Setting Up Your Python Environment with Pip, VirtualEnv and PyCharm (Windows)

This tutorial will cover setting up a Python Environment from scratch on a Windows Machine.

Install Python

You are welcome to use either Python 2 or Python 3, but for the sake of this tutorial we are using Python 2.7.

Download Python 2.7 for Windows from the official Python website here. Once the download is done, click on the package and follow along with the install wizard. If you are given a choice for where to install Python, chose C:\Python27

Open the command line
Search your apps for the application “cmd”. This is the command line tool.

Locate your Python installation on the command line
Mine is at C:\Python27

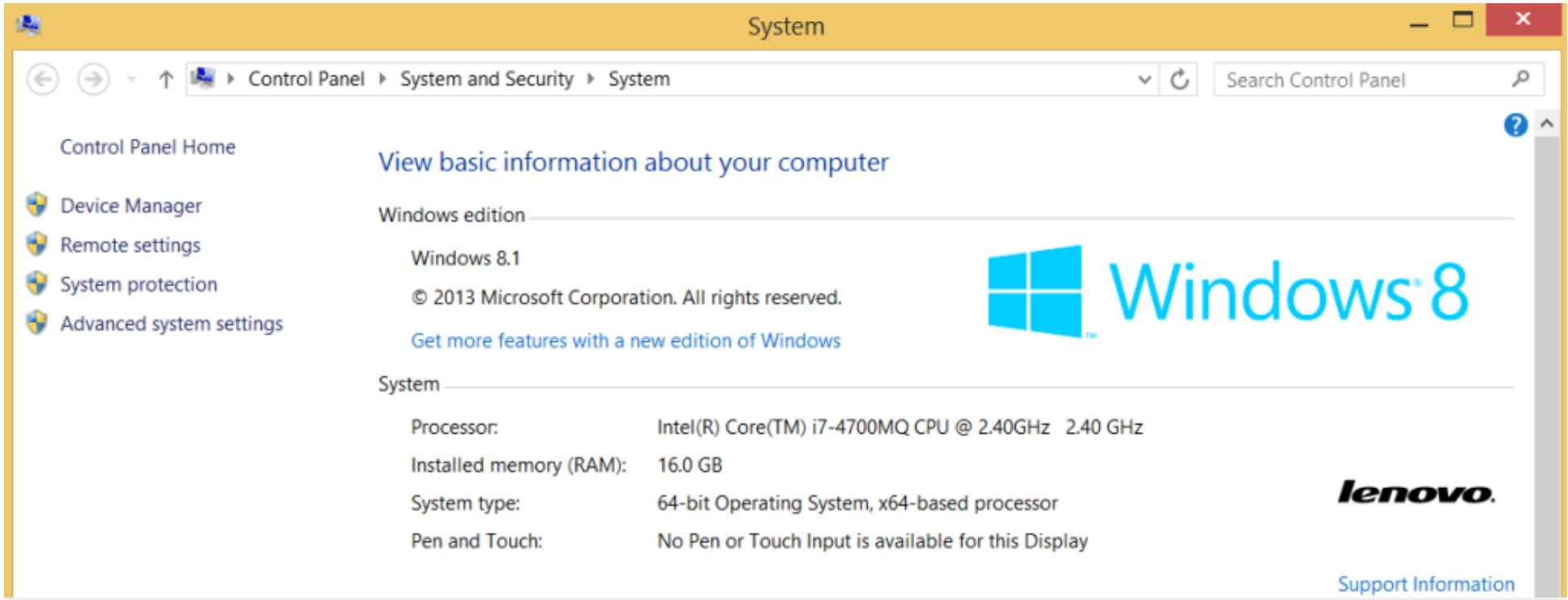
```
C:\Users\michelle>cd C:\Python27

C:\Python27>
```

