Python Tutorial: virtualenv

Reason:

Virtualenv is a way that you can separate different python environment for different projects. Say that you have multiple projects, and all they reply on a single package, e.g. flask, Django…. Each one of these projects may be using a different version of Django or different version of Flask. Now if you go and upgrade that package in your global size packages, then it could break a couple of your websites that might not what you wanna do. It would be better if each of these projects had an isolated environment where they had only the dependencies and the packages they need and the specific versions they need. That is what virtualenv allows us to do.

**Mac OS set up: using Bash/zsh**

$ pip list

# making the working directory

$ mkdir Environments

# get access into the working directory

$ cd xxxx

$ ls

# Create virtual environment called “project1\_env”

$ virtualenv project1\_env

# Activate the virtualenv

$ source project1\_env/bin/activate

# Check python, pip and pip list

$ which python

$ which pip

$ pip list

#install some packages

$ pip install numpy

$ pip install pytz

# export all these packages and their versions to use in another project

$ pip freeze – local > requirements.txt

# Check we got this file

$ ls

$ cat requirements.txt

# Get out of your virtualenv

$ deactivate

$ which python

# delete virtualenv

$ rm -rf project1\_env

# Create another project with a specific version of python

$ virtualenv -p /usr/bin/python2.6 py26\_env

$ ls

$ source py26\_env/bin/activate

$ which python

$ python –version

$ ls

# install all packages that are listed in requirements.txt

$ pip install -r requirements.txt

$ pip list

**Windows: using “Bash”**

# making the working directory

$ mkdir Environments

# get access into the working directory

$ cd xxxx

$ ls

# Create virtual environment called “project1\_env”

$ virtualenv project1\_env

# Activate the virtualenv

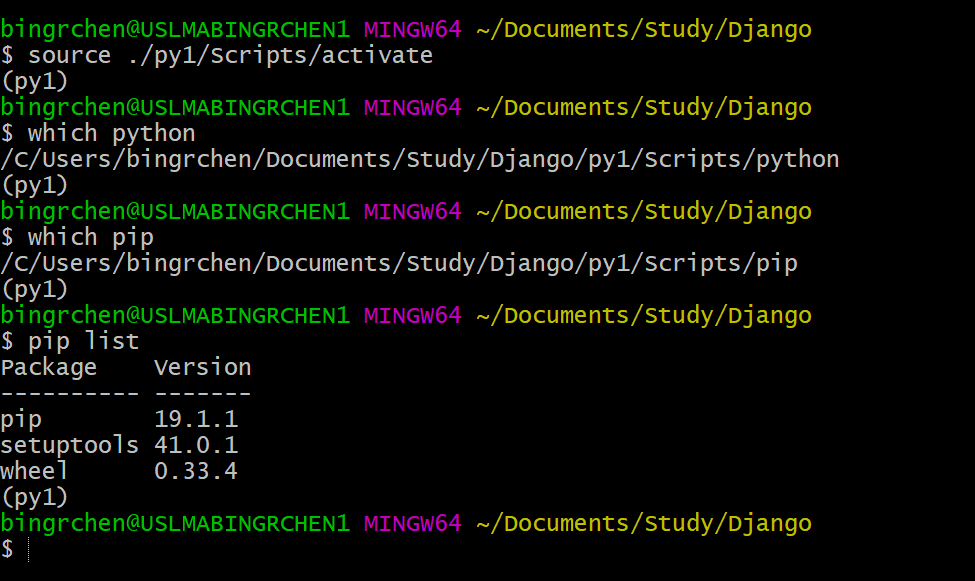
$ source ./project1\_env/bin/activate

# Check python, pip and pip list

$ which python

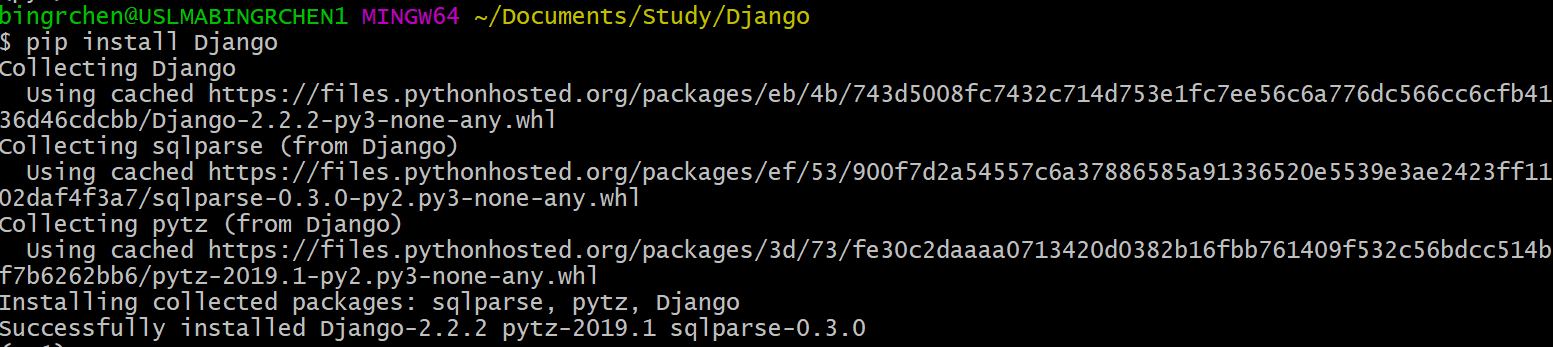
$ which pip

$ pip list



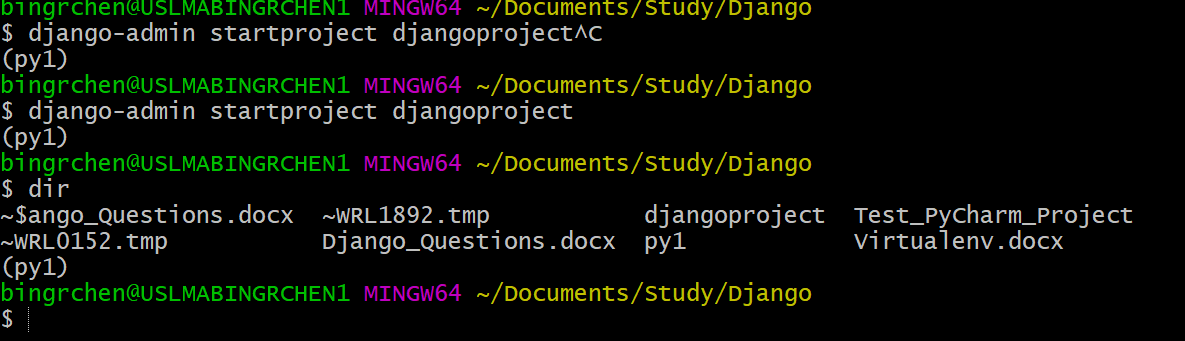
# Install Django

$ pip install Django



# start a Django project called “djangoproject”

$ django-admin startproject djangoproject

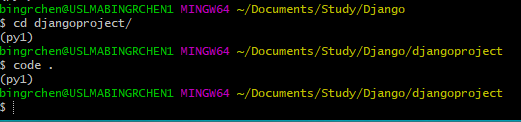


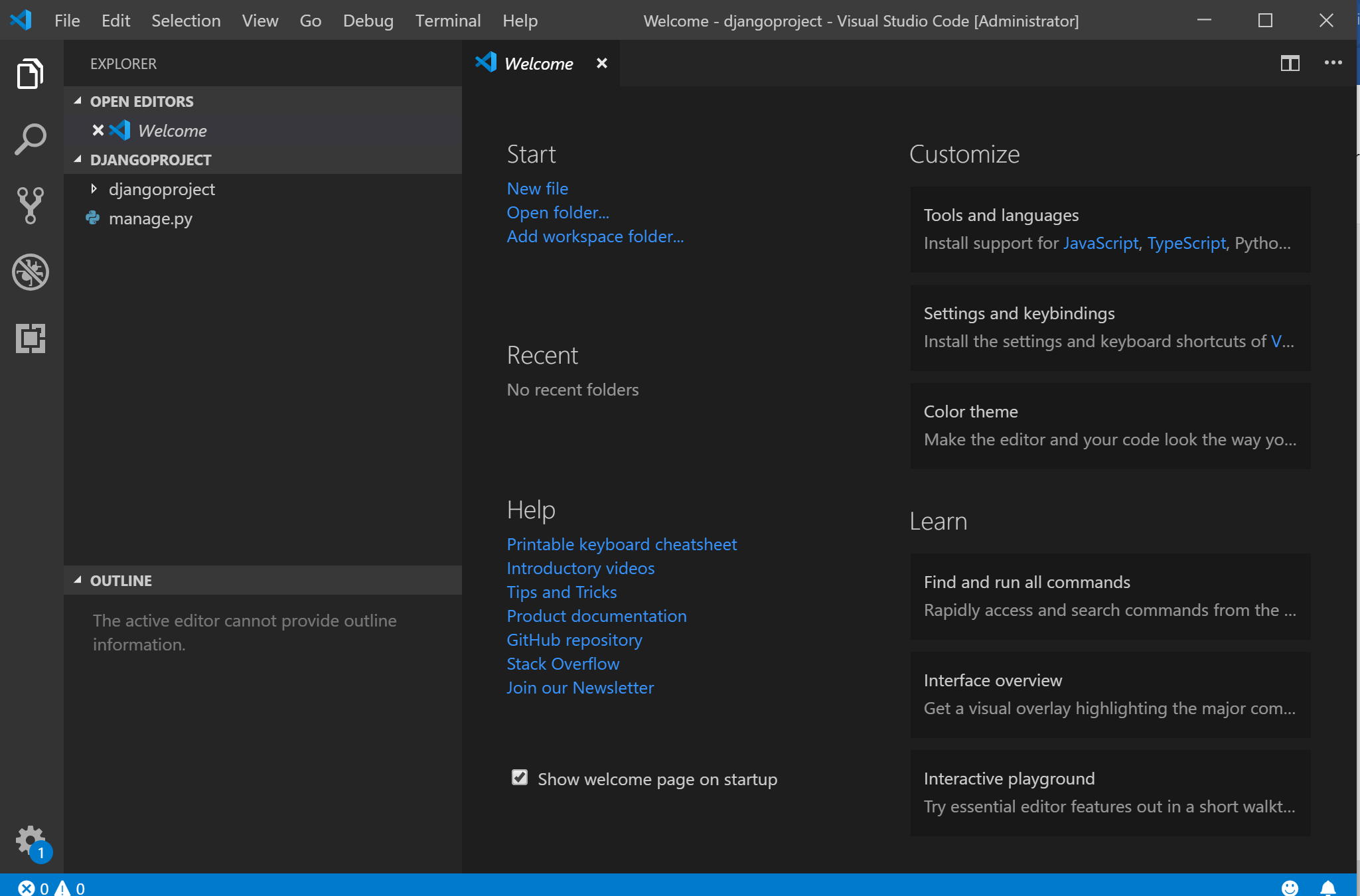
# get access to “djangoproject”

