Lecture 14: Supercomputing @CRC

LING 1340: Data Science for Linguists

Na-Rae Han

Objectives

- Supercomputing at CRC
  - Server access through SSH
  - Running a job on CRC

Let us now supercompute.

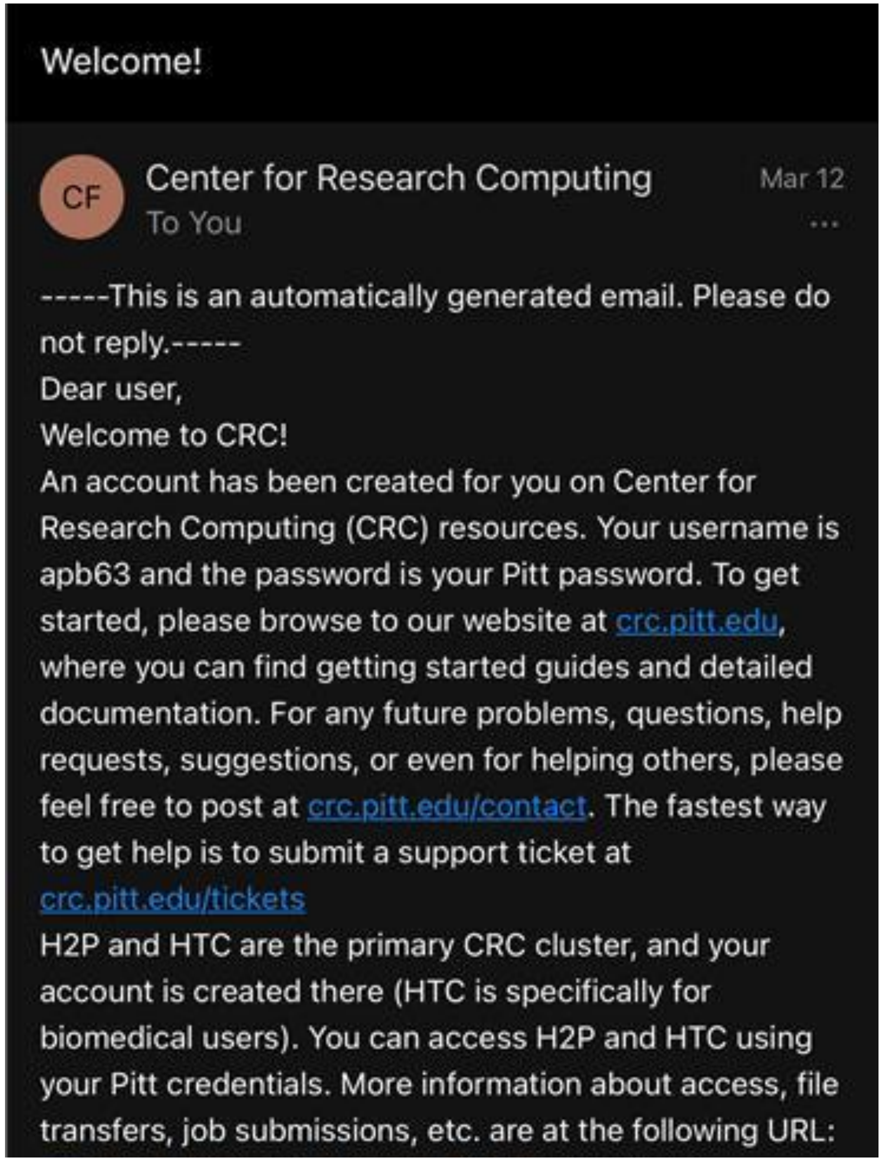
By Argonne National Laboratory's Flickr page - originally posted to Flickr as Blue Gene / PFrom Argonne National Laboratory Uploaded using F2ComButton, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=6412306

You got a supercomputing account.

- You received this mysterious email:

I got you all an account at Pitt's **Center for Research Computing** (CRC)

- CRC: Center for Research Computing
  - https://crc.pitt.edu
  - Handy links in "Resource" page!



Welcome!