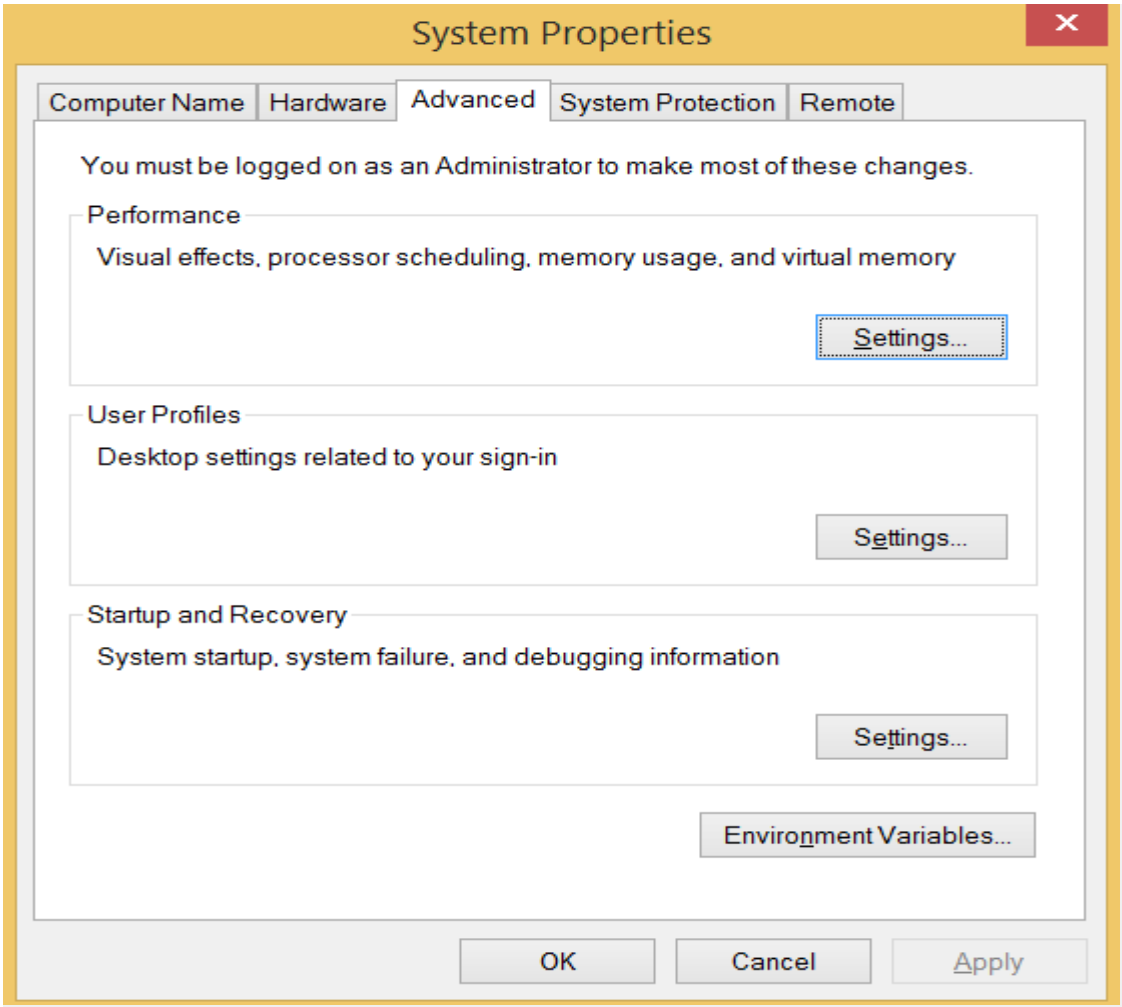
Update your Environment Variables

Environment variables tell your system where to find things on your machine, including software that you have installed. We are going to create an environment variable for Python’s directories so we can access it quickly and easily. This will enable us to run python by just typing “python”, instead of having to type “C:\Python27\python” all the time.