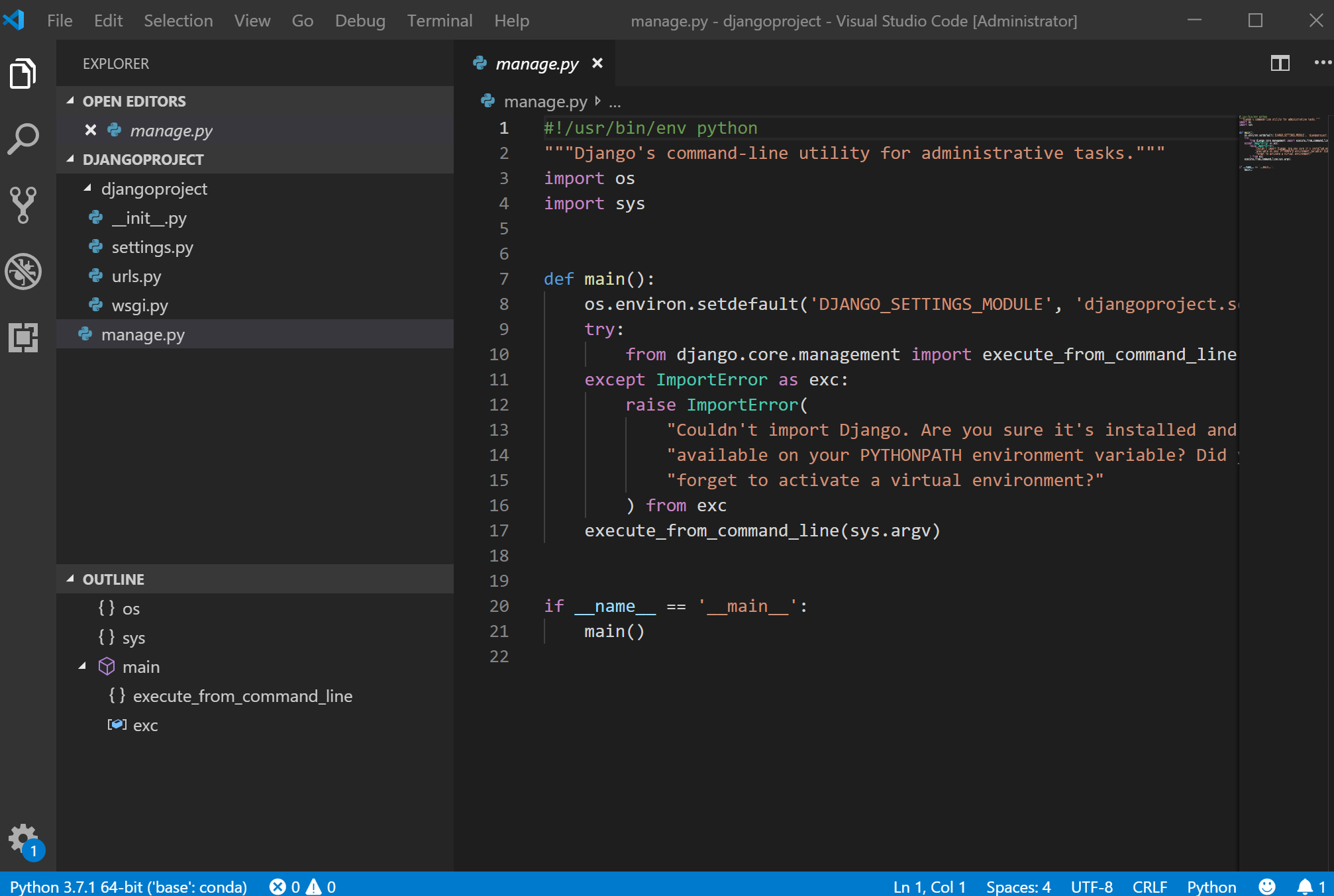
$ cd djangoproject

# open txt (we use VS code editor here)

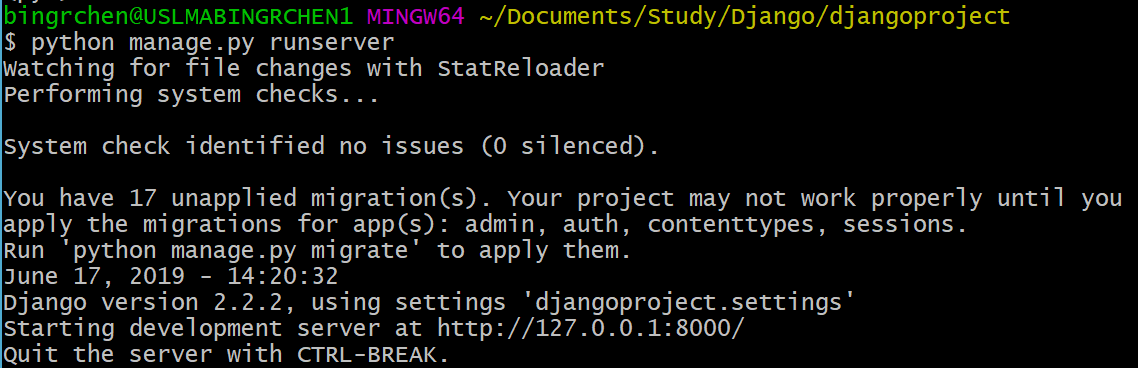
$ code .