CF Center for Research Computing Mar 12
To You

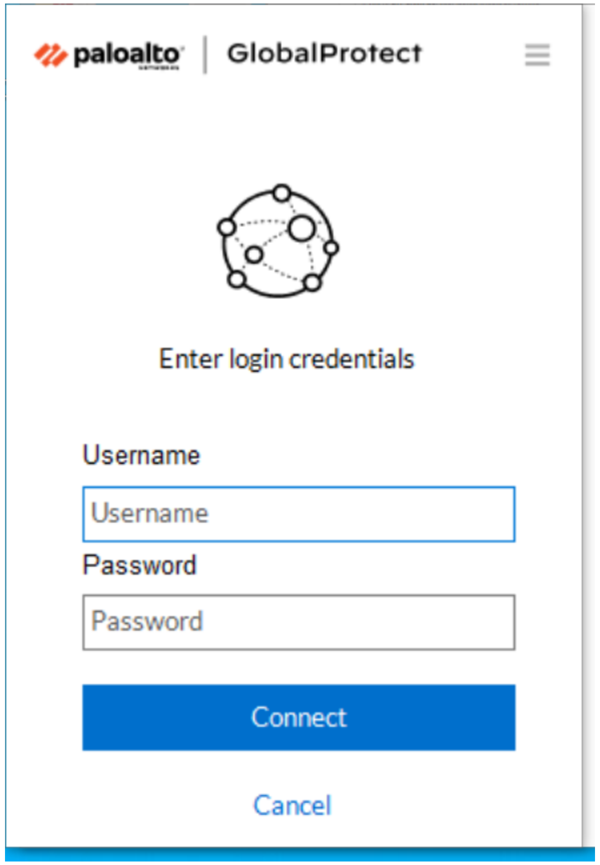
-----This is an automatically generated email. Please do not reply.-----
Dear user,
Welcome to CRC!
An account has been created for you on Center for Research Computing (CRC) resources. Your username is apb63 and the password is your Pitt password. To get started, please browse to our website at crc.pitt.edu, where you can find getting started guides and detailed documentation. For any future problems, questions, help requests, suggestions, or even for helping others, please feel free to post at crc.pitt.edu/contact. The fastest way to get help is to submit a support ticket at crc.pitt.edu/tickets
H2P and HTC are the primary CRC cluster, and your account is created there (HTC is specifically for biomedical users). You can access H2P and HTC using your Pitt credentials. More information about access, file transfers, job submissions, etc. are at the following URL:

Accessing CRC's cluster

- If you're **OFF CAMPUS**, your laptop should be running a **PittNet VPN client**.
  - Install and run **GlobalProtect** (Pitt IT how-to page) →
- Remote-access your account via SSH:
  - `ssh yourpittid@h2p.crc.pitt.edu`
  - Additional instructions on this CRC how-to page
- Getting your bearings:
  - Where are you? `pwd`
  - What is your user 'group'? `groups`
  - Is python installed on this machine? `which python`
  - What are your configuration files? `ls -a`
    - `.bash_profile`

      ← Customize with your own aliases, etc.
    - `.bash_history`

      ← Bash commands you typed in are logged here.
  - How to log out of CRC? `exit`



Na-Rae's .bash_profile on CRC

- PATH configuration
- Prompt in pink!! Add this line:

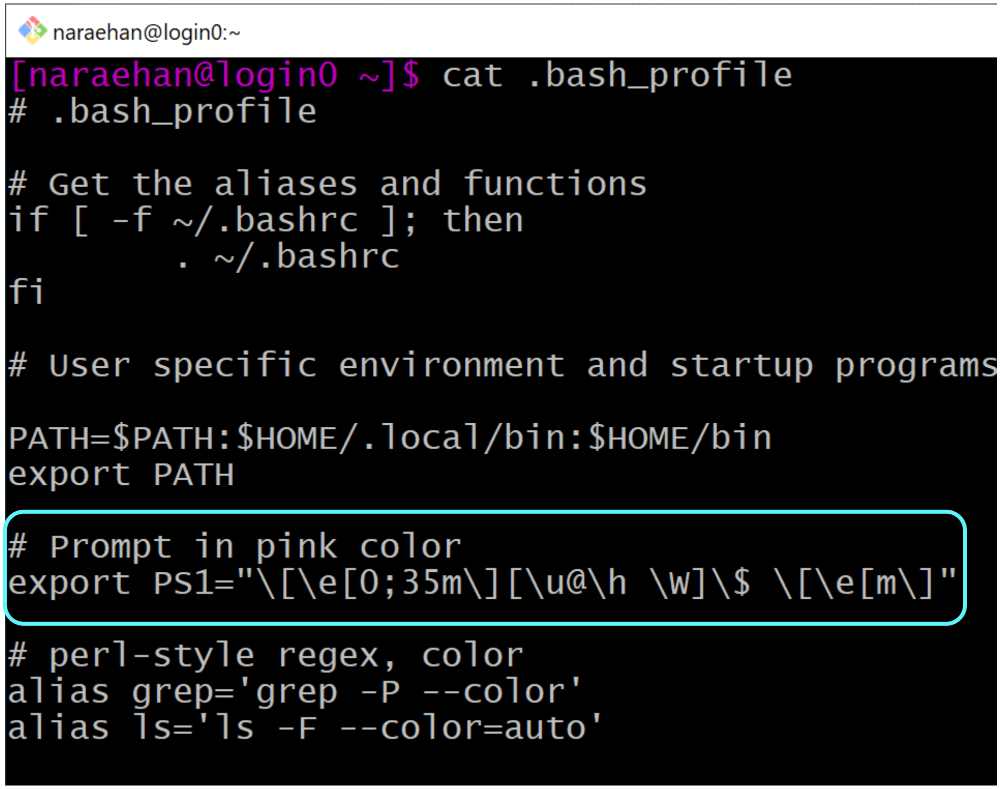
  `export PS1="\[\e[0;35m\][\u@\h \W]\$ \[\e[m\]"`