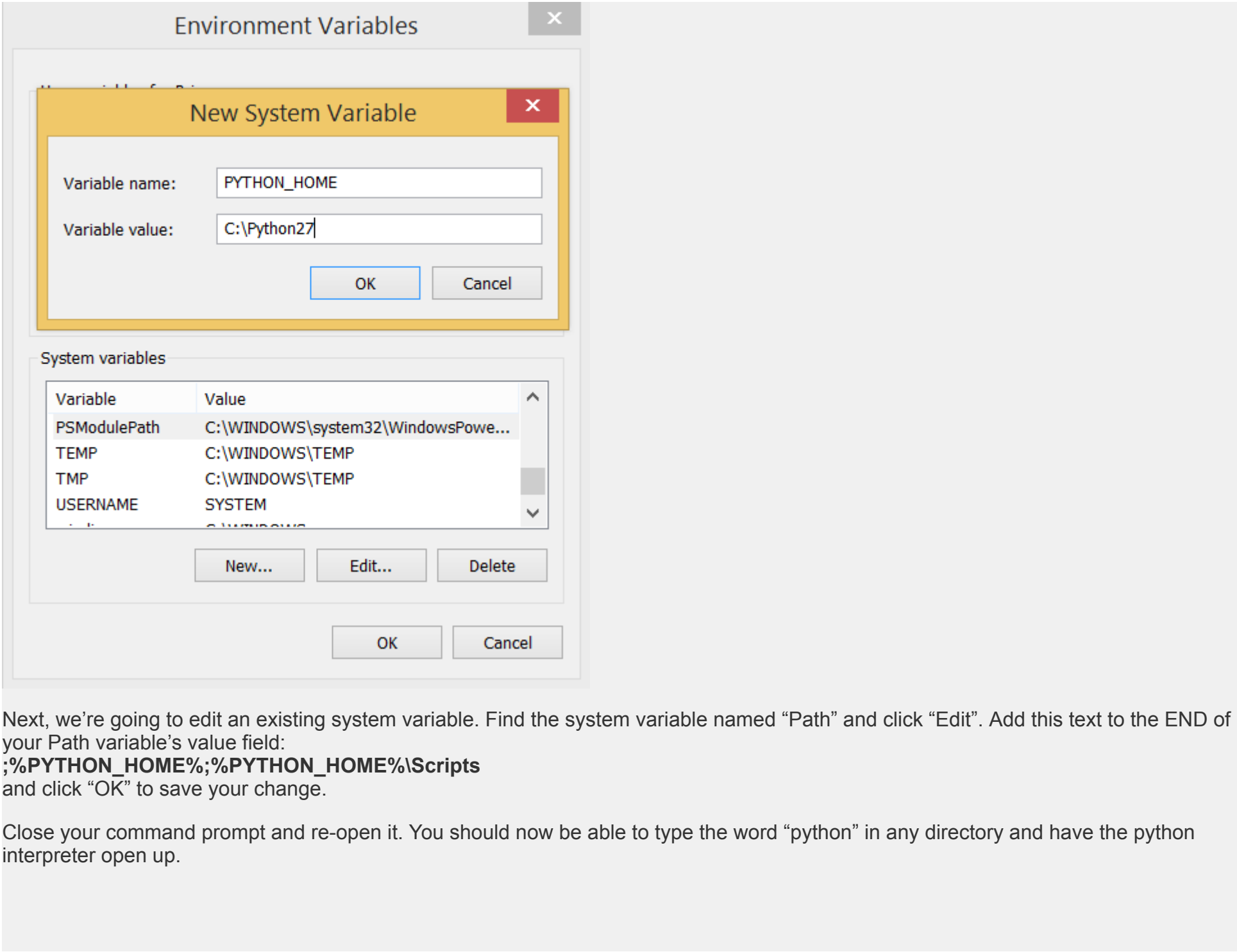
Search your machine for “System” to get to the System control panel. Click on “Advanced System Settings” in the left-hand menu.



On the “Advanced” tab, click on “Environment Variables” in the lower right-hand corner.