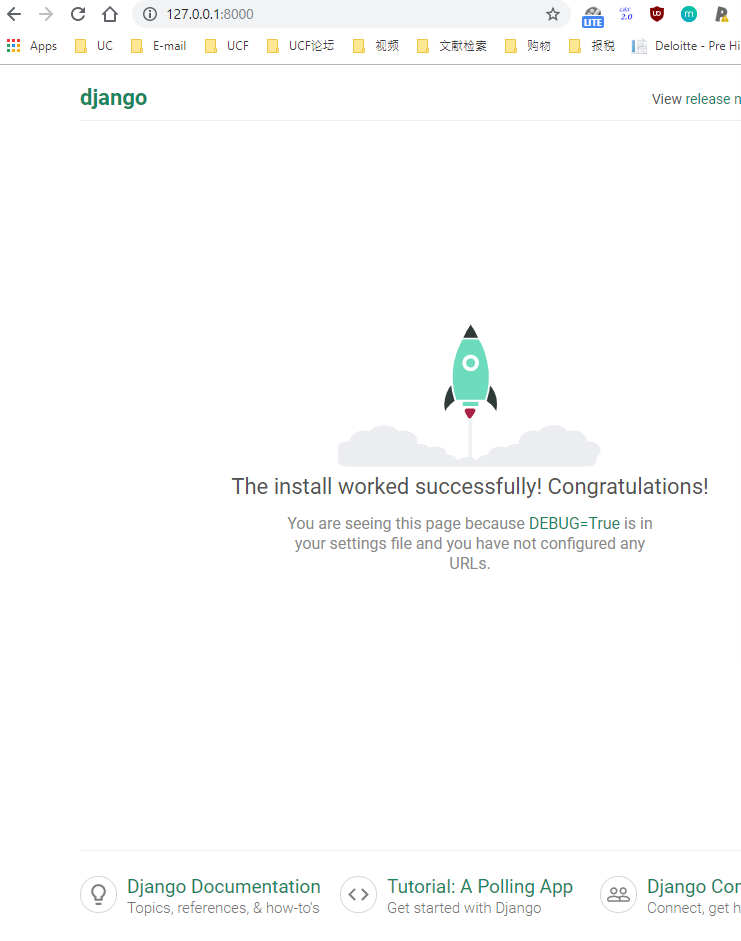






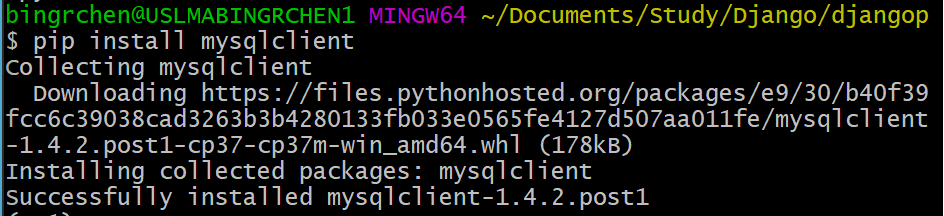
$ python manage.py runserver



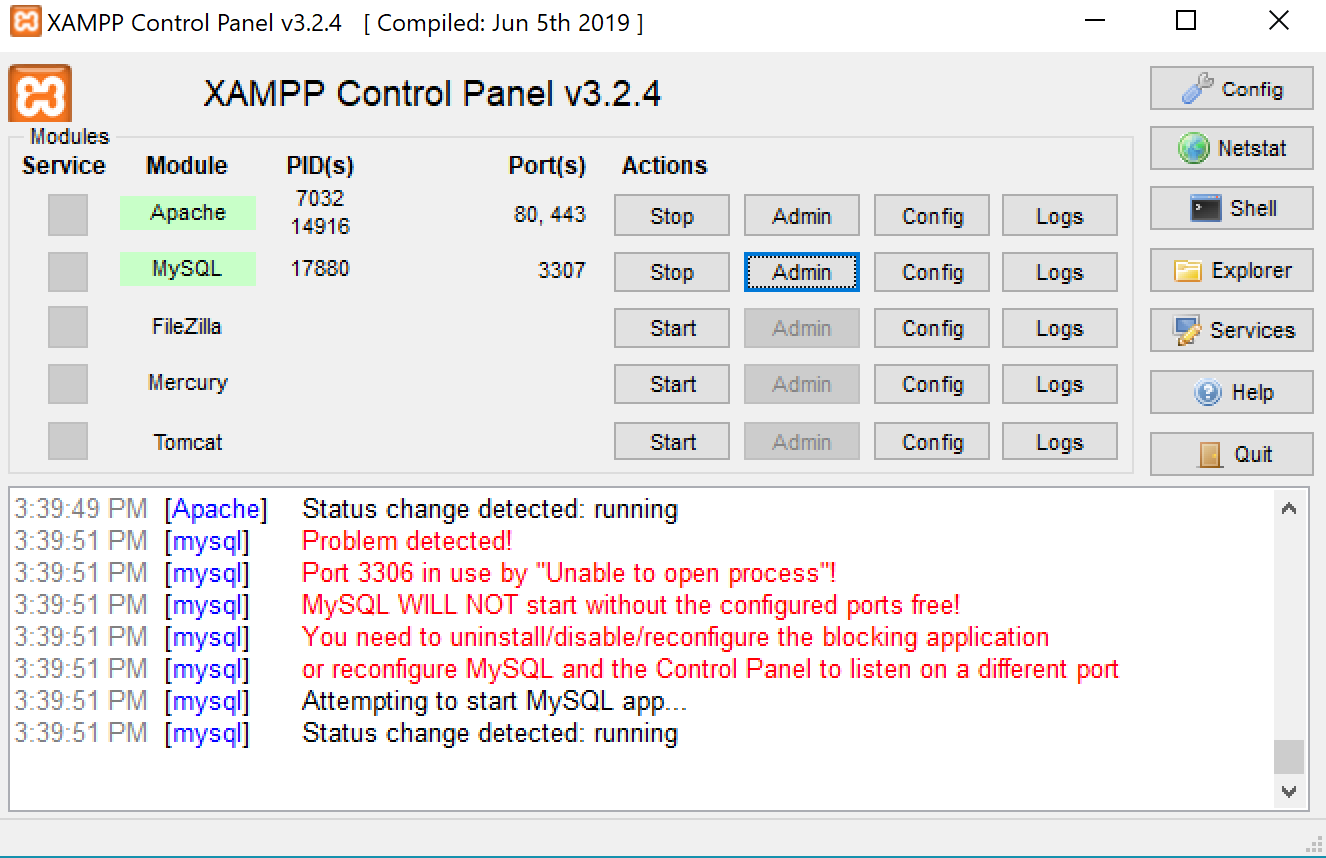


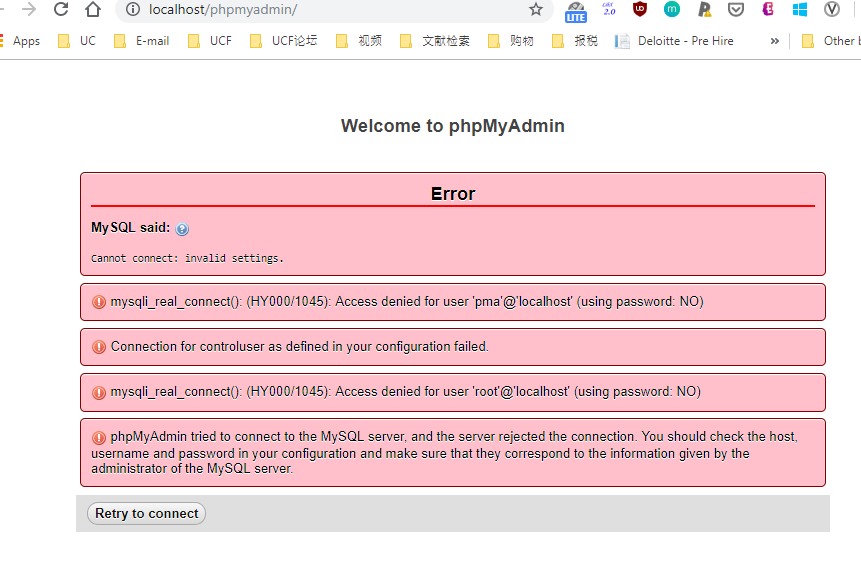
# Install Mysqlclient

$ pip install mysqlclient

