when we need to deactivate the virtualenv and move back to working in windows the command will be:

   ```
   deactivate
   ```

6. Now we are ready to **install Django**. With the virtualenv active, in the terminal type:

```
pip install django
```

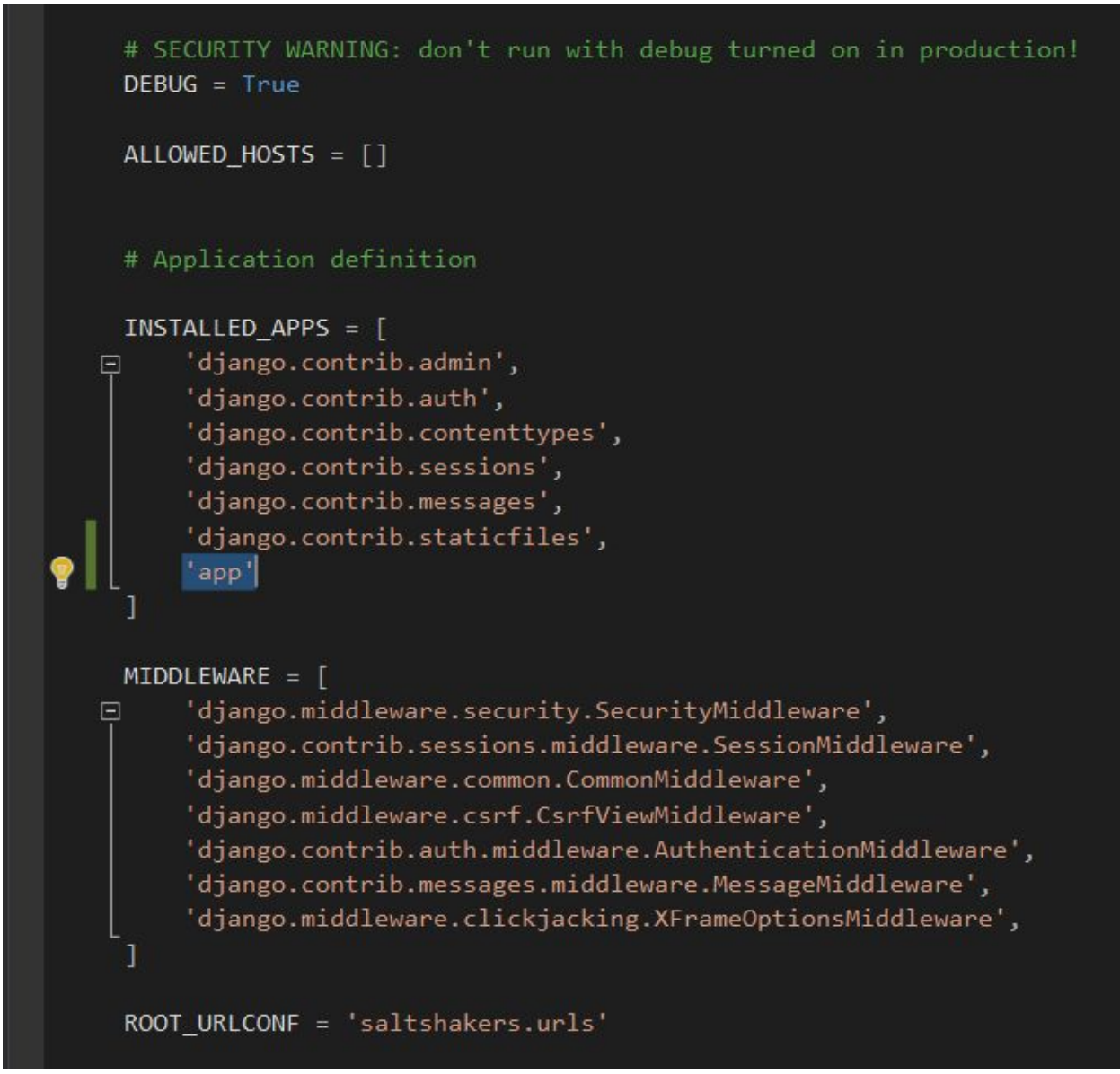
   a. To start a new django project, move back into the original project directory we created and type this into the terminal:

   ```
   django-admin startproject PROJECTNAMEHERE
   ```

   This will create the folder for the project's settings and configurations called PROJECTNAMEHERE, but you might notice that the models, views, and templates used in django aren't present in the project yet. To generate those files run the following command INSIDE the first PROJECTNAMEHERE folder (the folder that contains the manage.py file):

```
python manage.py startapp APPNAMEHERE
```

