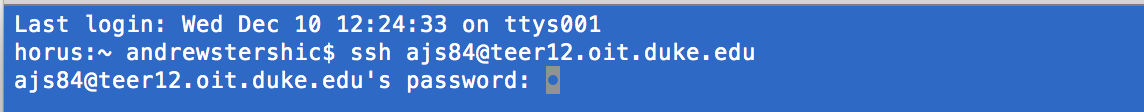
Code installation notes:

1. Access web server using SSH. This can be done from Linux/Mac using “Terminal” or from Windows using an SSH client (like “PuTTY” – free download). The format of the access command will be “ssh username@webserver.com” like below. You will be prompted for the password.



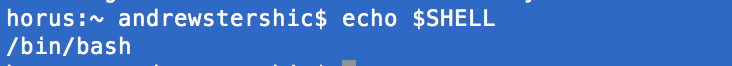
If successful, you will see that the host-name on the left-hand side of the terminal has changed to be your username at the server.

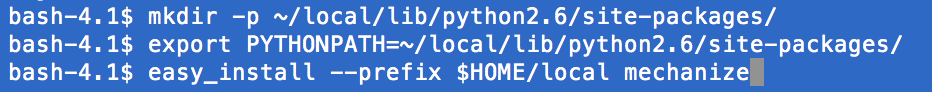
Macintosh HD:Users:andrewstershic:Desktop:Screen Shot 2014-12-10 at 10.28.30 PM.png

2. Next you need to check if python exists. Asking python’s version should give a logical result if it exists. Otherwise, install Python 2.x.x from <https://www.python.org/downloads/> .

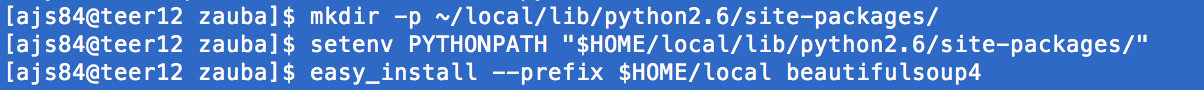


3. (harder) You need to get the python modules (add-ons) “mechanize” and beautifulsoup4 installed. See if you’re in a bash shell:



If you are, then do the following commands to install “mechanize” and “beautifulsoup4”. Be careful to use the version of python you saw in step 2 (2 and the next digit; mine was 2.6.6 so use 2.6).Macintosh HD:Users:andrewstershic:Desktop:Screen Shot 2014-12-10 at 11.14.02 PM.png

If you’re in a tcsh shell instead, it’s just slightly different:

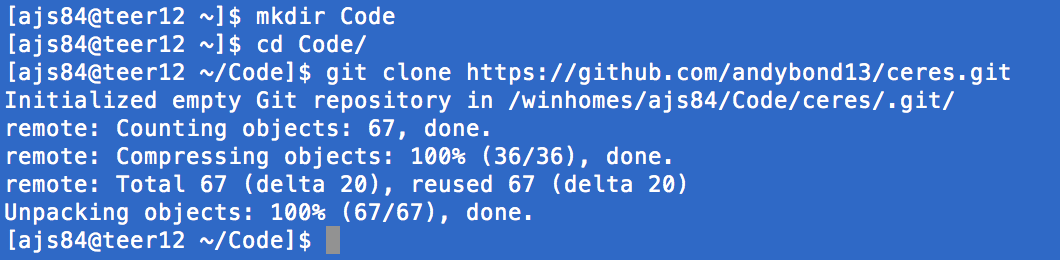


Macintosh HD:Users:andrewstershic:Desktop:Screen Shot 2014-12-10 at 11.34.45 PM.png

In any other shell, just type “bash” to get into a bash shell and follow the bash instructions.