In the **lower** box titled **System Variables**, click “New” to create a new system variable. Name it “PYTHON_HOME” and give it the value that corresponds to where YOUR python install is. Click “OK” to save.



Next, we're going to edit an existing system variable. Find the system variable named "Path" and click "Edit". Add this text to the END of your Path variable's value field:
;%PYTHON_HOME%;%PYTHON_HOME%\Scripts
and click "OK" to save your change.

Close your command prompt and re-open it. You should now be able to type the word "python" in any directory and have the python interpreter open up.

```
C:\Python27>cd \Users\michelle

C:\Users\michelle>python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:32:19) [MSC v.1500 32 bit (
Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type **ctrl+c** to close the python interpreter.

Install Pip

Pip is a package manager. A package manager is a tool that is used to keep your third-party libraries managed in a simple and orderly way on your machine.

Go to https://bootstrap.pypa.io/get-pip.py to download the script **get-pip.py**. The website should prompt you to save the file. Save it to your Downloads directory.

On the command line, navigate to your Downloads directory and use python to run the get-pip.py script.

```
C:\Users\michelle>cd Downloads

C:\Users\michelle\Downloads>python get-pip.py
```

Navigate to your Python “Scripts” directory

```
C:\Users\michelle\Downloads>cd C:\Python27\Scripts
```

And type “dir” to view the files inside this directory. You should see several new pip executables, which look like this:

```
06/21/2016  09:15 PM            89,420 pip.exe
06/21/2016  09:15 PM            89,420 pip2.7.exe
06/21/2016  09:15 PM            89,420 pip2.exe
```

Close the command line and re-open it.

To make sure that pip is installed correctly and your environment variables are set up to tell your computer where to look for python and it’s tools, type **pip freeze**.

```
C:\Users\michelle>pip freeze
```

If you see a list of contents or nothing at all, that's good! That means that you successfully ran the pip command. If you see a bunch of errors, that's bad.

FYI: The command we just ran, **pip freeze** is how you ask pip to show you all of the packages that you have installed. That's why it's ok if you saw nothing when you ran the command. That just means you haven't installed any packages yet.

Congrats! We now have the ability to easily install third-party packages.

Install VirtualEnvWrapper

Even though we won't do it today, it is common to work on many python projects at the same time. You might need to install a lot of packages for one project, but only need a few packages for another. Even worse, you might need one **version** of a package for one project, and a **different version** for another.

This could potentially be a real headache. Take this example:
You have project CatsBlogs which is using the package Django version 1.5. You have another project, DogsBlog, which is on Django 1.8. Every time you want to work on CatsBlog, you have to downgrade Django to 1.5… but then if you want to work on DogsBlog, you have to upgrade it back to 1.8! Ugghhhh.

Fortunately, we have a solution to this problem, and that solution is virtual environments. A virtual environment will keep all of your packages for each project isolated from each other, allowing you to install multiple versions of multiple packages on your computer at the same time, without them colliding with each other.

Let's install our virtual environment software, and we'll go through an example.

Use pip, our handy package manager, to download the virtualenvwrapper-win package.

```
C:\Users\michelle>pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
  Downloading virtualenvwrapper-win-1.2.1.zip
Requirement already satisfied (use --upgrade to upgrade): virtualenv in c:\pytho
n27\lib\site-packages (from virtualenvwrapper-win)
Building wheels for collected packages: virtualenvwrapper-win
  Running setup.py bdist_wheel for virtualenvwrapper-win ... done
  Stored in directory: C:\Users\michelle\AppData\Local\pip\Cache\wheels\97\87\19
\3c63b1815e3151970c1cb0be616f4ba0f9a2f3436bc0003839
Successfully built virtualenvwrapper-win
Installing collected packages: virtualenvwrapper-win
Successfully installed virtualenvwrapper-win-1.2.1
```

Next, use our new package to create a new virtual environment for this tutorial. I'm going to name my virtual environment "myenv".

```
C:\Users\michelle>mkvirtualenv myenv
New python executable in C:\Users\michelle\Envs\myenv\Scripts\python.exe
Installing setuptools, pip, wheel...done.