Use nano to edit text files (including .py)

- Some aliases
  - grep: perl style, colored output
  - ls: colored output, folders marked with "/"

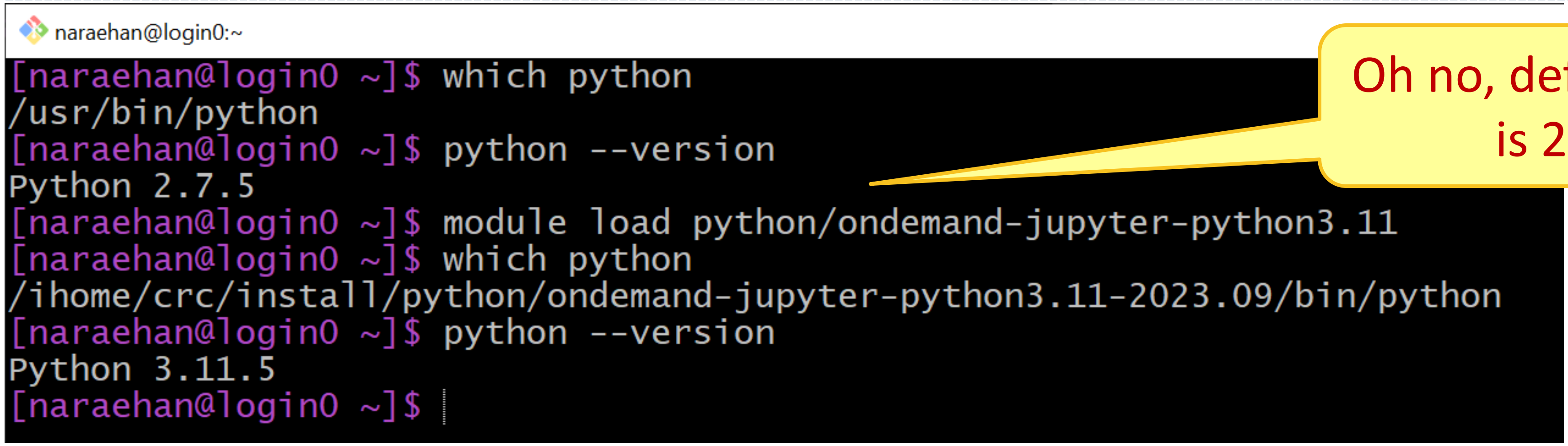
If you edit this file, changes take effect after logging back in.

For immediate effect, run:
`source .bash_profile`


naraehan@login0:~

```
[naraehan@login0 ~]$ cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin
export PATH

# Prompt in pink color
export PS1="\[\e[0;35m\][\u@\h \W]\$ \[\e[m\]"

# perl-style regex, color
alias grep='grep -P --color'
alias ls='ls -F --color=auto'
```

Using the right python module



naraehan@login0:~

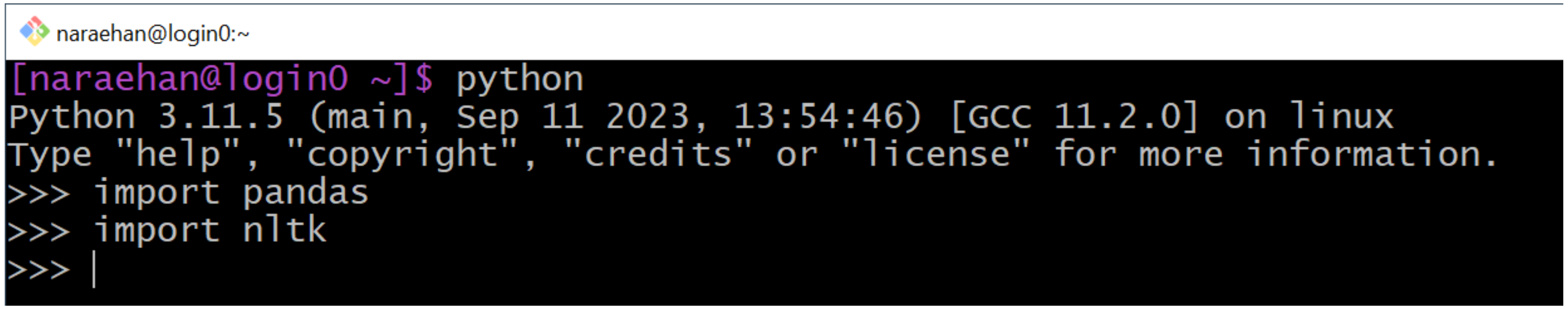
```
[naraehan@login0 ~]$ which python
/usr/bin/python
[naraehan@login0 ~]$ python --version
Python 2.7.5
[naraehan@login0 ~]$ module load python/ondemand-jupyter-python3.11
[naraehan@login0 ~]$ which python
/ihome/crc/install/python/ondemand-jupyter-python3.11-2023.09/bin/python
[naraehan@login0 ~]$ python --version
Python 3.11.5
[naraehan@login0 ~]$
```

Oh no, default Python is 2.7.5...

