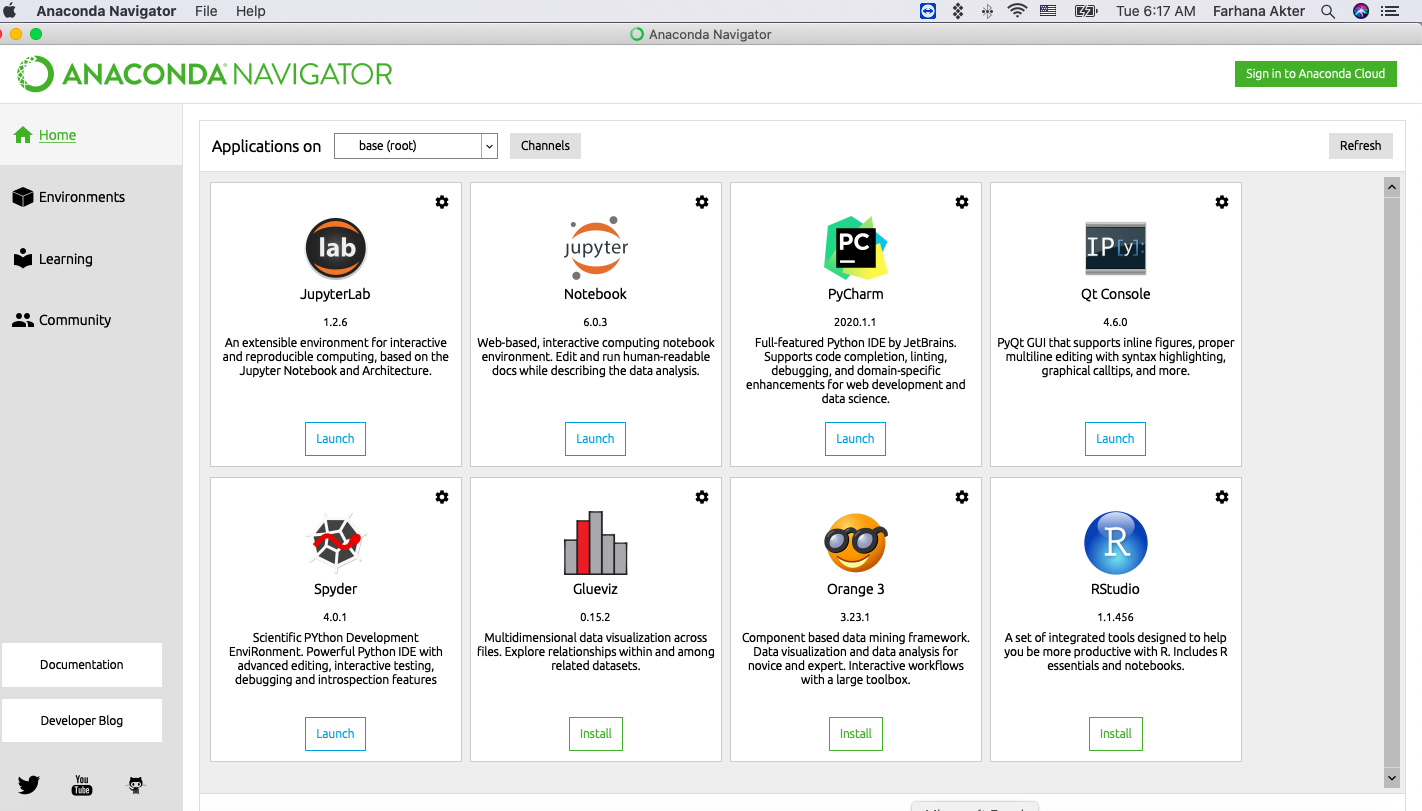
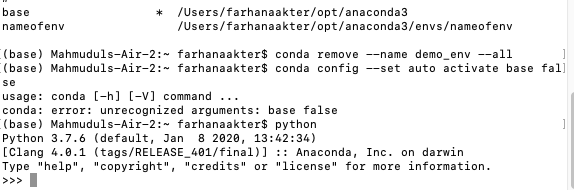
**DAY -10 ESTIMATE (1-2 hours)**