(myenv) C:\Users\michelle>
```

Notice how my command line now has **(myenv)** in front of the C:\Users\michelle>. This is the command line telling me that I am now inside of my “myenv” virtual environment.

Also note that your output told you where your **new python executable** is. Mine looks like this: C:\Users\michelle\Envs\myenv\Scripts\python.exe

This path is very important and we will need it later. So take a mental note of where it is.

Now that we have a virtual environment, let’s look around a bit to help us get comfortable with how it works. While we’re here inside the virtual environment, let’s install another package with pip. We’ll install the package **tweepy**.

```
(myenv) C:\Users\michelle>pip install tweepy
```

Now do a pip freeze. Notice that tweepy is in there now!

```
(myenv) C:\Users\michelle>pip freeze
oauthlib==1.1.2
requests==2.10.0
requests-oauthlib==0.6.1
six==1.10.0
tweepy==3.5.0
```

Cool. Also notice that I have more packages than you probably do. That’s ok. You only need to have tweepy to be successful at this point.

Now turn your virtual environment off.

```
(myenv) C:\Users\michelle>deactivate myenv

C:\Users\michelle>
```

Notice that the “(myenv)” has disappeared from your command line. This means you are no longer inside of your virtual environment.

Now type “pip freeze”. Notice that tweepy is gone! This is because we are now **outside** of the virtual environment. Tweepy is installed **only inside** our virtual environment.

Now turn your virtual environment back on.

