Muntaha Pasha
04/30/2020
Rotation 4 - CSG

CSG Rotation (NCAR SARP)

I want to start off this report by explaining the purpose of all this documentation. I'm writing this report to recount my experience here in the CSG team of CISL and what I learned, as well as what I am taking away from this rotation! In this rotation, I worked with my mentor Mick Coady to modify and update a working PERL script into a Python script!

1. **Introduction to CSG**

The first thing I did in this rotation was meet the team! I was introduced to some of the people that really make the Consulting Services Group what it is. I learned what it is that CSG does, and how they always aim to create a good experience with users who may be having technical difficulties getting data, or issues with accessing the Cheyenne Supercomputer. They do a vast array of work to mediate these kinds of troubles and work at the lower level, while the CISL Sys Admins work at a higher level.

In this rotation, there wasn't really a website where I did my work. Right off the bat on day one I was given some time to explore some projects the CSG team had laying around, and my interest went to an old PERL script they had that they wanted to convert into Python. Python was a language I was quite familiar in, and in my NETS rotation I was working closely with PERL, so I figured it was the perfect project for me to dive into and complete. Though their PERL script functionality was fine, they wanted to be able to convert their script into something more manageable by the standards of new technology. Not many people entering the workforce today know much about PERL, so being able to convert the script to Python makes it easier for new people coming in to understand what the script does without needing to read up on a new language. It was also going to create some ease for the team itself, as most people in Consulting Services knew Python more than they knew PERL.

2. **Learning the Script**

Once my project idea was set in stone, it was time to open up the PERL script and see exactly what it did. It was called licstats, and it did exactly as the name implies. It gathered statistics on the various license data that NCAR has, for instance, if they have 50 matlab licenses and 6 are in use, it would report that. Or if NCAR had a license for Intel, it would report how many of those licenses are available and how many are full. This makes it easy for the CSG team to help people struggling to get onto one of their available licenses because they can use the licstats script as a means to troubleshoot. If all licenses were in use at the time of the report, then they are easily able to tell the user that all available licenses are full. My job was to understand how the script was working, where it was pulling its license data from, and then converting that to a Python Script.

```
mpasha@cheyenne3:~> ./licstats

  Envi:   0 in use /   2 available
   IDL:   1 in use /  53 available
 Intel:   1 in use /  50 available
Matlab:   6 in use /  50 available
   PGI:   1 in use /   2 available
```

This is a sample output of how licstats works.

My first thoughts when I saw the PERL script were… pure and utter confusion.

The scripts in PERL I had worked with in NETS were small and concise, and I had spent about a month and a half just learning what they did. With this script, it was a lot longer, it was pulling variables every which way, and it was using complex forms of data structures like hash tables to store everything. When I saw the script, I was definitely intimidated, but I decided to piece apart the small things I could.

```
if ($opt_s) {
# display detail for software suite $opt_s
        if (lc($opt_s) eq 'matlab') {
                $fam = 'Matlab';
        } elsif (lc($opt_s) eq 'idl') {
                $fam = 'IDL';
        } elsif (lc($opt_s) eq 'pgi') {
                $fam = 'PGI';
        } elsif (lc($opt_s) eq 'intel') {
                $fam = 'Intel';
        } elsif (lc($opt_s) eq 'envi') {
                $fam = 'Envi';
        }


        print ("\nIn use / Total available -- $fam component\n");
        print ("-----------------------------------------------\n");
        foreach my $k (sort keys %$lic) {
                if ($fam eq $lic->{$k}->{fam}) {
                        printf ("%2d / %2d -- %s\n", $lic->{$k}->{issu}, $lic->{$k}->{tot}, $lic->{$k}->{name});
                }
        }
```

Some of the parts of the Script had comments here and there, so it wasn't that difficult to figure out what part of the script handled fetching the variables, or outputting them to the screen, but rather the hard part was… how?

3. Starting on the Project

I spent a few days on this, analyzing what was happening, where the variables were coming from, and even built a shell of a Python Script, trying some of the imports they used in the PERL script and seeing if I could get anything to run. I copied the PERL script over word for word and decided to do it the old fashioned way, commenting out everything and going line by line to convert it to Python…

This, however, proved very quickly that it was not going to work.