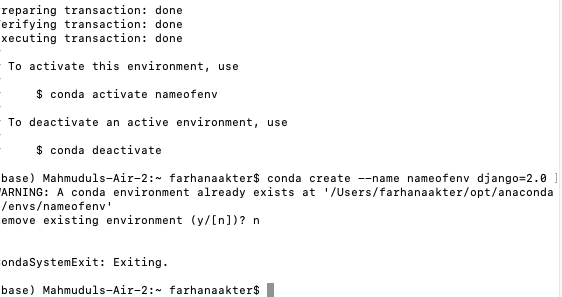
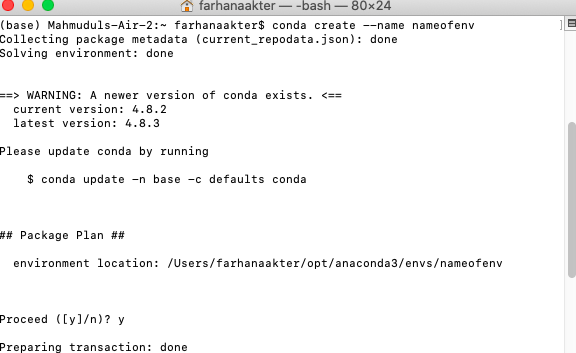
**TASK 01:**

* Create **virtual environment** using
  + Anaconda
  + Virtualenv
  + venv

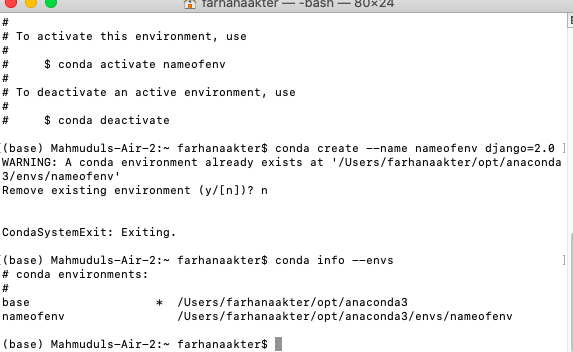




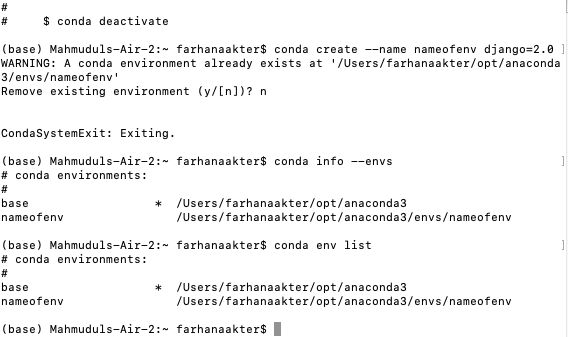
* Using **Anaconda** Package Manager:
  + Check if installed or not:
    - $ conda --version
  + Create Virtual Environment with or without package installation:
    - $ conda create --name demo\_env
* $ conda create --name nameofenv
  + - $ conda create --name demo\_env django=2.0



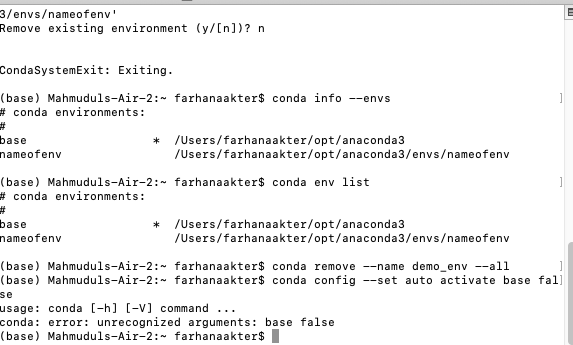
* + Activate Virtual Environment:
    - $ conda activate demo\_env
    - $ pip freeze
  + Deactivate Virtual Environment:
    - $ conda deactivate
  + Check Environments available:
    - $ conda info --envs



* + - $ conda env list



* + Remove and delete all packages inside:
    - $ conda remove --name demo\_env --all
  + Conda Environment not be activated on startup:
    - $ conda config --set auto\_activate\_base false



Name them as **env\_conda**, **env\_virtual\_env** and **env\_venv** respectively.