```
C:\Users\michelle>workon myenv
(myenv) C:\Users\michelle>
```

Type “pip freeze” again from inside of the virtual environment and see that tweepy is back again!

Congrats again! We now have the ability to keep our packages contained and safe for each project we make.

Setting up PyCharm

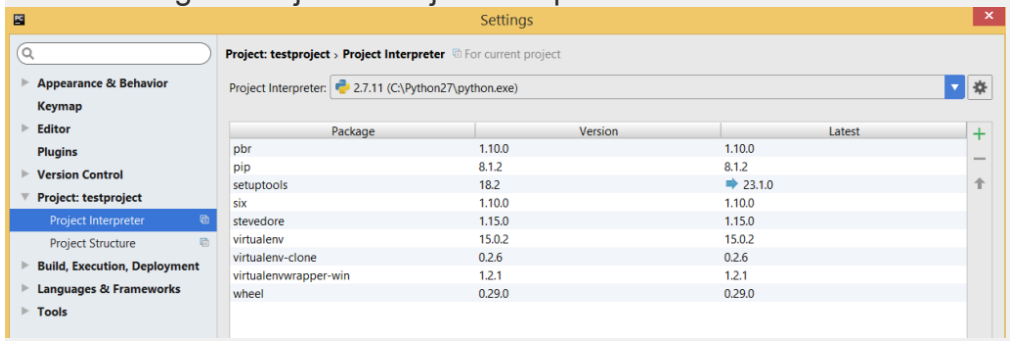
PyCharm is an Integrated Development Environment, which is used to write python code.

If you don’t have PyCharm yet, download the free Community edition here.

Our first order of business is to get PyCharm set up to use our new virtual environment, so that we can use the packages that we install with pip.

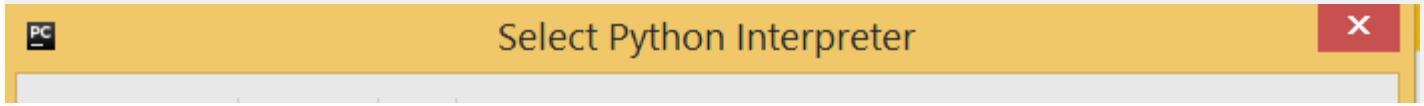
Open up PyCharm and create a new project.

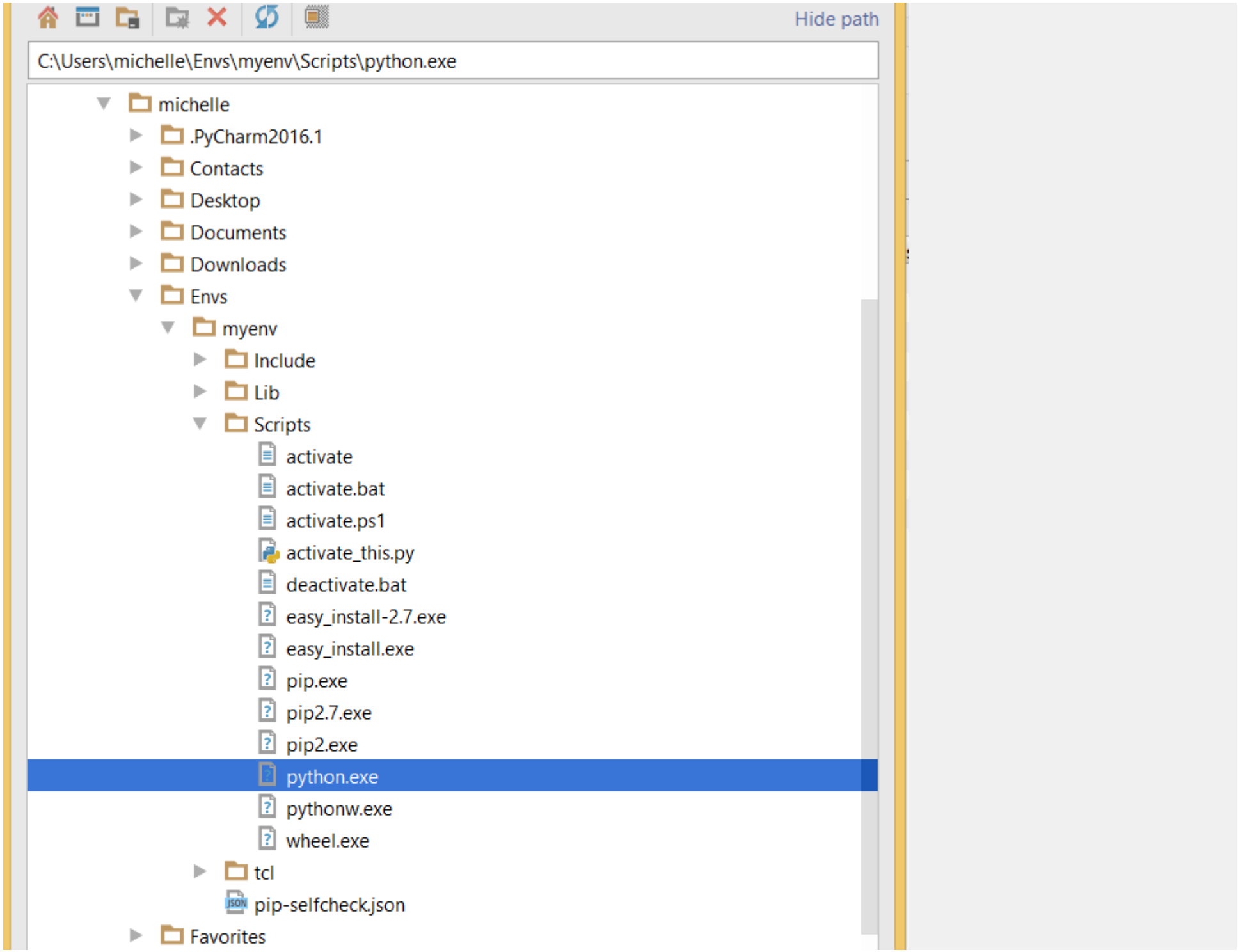
Navigate to:
File > Settings > Project > Project Interpreter

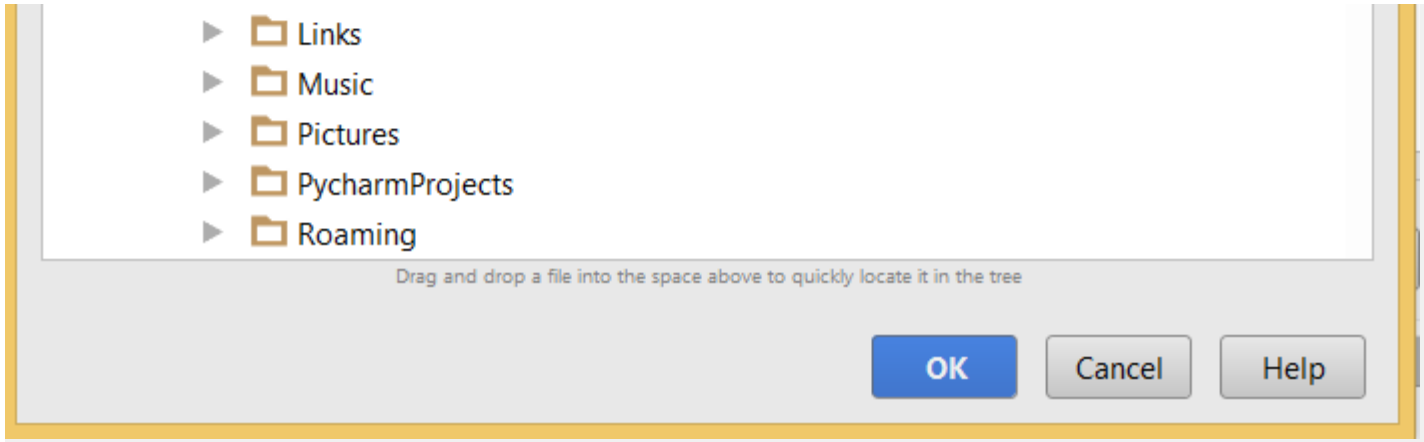