The licstats script was a complexly and intricately woven piece of code that was printing things and fetching things and storing things in parallel, so even trying to go line by line was a disaster! I'd start from the first line then realize I needed the 20th line to be uncommented in order for something to work, or I'd modify the middle then realize the last few lines had to be there for anything to display. Modifications were a nightmare and I felt absolutely lost trying to go line by line and rewrite everything manually. After about a week of getting nowhere, I knew it was time for a strategy change. It was time to get to the bottom of this script and figure out the roots of it all. I shoved aside the display statements, the data structures, and the formatting, focusing on where it was that the variables themselves were coming from. I asked my mentor Mick if he could provide me some insight into the licensing file that held all the different licenses of NCAR so that I could really start to brainstorm how to structure my script.

4. Progress in Python

After figuring out the exact command used to pull all licensing information from Mick, it was time to start putting what I was learning into the Script. He recommended I scrap the modification approach and start from square one, which is actually what made the process so much easier for me. With a blank canvas, I no longer needed to worry about testing certain lines and then realizing I needed to uncomment certain chunks of code for those to work… I could go top down and rewrite this script the exact way I wanted.

One of the most powerful parts of Python is it's built in functionality to work at the OS level. Mick had given me the linux command that was the root of licstats… Fetching license data.

I knew that my Python script needed to utilize that linux command to fetch its license data, the way licstats was doing. A quick google search lended me to a method I had never heard of before in Python, which allows users to interactively use terminal commands in a Python Script. It's an import known as "os", and it lets you use linux commands within Python.

This got the gears in my head turning, and all of a sudden, I knew exactly what I was going to do.

```
#!/usr/bin/python
# -----
#  Script:      lic_stats.py, revised from PERL
#  Revision by: Muntaha Pasha
#  Version:     --
#  Date:        April 2020
# -----

#from html import HTML
import sys
import re
import os
import subprocess

#Use os system to execute system level commands: chdir, regEX, etc.
#Use egrep to display only the information we want to see.
os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 28518\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error" |
 grep -v "license server" | egrep -w "PGI2018|IBDB034BC" > tmp.out')
os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 27000\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error" |
 grep -v "license server" | egrep -w "MATLAB|idl|envi" > tmp2.out')
```

After setting up my script header and adding a few comments (for readability purposes), I put the os import into full force. I copied the Licensing data fetch command that Mick had provided me, and used a grep with it to pick out only the licenses that licstats used. One thing that was interesting was there were two different sets of licensing data with two different linux commands, so I needed two os.system calls with each command, and an "egrep", specifying exactly what licensing data I wanted. Licstats displays 5 licenses, which include MATLAB, IDL, Intel, PGI, and Envi.

5. Variable Struggles

One thing I struggled with were some of the specific variables, such as Intel and PGI. There were many upon many PGI licenses, so trying to figure out which one licstats was using was a bit of a hassle. Once I determined which ones the original script used, I noticed that it was somehow able to actually concatenate two PGI variables into one stat. I was actually unsure of how I could do this in Python, so I modified my script to just print both PGI variables instead of trying to complicate things further by using concatenation in linux. Intel was another tricky variable. It was actually not named Intel in the long licensing list, it was a variable called "IBDB034…". I had to actually match licenses in use with licenses available to spot which one was Intel.

6. Finishing Touches

After I had all 5 licensed stats printing, it was time to make it look more appealing! Currently, the way mine was printing it looked a bit cluttered and hard to make out what numbers were where.

```
Users of PGI2018-532568:  (Total of 1 license issued;  Total of 0 licenses in use)
Users of PGI2018-532569:  (Total of 1 license issued;  Total of 0 licenses in use)
Users of IBDB034BC:  (Total of 50 licenses issued;  Total of 3 licenses in use)
Users of idl:  (Total of 320 licenses issued;  Total of 6 licenses in use)
Users of envi:  (Total of 2 licenses issued;  Total of 0 licenses in use)
Users of MATLAB:  (Total of 50 licenses issued;  Total of 6 licenses in use)
```

In doing some brainstorming with my mentor, he told me to print my output into some kind of temporary file, and then modify the individual pieces of the licstats variables in order to get it to print in a more readable manner.