- We have to "load" a correct (for us) python module via

  `module load python/ondemand-jupyter-python3.11`

- Popular data science libraries are already installed (pandas, sklearn, nltk...):



naraehan@login0:~

```
[naraehan@login0 ~]$ python
Python 3.11.5 (main, Sep 11 2023, 13:54:46) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
>>> import nltk
>>> 
```

Using CRC clusters

- Job submissions
  - On a computing cluster, many people are using the same resources so we have a "job queue" that accepts job submissions
  - CRC and many other clusters use Slurm for managing and scheduling these jobs.

- What this means:
  - You don't directly execute your Python script. (A big NO-NO)
  - You create a BASH SCRIPT to run a PYTHON SCRIPT (job).

Before you get carried away

- Do NOT yet run any commands/jobs that may be resource-intensive.
- This is a powerful super-computer, shared by many research groups at Pitt.
  - Our class as a group has a limited, shared allocation. We have a reserve of **10000 Service Units** (SUs), which is 10k hours of computing time.
  - You do not want to accidentally initiate a run-away process and hog resources.
- There are PROPER ways to run jobs.
  - We will show you now.

Slurm Jobs

- To make a slurm job script, you basically need to write a **bash script** of what you would do to run your program on the command line. This is just a text file, usually with a **.sh** ending.
- Also need some slurm configs
- Example (let's call this `hello.sh`)

```
#!/usr/bin/env bash

#SBATCH --job-name=hello
#SBATCH --output=hello.out
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=smp
#SBATCH --cluster=smp

echo "hello world"
```

<-- Copy this into a file and name it something like **hello.sh**

Below are some other Slurm config options (prefix with #SBATCH) as in hello.sh. EVEN MORE at https://slurm.schedmd.com/sbatch.html

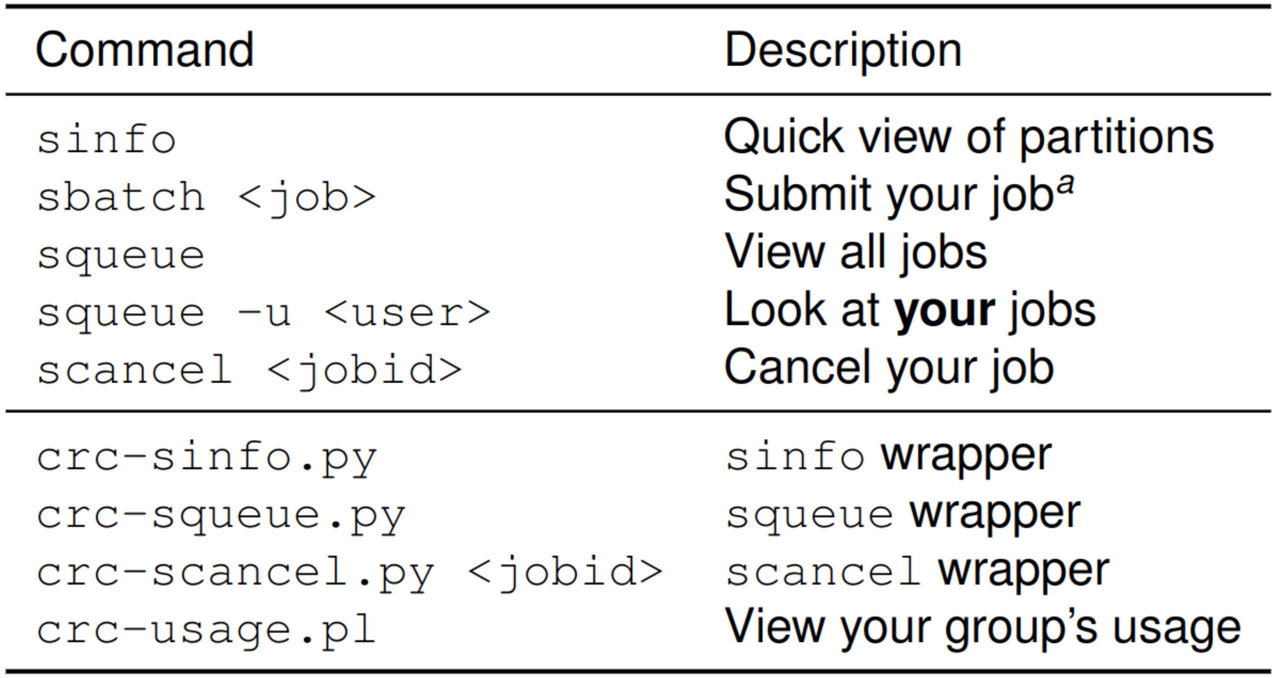
Option	Environment Variables
--output	–
--time	(Format: DAYS-HOURS:MINUTES:SECONDS)
--job-name	SLURM_JOB_NAME
--nodes	SLURM_NNODES
--ntasks	SLURM_NTASKS
--cpus-per-task	SLURM_CPUS_PER_TASK
--ntasks-per-node	SLURM_NTASKS_PER_NODE
--partition	SLURM_JOB_PARTITION
--mem	SLURM_MEM_PER_NODE
--account	SLURM_JOB_ACCOUNT