Remember earlier when I told you to remember where the python executable for your virtual environment is? Mine was: C:\Users\michelle\Envs\myenv\Scripts\python.exe

Click the gear icon in the top-right corner. Select “Add Local”. This is going to let us point PyCharm to the version of Python that is inside of our virtual environment.

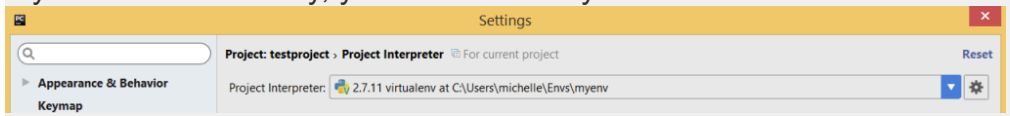



Hide path
C:\Users\michelle\Envs\myenv\Scripts\python.exe
michelle
.PyCharm2016.1
Contacts
Desktop
Documents
Downloads
Envs
myenv
Include
Lib
Scripts
activate
activate.bat
activate.ps1
activate_this.py
deactivate.bat
easy_install-2.7.exe
easy_install.exe
pip.exe
pip2.7.exe
pip2.exe
python.exe
pythonw.exe
wheel.exe
tcl
pip-selfcheck.json
Favorites



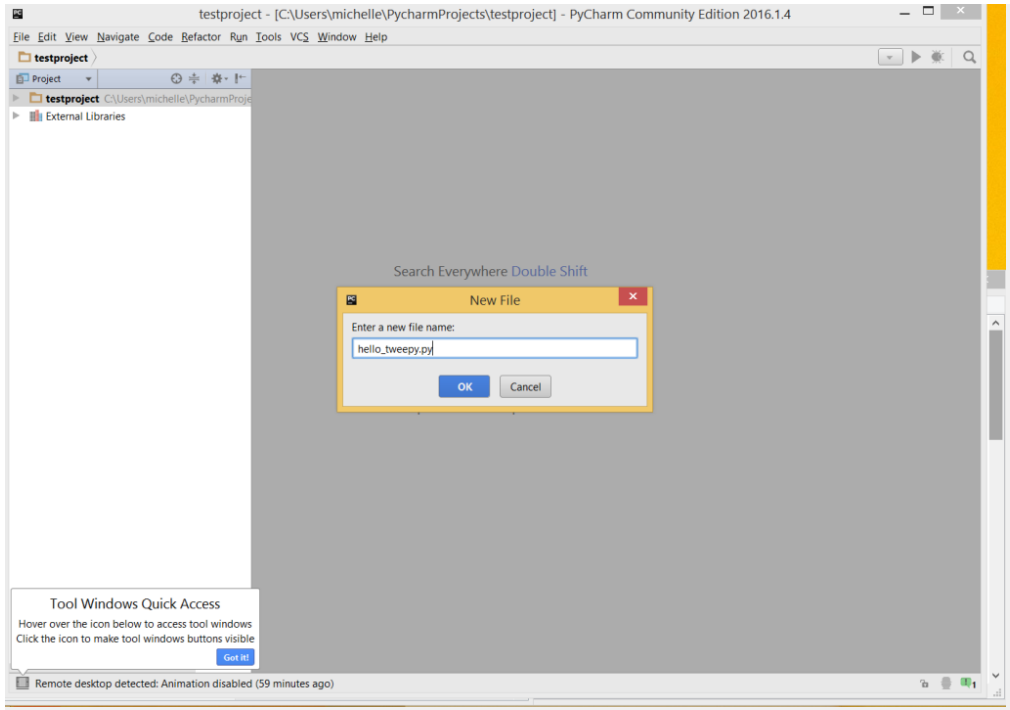
Select your Python from inside your virtual environment and click “OK”.

If you did this correctly, you will now see your virtual environment selected in the Python Interpreter window.