4. The best way to actually get the code onto the webserver is using a program called ‘git’ which comes installed on most Linux/Mac systems (and thus, most webservers). You can check for it by asking its version:

Macintosh HD:Users:andrewstershic:Desktop:Screen Shot 2014-12-10 at 10.36.10 PM.png

You can then grab the code by using the git “clone” command. The code is located at <https://github.com/andybond13/ceres.git> . You might first want to change directories (“cd” : switch folders) or make a folder (“mkdir”) for the code. 

I’m not sure if it will ask or not, but the git account to download this is:

Username: DisiKrishiStar

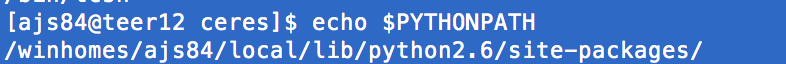
Password: T0matoes

So now, looking at my example, I now have a folder called “Code” in my home directory (represented by the squiggle “~”; in this case “/winhomes/ajs84/” is my home), and I now have a folder called “ceres” in the Code folder which contains all the code. We can now enter the “ceres” directory (“cd ceres”) and look around (“ls”).

There are three web-scraping scripts “commerce.py” in the “commerce” folder, “nhb.py” in the “nhb” folder, and “zauba.py” in the “zauba” folder which each make results in .csv files in their respective folders.

5. The code can be run manually by running the script directly using python (“python scriptname.py”). If you’ve just installed the packages (step 3) and so you have PYTHONPATH defined (you can check by running “echo $PYTHONPATH”) then you can do this immediately. Otherwise, if you’ve logged out after step 3, you will need to do step 6 first.

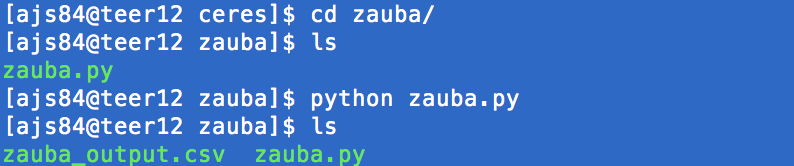
This means it’s defined and you’re good:

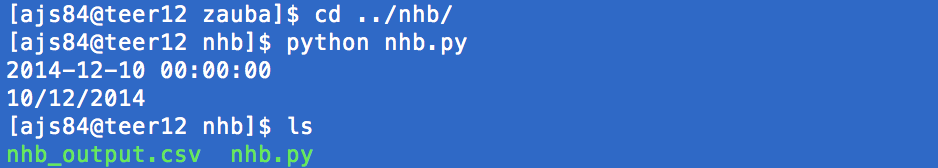


An empty response means it’s not defined, so you should do step 6 first:

Macintosh HD:Users:andrewstershic:Desktop:Screen Shot 2014-12-11 at 8.25.03 AM.png

You have to be in the script’s folder to do this. Here’s zauba:



and you can switch to the nhb folder (“cd ../nhb”; “..” means back one folder) and run that:

6. (harder) Automation can be done by editing and then launching “setLocation.sh” in the “script” folder. This gets a little user-unfriendly, but on the bright side, it only needs to be done once and it’s good forever. What “setLocation.sh” does is three things: first, it replaces lines in the “crontab.bash and crontab.tcsh” folder to match your code location; second, it adds the Python module location permanently to the python path, so python knows where they are whenever you log in; third, it adds the appropriate crontab.(bash,tcsh) file to cron. Cron is a scheduling program, so by adding our crontab file to cron, it’s scheduling these scripts to run on the schedule indicated in the crontab file (daily).

The user doesn’t need to know how all this works, just to be able to edit setLocation.sh and run it. In order to edit the file, you must use a terminal text editor (like “vi”,”emacs”,”nano”); these are not fun, but there’s lots of notes on the internet for how to do it.

There are four places the user needs to edit things in setLocation.sh.