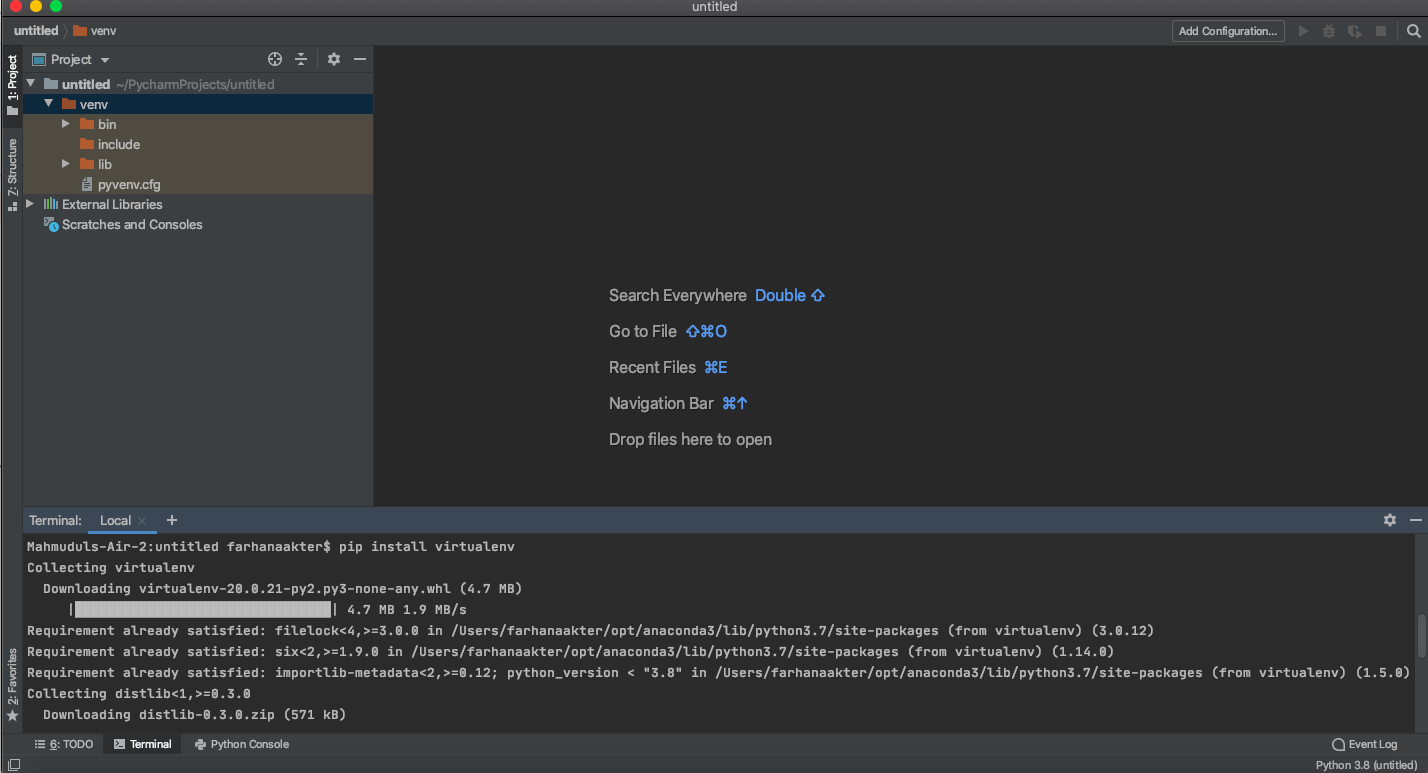
* Create a Git Repository with the name Virtual\_Env\_Demo
  + Make sure to have one file inside the repo with the name called requirements.txt
    - The content of requirements.txt is

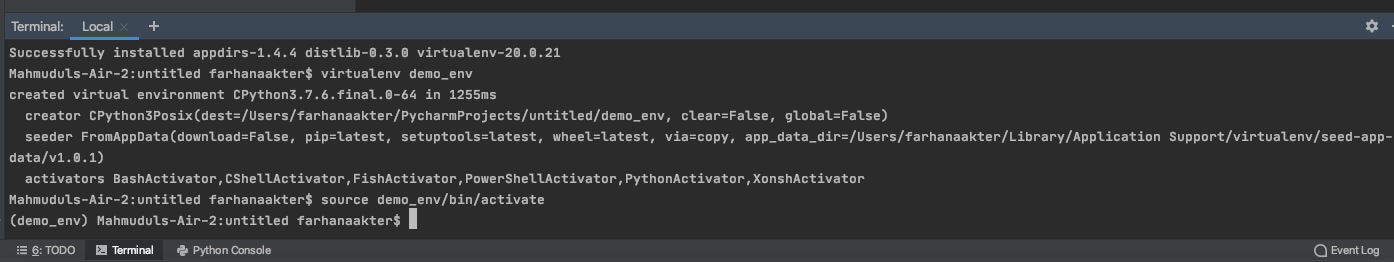
|  |
| --- |
| ordereddict==1.1  argparse==1.2.1  python-dateutil==2.2  matplotlib==1.3.1  nose==1.3.0  numpy==1.8.0  pymongo==3.3.0 |