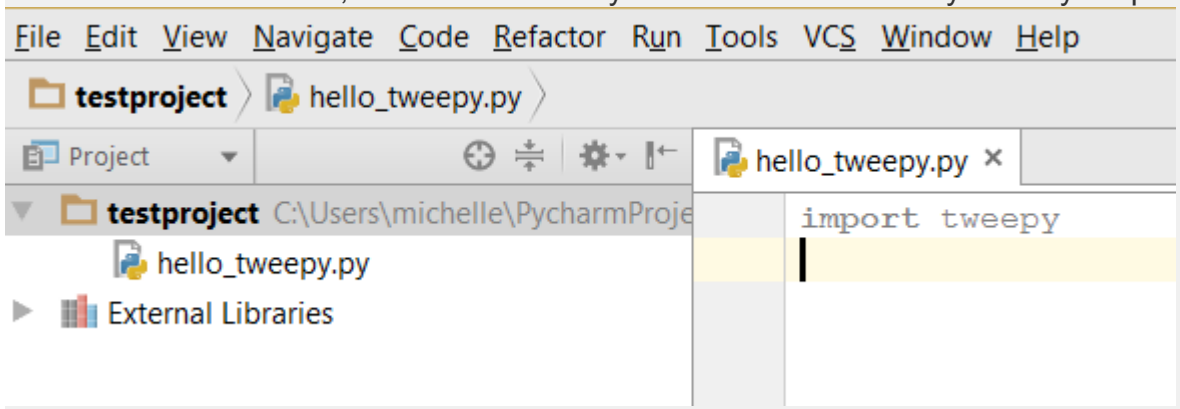


Now, let’s test and make sure that we can access our python packages from inside of PyCharm! Remember that the package that we installed was called tweepy. Let’s try to import it in PyCharm.

In PyCharm, right-click the name of your project (mine is “testproject” in the screenshot) and choose “New > Python File”. Name it hello_tweepy.py



Just to test that our package is available to us, type “import tweepy”. If PyCharm underlines your import statement with red squiggles, that means that PyCharm cannot find the package, and your setup was not successful. If PyCharm greys out your import statement like in the screenshot below, that means that PyCharm has successfully found your package and is now just waiting to use it!



Congrats! You can now access you packages from inside of PyCharm.