Job management commands

So from the directory with our `hello.sh` script, we can submit it with:

```
sbatch hello.sh
```

This should run pretty much instantly and we can check our `hello.out` output file.



Command	Description
`sinfo`	Quick view of partitions
`sbatch <job>`	Submit your job[a]
`squeue`	View all jobs
`squeue -u <user>`	Look at **your** jobs
`scancel <jobid>`	Cancel your job
`crc-sinfo.py`	`sinfo` wrapper
`crc-squeue.py`	`squeue` wrapper
`crc-scancel.py <jobid>`	`scancel` wrapper
`crc-usage.pl`	View your group's usage

To-do #14 redux on CRC: setting up

(1) Location of yelp review data file (you all have access):

```
/ix1/ling2340_2025s/shared_data/yelp_dataset_2021/yelp_academic_dataset_review.json
```

(2) We'll sample 1 million lines, shuffled (make sure to use correct file path):

```
shuf yelp_academic_dataset_review.json -n 1000000 > ~/review_1mil.json
```

(3) Copy over our python script. Running it on this data will look like: (but we shouldn't be running this command directly!)

```
python process_reviews.py review_1mil.json
```

(4) But before that, we should load the appropriate python environment:

```
module load python/ondemand-jupyter-python3.11
```

(5) Now we can toss all this into a bash script. Let's call it `todo14.sh` :

- Start with `hello.sh` (make a copy using `cp file1 file2`, then edit)
- Change the bash commands at the bottom to run our script for To-do 14, and change the job name and output file to something like `todo14` and `todo14.out`

todo14.sh

```
#!/usr/bin/env bash

#SBATCH --job-name=todo14
#SBATCH --output=todo14.out
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=smp
#SBATCH --cluster=smp

module load python/ondemand-jupyter-python3.11
python process_reviews.py review_1mil.json
```

SLURM Job script

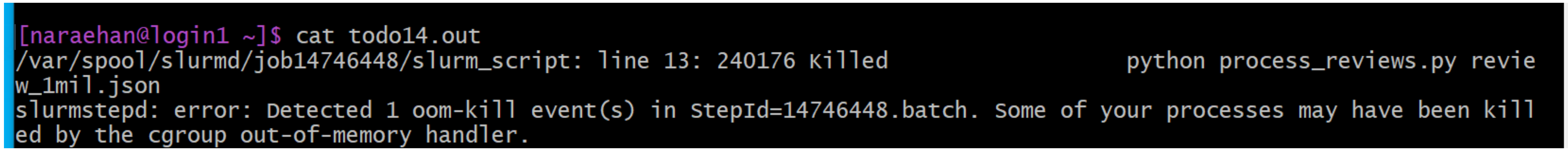
Python script

process_reviews.py
(from To-do #14)

```
import pandas as pd
import sys
from collections import Counter

filename = sys.argv[1]

df = pd.read_json(filename, lines=True, encoding='utf-8')
print(df.head(5))

wtoks = ' '.join(df['text']).split()
wfreq = Counter(wtoks)
print(wfreq.most_common(20))
```

Nope, the job gets killed…

- Processing 1mil reviews required too big a memory usage, which prompts slurm to kill the job:



```
[naraehan@login1 ~]$ cat todo14.out
/var/spool/slurmd/job14746448/slurm_script: line 13: 240176 Killed                  python process_reviews.py revie
w_1mil.json
slurmstepd: error: Detected 1 oom-kill event(s) in StepId=14746448.batch. Some of your processes may have been kill
ed by the cgroup out-of-memory handler.
```

- We need to configure our job to request more **memory** up front:

