Virtualenv Instructions

Second Semester, 2022

1 virtualenv

1.1 Instructions for Quick Start

The quickest way to get your virtualenv up and running, so you can focus on your project, is as follows:

- Install virtualenv. It is already installed on cs computers, but if you're working outside then install using
  $ pip install virtualenv
- Go to the directory where your project files are at and create a virtualenv with the name of your choice (e.g. candy)
  $ virtualenv "candy"
- To use this virtual environment in your terminal session type:
  $ source candy/bin/activate
- When the environment is active you can install packages in it (e.g. tensorflow) as usual:
  $ pip install tensorflow
  These packages are installed only for the virtual environment, away from the global python installation. To see if the package you need is available without explicitly installing it, use
  $ pip list. If it is in the list, no need to install it.
- To stop using the currently active virtualenv simply type:
  $ deactivate
- When you finish the project, make sure the environment is active and type: **$ pip freeze requirements.txt** This will create a file called requirements.txt with all the specifics of the environment. You must submit this file, this enables us to run your code without errors due to missing packages/version inconsistencies.

If you run into trouble or would like to know how to do a few more things with virtualenv (e.g. using the requirements.txt file to install packages your partner has in his virtualenv), please consult us or the detailed instructions below.

2 Detailed Instructions

In order to give you as much freedom as possible, in such a way that we would still be able to test your results, we have decided to use virtualenv. This is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executable to use the packages that a Python project would need. Virtualenv is already installed on all cs computers and is very easy to use.

2.1 How to create a virtual environment

First, create a specific folder (with enough memory) for the project:
$ mkdir "my_project_folder"
Go to this folder:
$ cd "my_project_folder"
Secondly, initiate virtualenv by typing:
$ virtualenv "my_project"
virtualenv my_project will create a folder in the current directory which will contain the Python executable files, and a copy of the pip library which you can use to install other packages. The name of the virtual environment (in this case, it was my_project) can be anything; omitting the name will place the files in the current directory instead.
This creates a copy of Python in whichever directory you ran the command in, placing it in a folder named my_project.

2.2 How to activate and install packages

To begin using the virtual environment, it needs to be activated in the following manner:
$ source my_project/bin/activate
The name of the current virtual environment will now appear on the left of the prompt (e.g. (my_project)Your-Computer:your_project UserName$) to let you know that it's active. From now on, any package that you install using pip will be placed in the my_project folder, isolated from the global Python installation. Install packages as usual, for example:
$ pip install tensorflow
If you are done working in the virtual environment for the moment, you can deactivate it:
$ deactivate
This puts you back to the system's default Python interpreter. To delete a virtual environment, just delete its folder. (In this case, it would be rm -rf my_project.) One of the most important tool of virtualenv is the ability to "freeze" the current state of the environment packages. To do this, run:

$$\$pip\ freeze > requirements.txt$$

This will create a requirements.txt file, which contains a simple list of all the packages in the current environment, and their respective versions. You can see the list of installed packages without the requirements format using "pip list". The opposite of "freezing", is installing packages in a new environment based on requirements.txt. This can be done using the command:
$ pip install -r requirements.txt

All groups are required to submit a requirements.txt file. Your code must run properly with the environment created by your requirements.txt file on the CS computers.

2.3 How to configure PyCharm to work with virtualenv

Open PyCharm:
$ pycharm
Select File, click Settings.
In the left pane, enter Project Interpreter in the search box, then click Project Interpreter.
In the right pane, click the gear icon, click More...
In the Project Interpreters dialog box, click the plus sign +, click Add Local. Enter

$$< full\_path > / < my\_project > /bin/python$$

in the path. Click Apply, click OK
You are set to go.
If you are using other python editors (eclipse, spyder, and so on), information regarding configuration with virtualenv can be found online.

2.4 Important Notes

Running virtualenv with the option –no-site-packages will not include the packages that are installed globally. This can be useful for keeping the package list clean in case it needs to be accessed later. [This is the default behavior for virtualenv 1.7 and later.]
Excluding the virtual environment folder from source control by adding it to the ignore list is highly recommended. If you are using your own laptop, you can Install virtualenv via pip:
$ pip install virtualenv
You can test your installation via:
$ virtualenv –version
You can use virtualenvwrapper if you want to, it is even installed on the CS computers. You can use any IDE (such as eclipse, spyder an so on..), however you have to make sure that your code works on the CS computers. If you encounter memory issues please consult this link.