1. “export $ceres=\_\_\_\_\_\_\_\_” : here you must write the location of the code folder for this project. For me, this would be “export $ceres=~/Code/ceres”

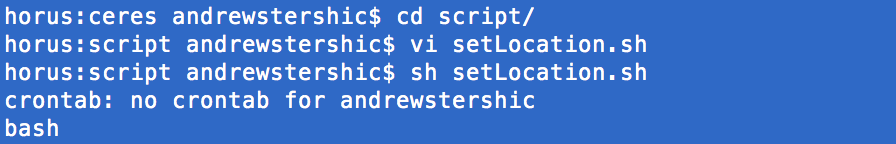
2. “perl –pi –e ‘s/\$ceres/\_\_\_\_\_\_\_/g’ crontab.tcsh” (and same for .bash) : here you must again write the location of the folder for this project **but** you have to replace the / slashes with \/. Mine looks like this “perl –pi –e

‘s/\$ceres/~\/Code\/ceres/g” crontab.tcsh . Do this for both lines (tcsh & bash).

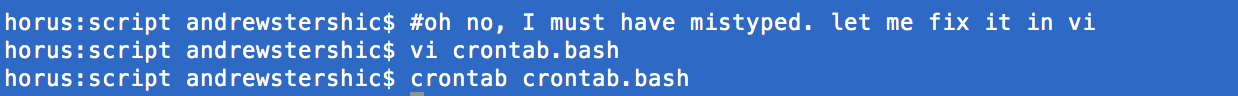
3. “export $PYP=\_\_\_\_\_\_\_\_” : here you must write the location of the Python modules folder (same folder that you made in step 3!). For me, this would be “export $PYP=~/local/lib/python2.6/site-packages”

4. “perl –pi –e ‘s/\$PYP/\_\_\_\_\_\_\_/g’ crontab.tcsh” (and same for .bash) : here you must write the location of the Python modules folder (same folder that you made in step 3!) **but** you have to replace the / slashes with \/. Mine looks like this “perl –pi –e ‘s/\$ceres/~\/local\/lib\/python2.6\/site-packages/g’ crontab.tcsh” . Do this for both lines (tcsh & bash).

You can then run setLocation.sh by typing “sh setLocation.sh”.



If this runs without error (like mine; “no crontab for …” is not an error), then you’re good – the code will run automatically at midnight daily, and you can also run it manually. If there’s an error, then you can look at the output for your shell type (mine is bash), and then debug the crontab.(bash,tcsh) file of that shell type (so crontab.bash for me). If you typed everything right, the lines should include the code directory and the Python module directory. You can then either run “sh setLocation.sh” or run “crontab” and then the crontab file name (“crontab crontab.bash” for me).



Script:

Just a quick explanation about each script.

**commerce.py**

This script gets information from <http://commerce.nic.in/eidb/icomq.asp>, and is set to get all the information from 2014-2015 (2014-2015 Apr-Sept.) for the product code 0702 (Tomatoes, fresh or chilled). You can change the year or the product code in the script (using vi, emacs, nano, etc.), save the file, and it’s good to go. You can find the other product codes by searching here: <http://commerce.nic.in/eidb/searchq.asp?fl=Icomq.asp> .

**nhb.py**

This script gets information from <http://nhb.gov.in/OnlineClient/categorywiseallvarietyreport.aspx>, and is set to get all the information from yesterday for Tomato Hybrid and Tomato Local. The date can be changed to a specific day by editing lines 44-46, but be sure to put the value in 2-digits for month and day and 4-digits for year **and** use quotes (like this: day = ‘23’ month = ‘05’ year = ‘2012’ ). Again, save the file and it should run. To change the product to something else, you’ll have to follow the code example for the new product. In this script, 15 and 16 refer to the column numbers in the output table (you can check this by hand) which begin at zero.

**zauba.py**

This script gets all the information from <https://www.zauba.com/export-tomato+puree-hs-code.html> and <https://www.zauba.com/import-tomato+puree-hs-code.html>. Since it gets all the information, there’s no date or product to worry about. If a different product was desired, you can find the right url on zauba.com, and then substitute that into this script.