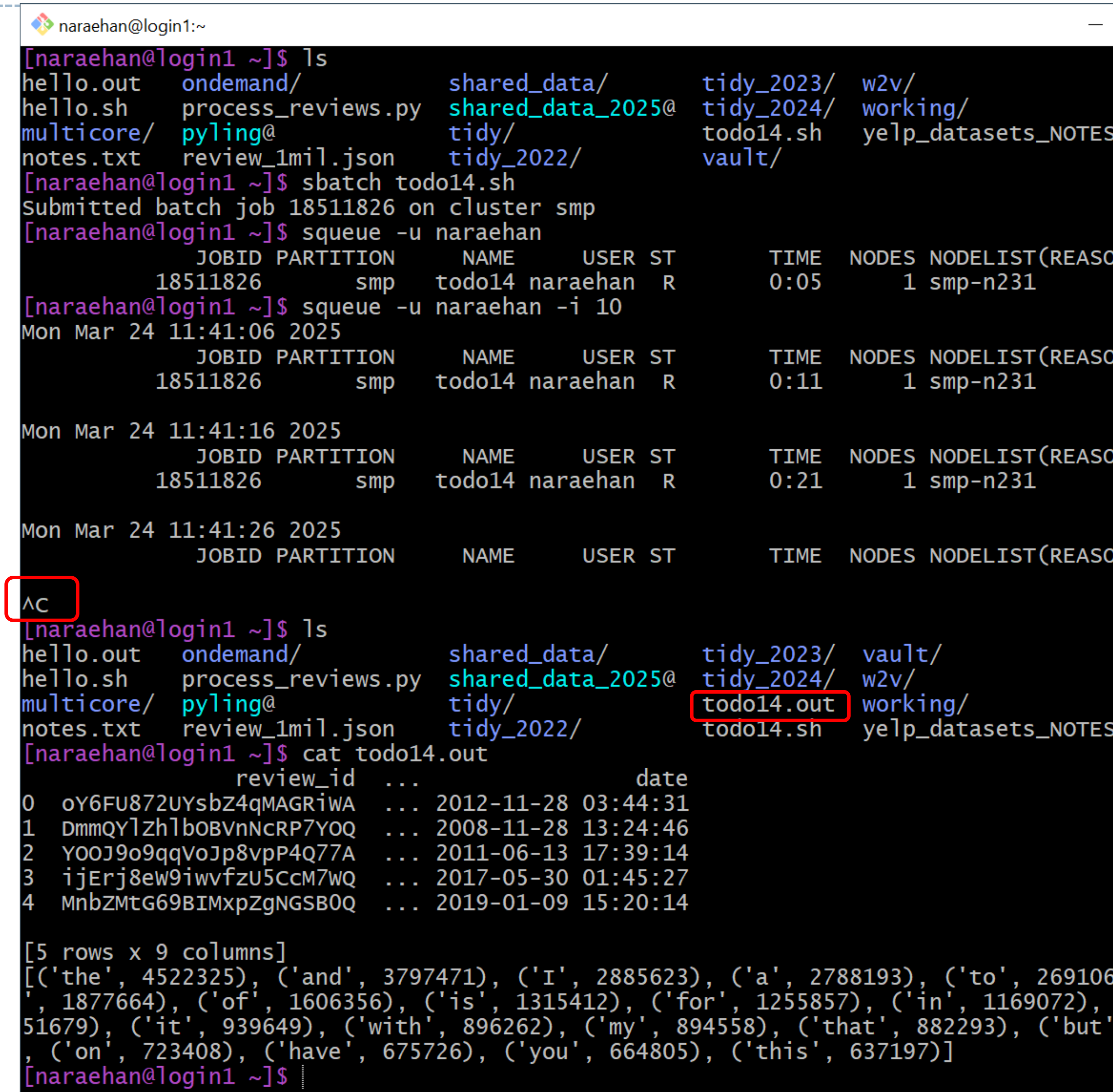
todo14.sh

```
#!/usr/bin/env bash

#SBATCH --job-name=todo14
#SBATCH --output=todo14.out
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=smp
#SBATCH --cluster=smp
#SBATCH --mem-per-cpu=12000

module load python/ondemand-jupyter-python3.11
python process_reviews.py review_1mil.json
```

12000 MB = about 12GB of memory per CPU
Hopefully this is enough…?

To-do #14 redux on CRC

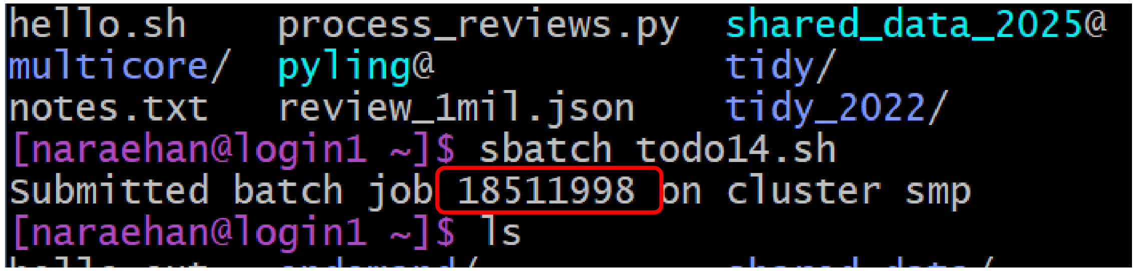
- Submit your job:
  - `sbatch todo14.sh`
- and check status with:
  - `squeue -u <user-id>`
  - Done when squeue no longer shows job (keep re-running with up arrow)
  - Or: add `-i 10` to auto-run every 10 seconds (Ctrl+c to get out)
- Check the output with:
  - `cat todo14.out`
- Success! 1 million reviews didn't take too long