* + Clone the repository on EC2 Machine.
  + Make sure to create a virtual environment on EC2 using virtualenv and name it as Virtual\_on\_ec2
  + Activate the virtual environment
  + Install all the dependencies from cloned requirements.txt
  + do **PIP FREEZE**
  + **Share the Screenshot :)**
* Using **Virtualenv** Package Manager:

<https://virtualenv.pypa.io/en/latest/>

* + Install Virtual Env:
    - $ pip install virtualenv
  + Create Virtual Environment:
    - $ virtualenv demo\_env
  + Activate Virtual Environment:
    - $ source demo\_env/bin/activate
  + Deactivate Virtual Environment:
    - $ deactivate





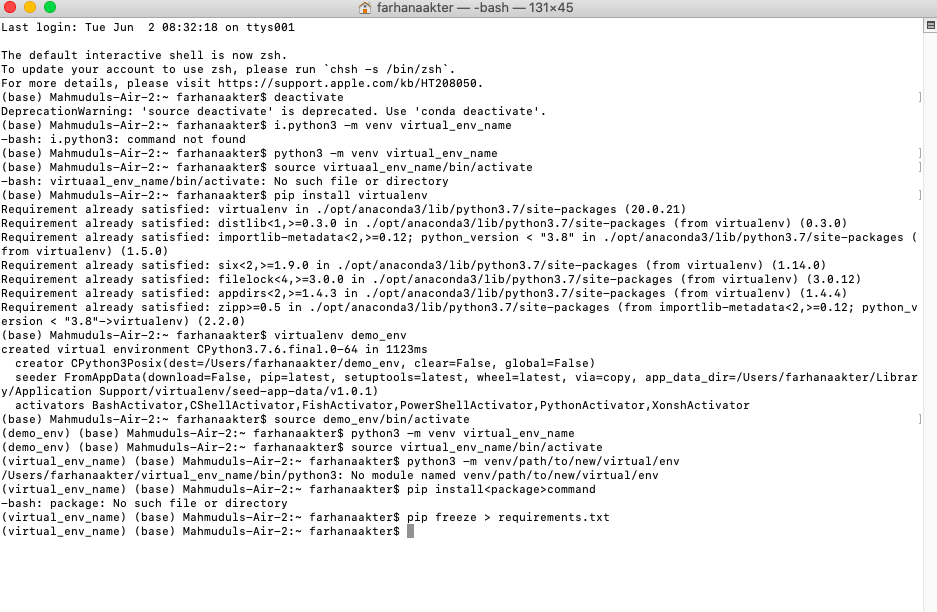
* Using **venv** Package Manager:

[**https://docs.python.org/3/tutorial/venv.html**](https://docs.python.org/3/tutorial/venv.html)

* 1. Create Virtual Environment:
     1. $ python3 -m venv virtual\_env\_name
  2. Activate Virtual Environment:
     1. $ source virtual\_env\_name/bin/activate - MacOS/Linux
     2. $ virtual\_env\_name\Scripts\activate.bat - Windows
  3. Deactivate Virtual Environment:
     1. $ deactivate

**Typical way of managing project dependency**

* Actions will be similar to the one below:
* Create a virtual environment $ python3 -m venv /path/to/new/virtual/env
* Install packages using $pip install <package> command
* Save all the packages in the file with
  + **$ pip freeze > requirements.txt**
* Keep in mind that in this case, requirements.txt file will list all packages that have been installed in virtual environment, regardless of where they came from
* Pin all the package versions. You should be pinning your dependencies, meaning every package should have a fixed version.
* Add requirements.txt to the **root** directory of the project. Done.



**IMPORTANT:**

|  |
| --- |
| Pin Explicitly  Don’t ever use these styles in requirements.txt:   * lxml * lxml>=2.2.0 * lxml>=2.2.0,<2.3.0   Instead, pin them:   * lxml==2.3.4   If you don’t, you can never know what you’ll get when you run pip install. Even if you rebuild the env every time, you still can’t predict it. |

**EXAMPLE:**

**requirements.txt**

**TASK 02:**

* Learn About STATIC AND DYNAMIC WEBSITE

|  |
| --- |
| novas==3.1.1.3  numpy==1.9.2  requests==2.7.0  pandas=1.0.1  matplotlib |

**$ pip install -r requirements.txt**