This will create a folder with the files we need. Next we have to add the created app with the models, views, and templates to the settings.py file for django to pick it up and run. To do this go into the settings.py file inside the second PROJECTNAMEHERE folder and add ‘APPNAMEHERE’ to the INSTALLED_APPS array like such (Note: I simply named the app ‘app’ here):



```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'saltshakers.urls'
```

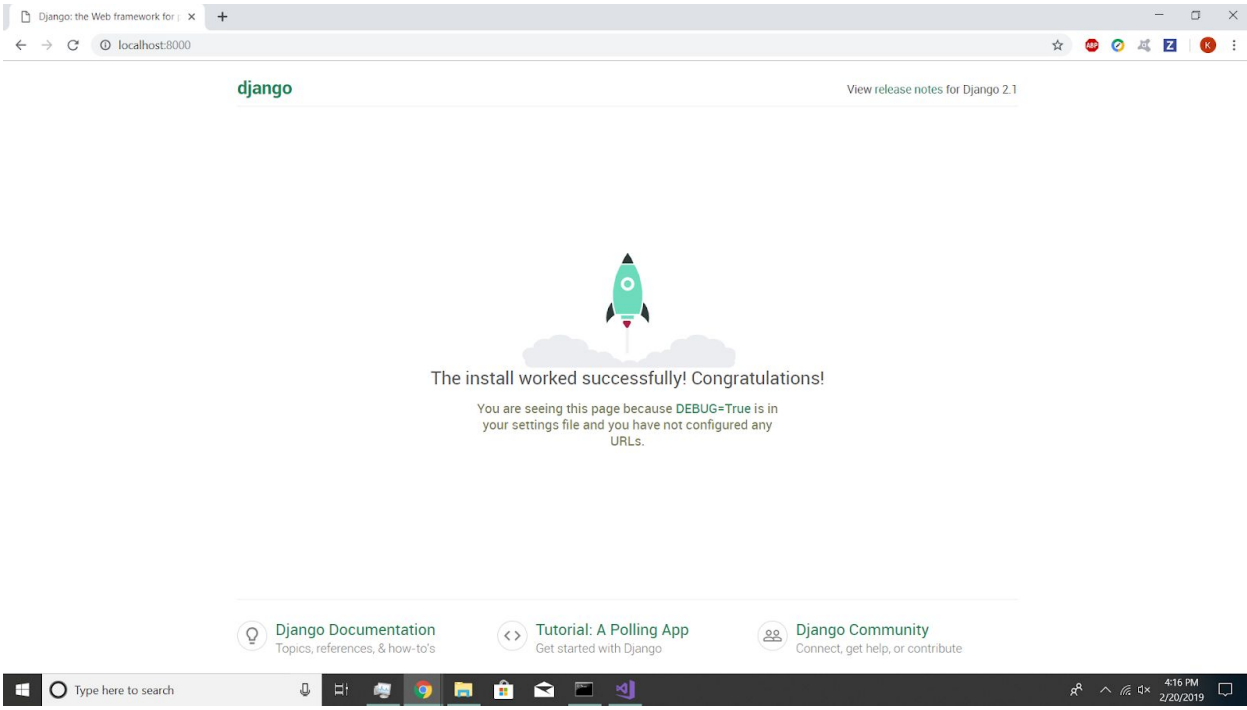
One more thing we should do when starting a new project is to migrate the default django models into django database schemas. Any time after updating the models we will also need to migrate again so that in the future our database will match the models we write for it. The command to migrate the models is:

```
python manage.py migrate
```

b. To **run a already existing django project** (such as a project cloned off of Github or previously created), inside your PROJECTNAMEHERE folder run the command:

```
python manage.py runserver
```

In your browser you may go to either 127.0.0.1:8000 or http://localhost:8000/ to see the django project running. The first time you load the site it may take awhile to get setup. Once running it should look like this in the browser:



c. To stop the django server do CTRL + C in the terminal

CONGRATULATIONS!
We've now setup a properly working Django environment.

One of the tutorials I used online to also help get me set up, and will reiterate some of what I've said here, is:
https://scotch.io/tutorials/build-your-first-python-and-django-application#creating-your-own-app

In the future we can clone our git repository into this virtualenv / the first project directory we created and work from there.