naraehan@login1:~

```
[naraehan@login1 ~]$ ls
hello.out    ondemand/           shared_data/       tidy_2023/  w2v/
hello.sh     process_reviews.py  shared_data_2025@  tidy_2024/  working/
multicore/   pyling@             tidy/              todo14.sh   yelp_datasets_NOTES
notes.txt    review_1mil.json    tidy_2022/         vault/
[naraehan@login1 ~]$ sbatch todo14.sh
Submitted batch job 18511826 on cluster smp
[naraehan@login1 ~]$ squeue -u naraehan
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASO
          18511826       smp   todo14 naraehan  R       0:05      1 smp-n231
[naraehan@login1 ~]$ squeue -u naraehan -i 10
Mon Mar 24 11:41:06 2025
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASO
          18511826       smp   todo14 naraehan  R       0:11      1 smp-n231

Mon Mar 24 11:41:16 2025
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASO
          18511826       smp   todo14 naraehan  R       0:21      1 smp-n231

Mon Mar 24 11:41:26 2025
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASO

^C
[naraehan@login1 ~]$ ls
hello.out    ondemand/           shared_data/       tidy_2023/  vault/
hello.sh     process_reviews.py  shared_data_2025@  tidy_2024/  w2v/
multicore/   pyling@             tidy/              todo14.out  working/
notes.txt    review_1mil.json    tidy_2022/         todo14.sh   yelp_datasets_NOTES
[naraehan@login1 ~]$ cat todo14.out
                review_id  ...                 date
0  oY6FU872UYsbZ4qMAGRiWA  ...  2012-11-28 03:44:31
1  DmmQYlZhlbOBVnNCRP7YOQ  ...  2008-11-28 13:24:46
2  YOOJ9o9qqVoJp8vpP4Q77A  ...  2011-06-13 17:39:14
3  ijErj8ew9iwvfzU5CcM7WQ  ...  2017-05-30 01:45:27
4  MnbZMtG69BIMxpZgNGSBOQ  ...  2019-01-09 15:20:14

[5 rows x 9 columns]
[('the', 4522325), ('and', 3797471), ('I', 2885623), ('a', 2788193), ('to', 269106
', 1877664), ('of', 1606356), ('is', 1315412), ('for', 1255857), ('in', 1169072),
51679), ('it', 939649), ('with', 896262), ('my', 894558), ('that', 882293), ('but'
, ('on', 723408), ('have', 675726), ('you', 664805), ('this', 637197)]
[naraehan@login1 ~]$
```

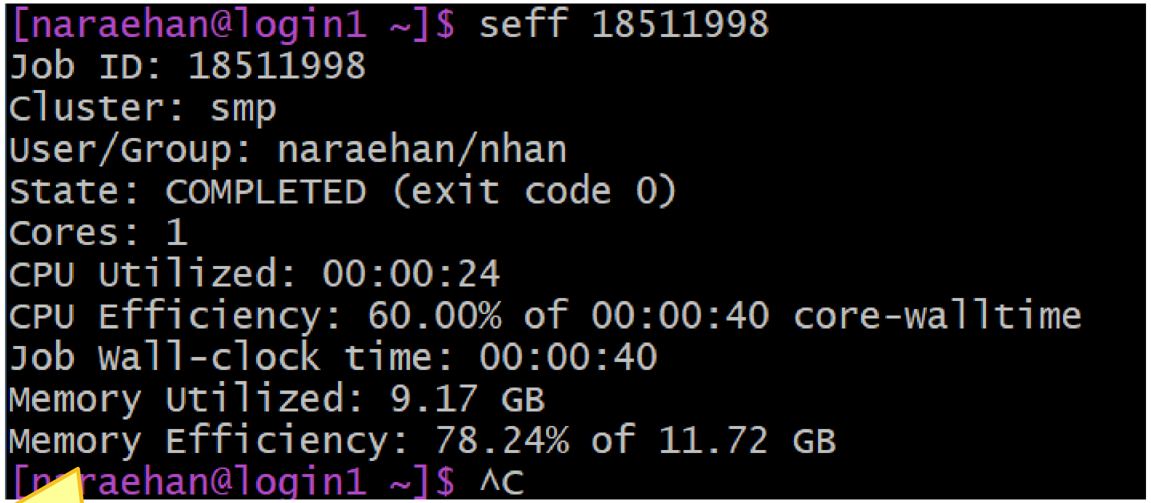
How did the job go?