I knew there was a SED command I could use for replacement. So what I did was import yet another Python feature which lets users use specific regex commands like sed, awk, etc. It was the subprocess import.

```
subprocess.call(["cat tmp.out | sed 's/Users of //g' | sed 's/Total of //g' | sed -e 's/(//g' |sed -e 's/)//g' | sed 's/IBDB034BC/Intel/g'"], shell=True)
subprocess.call(["cat tmp2.out | sed 's/Users of //g' | sed 's/Total of //g' | sed -e 's/(//g' |sed -e 's/)//g'"], shell=True)
```

I called subprocess, and fed it my temporary files, then used the sed command Mick had provided me to get the data looking more concise and finished. I also decided to use the opportunity to rename Intel's licensing key to it's actual name!

```
mpasha@cheyenne3:~> ./lic_stats_fix.py
PGI2018-532568:  1 license issued;  0 licenses in use
PGI2018-532569:  1 license issued;  0 licenses in use
Intel:  50 licenses issued;  2 licenses in use
idl:  320 licenses issued;  6 licenses in use
envi:  2 licenses issued;  0 licenses in use
MATLAB:  50 licenses issued;  6 licenses in use
```

This ended up being the final product. The PERL script was 231 lines in total, and my Python script ended up being 25 lines, with 4 lines being the actual code. That was it! 4 lines. It honestly left me awestruck when I realized the power of modern day languages, and how much I was able to achieve in just 4 lines of Python Code. I not only condensed down the PERL script to something very small and easily readable, but also to something more efficient overall. I didn't need to store anything in any kind of data structure or use up memory resources, I was able to just pipe through a linux command using an OS import, and then perform a simple regex command on it to make it look more concise. All in all, it was an amazing feeling to be able to see the working script and know it was 227 lines of less code than the original.

```
#!/usr/bin/python
# -----
#  Script:      lic_stats.py, revised from PERL
#  Revision by: Muntaha Pasha
#  Version:     --
#  Date:        April 2020
# -----

#from html import HTML
import sys
import re
import os
import subprocess

#Use os system to execute system level commands: chdir, regEX, etc.
#Use egrep to display only the information we want to see.
os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 28518\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error" |
 grep -v "license server" | egrep -w "PGI2018|IBDB034BC" > tmp.out')
os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 27000\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error" |
 grep -v "license server" | egrep -w "MATLAB|idl|envi" > tmp2.out')
#os.system('cat tmp.out | sed 's/Users of //g' | sed 's/Total of //g' | sed -e 's/(//g' |sed -e 's/)//g'')
subprocess.call(["cat tmp.out | sed 's/Users of //g' | sed 's/Total of //g' | sed -e 's/(//g' |sed -e 's/)//g' | sed 's/IBDB034BC/Intel/g'"], shell=True)
subprocess.call(["cat tmp2.out | sed 's/Users of //g' | sed 's/Total of //g' | sed -e 's/(//g' |sed -e 's/)//g'"], shell=True)
#Print Statement to see ALL Licenses
#os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 27000\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error"
| grep -v "license server"')
#os.system('/glade/u/apps/ch/opt/matlab/R2019a/etc/glnxa64/lmutil lmstat -c 28518\@stargate11.nwsc.ucar.edu -a | grep -E "Users|ysmgt" | grep -vE "License|vendor|MASTER|Vendor|Error"
| grep -v "license server"')
```

This is an image of my final script.

7. Concluding Thoughts

Coming into this rotation, I did feel rather intimidated at first by the large licstats script awaiting my modifications, but taking it day by day and learning that it was okay to be stuck, and that it was perfectly acceptable to not understand everything right off the bat was really helpful. This rotation occurred in the midst of the COVID-19 Pandemic, so dealing with that alongside my work here presented its own set of challenges, but I was lucky to have an amazing Mentor who was patient and immensely helpful throughout all stages of the project, and I was also lucky to have had some prior experience in Python to help me build up my script the way I wanted. Things that went well for me were being able to

always reach out for help via email, or hangouts, because communication during these trying times was a really important thing to uphold. There would be times where I couldn't really make progress until I had some deeper level understanding of where the licenses were located, how I could access them, etc. Things that I would have done differently might've been to have started from scratch since the beginning. I think copying the PERL script and trying to modify it line by line got way too overwhelming way too fast, and it definitely left me questioning if this was a feat I would be able to accomplish. However, once I re-approached the problem in terms of its basic functionality, I was really able to take this project and make the most of it. I learned so much about Python scripting in the process, and these things are what I plan to take with me into my future endeavors.

Overall, it was such a fulfilling experience, and I thank both the CSG team and my mentor Mick Coady for making this rotation such a great one. It was the perfect note to end my time here in the Student Assistant Rotation Program and I couldn't be more grateful for all the things I learned during this time.