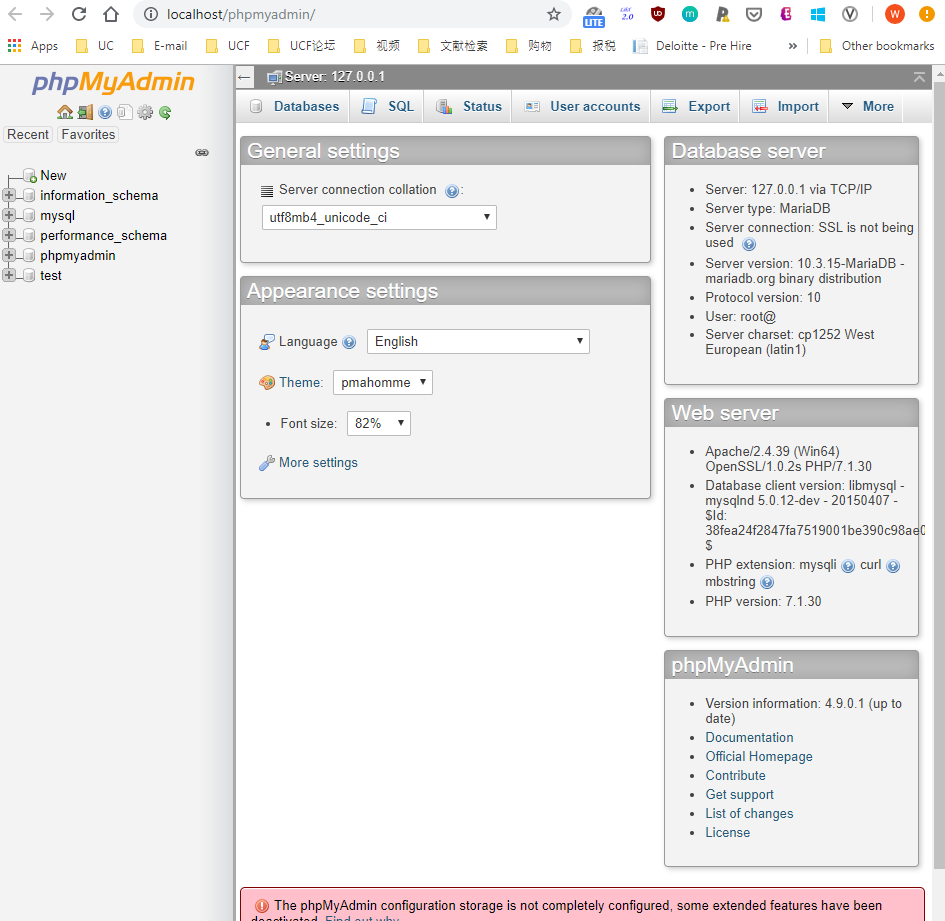


# go to setting.py and comment out the Database, and create the new Database part. The “Name” of the database should be create on website “localhost/phpmyadmin” ( Before type “localhost/phpMyAdmin/”, we should install XAMPP and start Apache and MySQL. Then Click the Admin button next to the MySQL. If there an error pops up, we should click the Config button of the MySQL and change all 3306 to 3307 and at the same time change all 3306 to 3307 in the “php.ini” which can be found in XAMPP/php/php.ini.)