- Job ID was shown earlier →

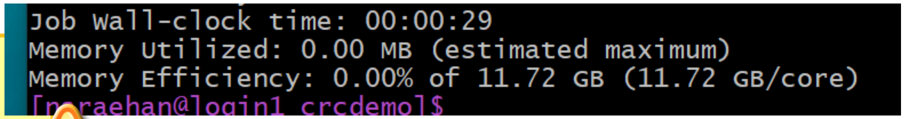


```
hello.sh     process_reviews.py   shared_data_2025@
multicore/   pyling@              tidy/
notes.txt    review_1mil.json     tidy_2022/
[naraehan@login1 ~]$ sbatch todo14.sh
Submitted batch job 18511998 on cluster smp
[naraehan@login1 ~]$ ls
```

- Check finished job's stats by:
  - `seff <job-id>`
- Our Python script on 1million reviews took up:
  - 24 seconds of CPU time
  - 9.17GB of memory (RAM)



```
[naraehan@login1 ~]$ seff 18511998
Job ID: 18511998
Cluster: smp
User/Group: naraehan/nhan
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:00:24
CPU Efficiency: 60.00% of 00:00:40 core-walltime
Job Wall-clock time: 00:00:40
Memory Utilized: 9.17 GB
Memory Efficiency: 78.24% of 11.72 GB
[naraehan@login1 ~]$ ^C
```

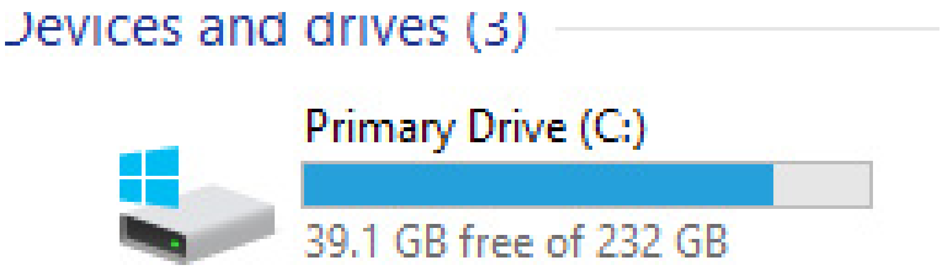


```
Job Wall-clock time: 00:00:29
Memory Utilized: 0.00 MB (estimated maximum)
Memory Efficiency: 0.00% of 11.72 GB (11.72 GB/core)
[naraehan@login1 crcdemo]$
```

For short jobs (< 30 seconds) like this, memory footprint might not get captured, reported as 0.00 MB

Quick aside: computing hardware

- **Memory** here refers to **Random Access Memory** (RAM)
  - You probably have 8 or 16 GB on your laptop
  - Running programs uses **RAM** to store temporary data (in our case opened file content, variables, lists, DataFrame, etc) that they use or produce
  - Stuff stored in RAM is removed when a program terminates, or if your computer shuts off.
  - Running out of RAM on your laptop could cause your computer to freeze/crash
  - Expensive per GB



Devices and drives (3)

Primary Drive (C:)

39.1 GB free of 232 GB

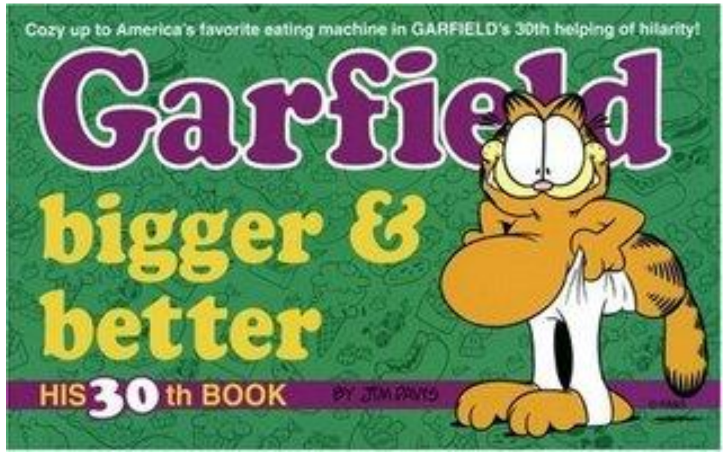
- NOT disk drive -->
  - Disk space stores files long-term
  - Cheap per GB, 256+GB is pretty standard.

To-do #15: bigger data + better code @ CRC!

- Take 1: use 4 million reviews
- Take 2: use 4 million reviews, with a new (better!) python script

  🡐 Compare Take 1 vs. Take 2

- Take 3 (optional): all 7 million reviews, with the new (better!) python script


Cozy up to America's favorite eating machine in GARFIELD's 30th helping of hilarity!
Garfield
bigger & better
HIS 30 th BOOK
BY JIM DAVIS