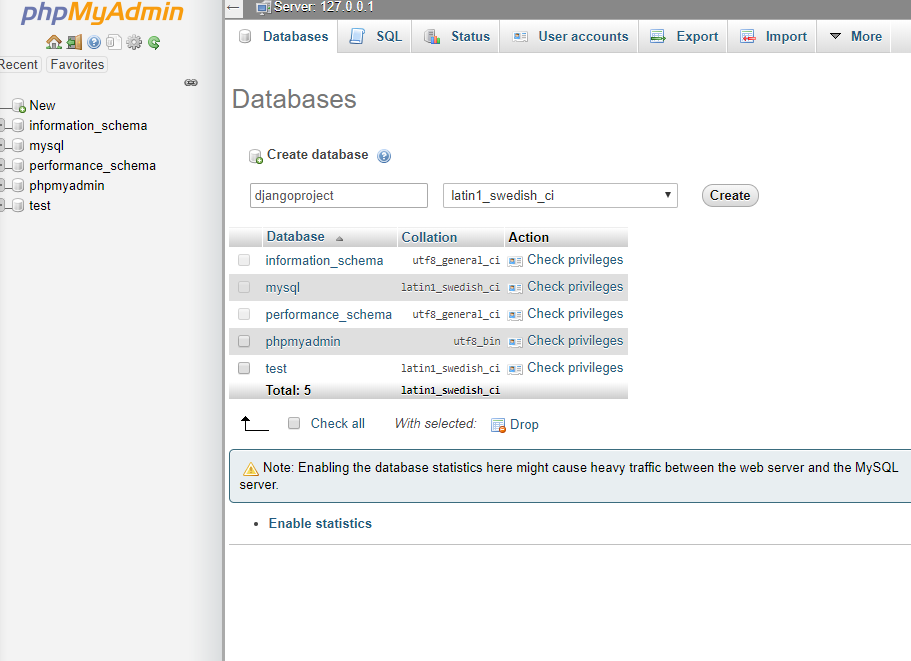


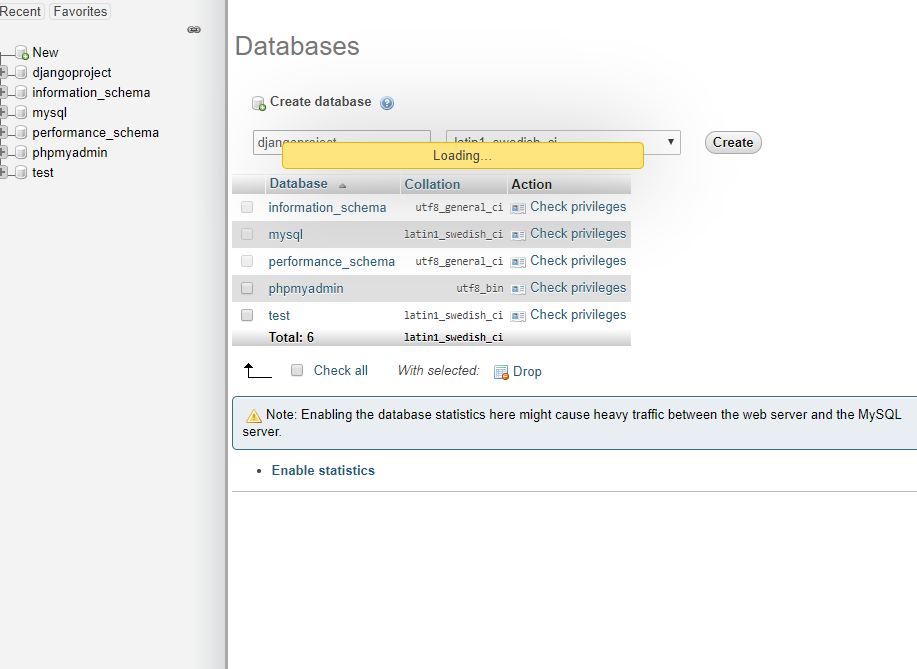




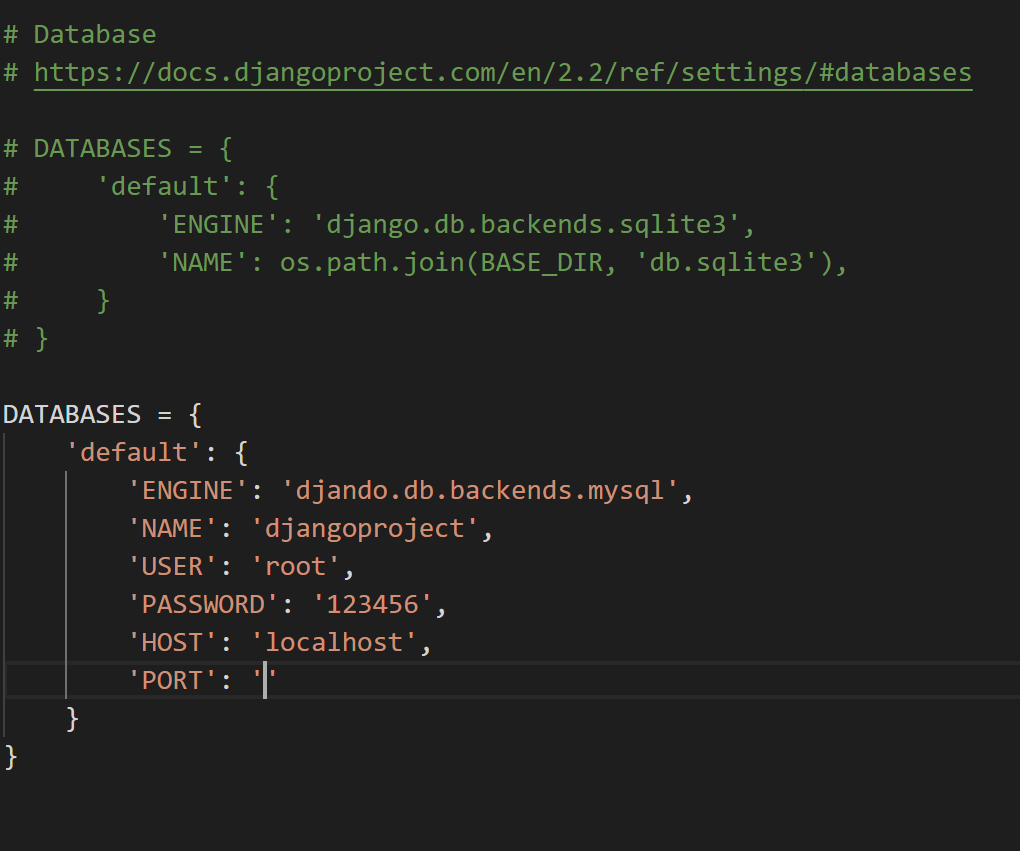


# Create djangoproject in Database and click “Create”

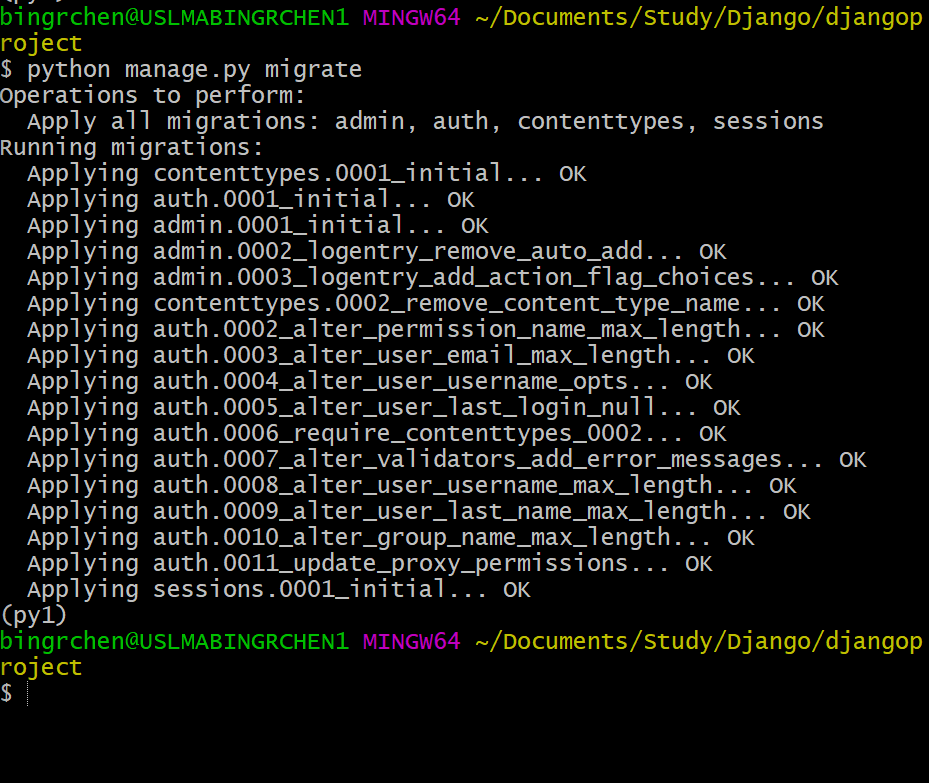




# Fill DATABASE



$ python manage.py migrate



$ python manage.py runserver

# Ctrl +C close server

$ python manage.py createsuperuser --username=bing [--email=bing@gmail.com](mailto:--email=bing@gmail.com)

# fill password

# run server again

$ python manage.py runserver