"Bash vs Python Throwdown"

-or-

"How you can accomplish common tasks using each of these tools"

Bash Examples

Copying a file:

```
$ cp file1 file2
```

Wrangling "csv" files:

Consider a file named 20140209.csv containing batch job data; some example lines are

```
2488112,lucy,UCB178,crc-gpu,1,1,14400,12963,547328kb,12987,1391793880,1391905890,1391914447,1391927435,0,singlejob,43.2900
2490186,wyldstyle,UCB191,janus-normal,21,12,86400,21710392,113415kb,86429,1391841,1391841,1391841,1391927463,-11,default,6050.0300
2468732,metalbeard,S232,janus-small,1,12,43200,159309,298188kb,13393,1391489055,1391914171,1391914191,1391927583,0,default,44.6433
2468698,metalbeard,S232,janus-small,1,12,43200,344489,310540kb,28871,1391488981,1391898836,1391898863,1391927733,0,default,96.2366
2488073,gandalf,UCB178,janus-small,1,12,86400,820612,124072kb,69136,1391791582,1391858648,1391858666,1391927801,0,singlejob,230.4533
```

The most interesting fields are 1-jobID, 2-username, 3-allocation, 17-SUconsumed

First, sum all SU consumed by all jobs:

```
$ awk -F, '{ sum+=$17} END {print sum}' 20140209.csv
  38062.8
```

Print the full record for all of gail's jobs:

```
$ awk -F, '$2=="gail"' 20140209.csv
```

```
2493201,gail,UCB00000256,janus-small,18,12,28800,0,0kb,0,1392007772,1392007772,1392008303,1392008589,0,default,0
2493202,gail,UCB256,janus-small,18,12,28800,400019,86607652kb,2660,1392007841,1392007841,1392007886,1392010552,0,default,159.6000
2493203,gail,UCB256,janus-small,18,12,28800,623527,83139976kb,3618,1392007890,1392007890,1392007918,1392011536,0,default,217.0800
```

Sum gail's total SU consumed:

```
$ awk -F, '$2=="gail" {sum+=$17} END {print sum}' 20140209.csv
  376.68
```

Sum all of the users' SU individually:

```
$ for u in `awk -F, '{print $2}' 20140209.csv | sort -u`; do
> echo $u `awk -F, -v user="$u" '$2==user {sum+=$17} END {print sum}' 20140209.csv`
> done

badcop 293.77
batman 6136
benny 6979.05
business 1390.05
emmet 83.3683
gail 376.68
gandalf 3898.31
hansolo 34.8965
lucy 476.396
metalbeard 2188.77
shaq 646.053
unikitty 1505.71
vitruvius 432.335
wyldstyle 13621.4
```

Pick out the top 6 by SU used

```
$ for u in `awk -F, '{print $2}' 20140209.csv | sort -u`; do
> echo $u `awk -F, -v user="$u" '$2==user {sum+=$17} END {print sum}' 20140209.csv`
> done |sort -k2 -nr | head -6

wyldstyle 13621.4
benny 6979.05
batman 6136
gandalf 3898.31
metalbeard 2188.77
unikitty 1505.71
```

Unstructured Text Manipulation: Hamlet

Make all words lower case, take out punctuation marks, translate white space into newlines to create a list of individual words, remove blank lines, and count total lines/words/chars:

```
$ cat hamlet.txt | tr '[:upper:]' '[:lower:]' | tr -d '[:punct:]' \
 | tr -s '[:space:]' '\n' | sed '/^[[:space:]]*$/d' | wc

26892   26892  140810
```

How many unique words?

```
$ cat hamlet.txt | tr '[:upper:]' '[:lower:]' | tr -d '[:punct:]' \
 | tr -s '[:space:]' '\n' | sed '/^[[:space:]]*$/d' | sort -u | wc -l

4263
```

What are the most common words?

```
$ cat hamlet.txt | tr '[:upper:]' '[:lower:]' | tr -d '[:punct:]' \
 | tr -s '[:space:]' '\n' | sed '/^[[:space:]]*$/d' | sort | uniq -c \
 | sort -nr | head -10

    929 the
    842 and
    629 to
    562 of
    488 you
    463 i
    438 my
    438 a
    370 in
    363 hamlet
```

What are the most common words with 4+ letters?

```
$ cat hamlet.txt | tr '[:upper:]' '[:lower:]' | tr -d '[:punct:]' \
 | tr -s '[:space:]' '\n' | sed '/^[[:space:]]*$/d' | sort | uniq -c \
 | awk 'length($2)>3' | sort -nr | head -10

    363 hamlet
    330 that
    277 lord
    237 this
    231 with
    211 your
    174 what
    167 king
    147 have
    134 will
```

Locating and deleting my queued compute jobs

The qstat command lists all jobs that the scheduling system knows about. Its output looks like this

```
moab.rc.colorado.edu:                                                        Req'd    Req'd       Elap
Job ID               Username    Queue    Jobname          SessID NDS   TSK    Memory   Time    S   Time
-------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
2515176.moab.rc.     badcop      janus-sm nmphi50r150-2     30420     1     12    --   24:00:00  R  00:43:52
2447765.moab.rc.     wyldstyle    janus-sm 1_Poly2_2_1_17      --      1     12    --   07:00:00  H  00:00:00
2450992.moab.rc.     wyldstyle    janus-sm 1_Poly5_2_1_39      --      1     12    --   10:00:00  H  00:00:00
2497824.moab.rc.     badcop      janus-de vasp               --      1     12    --   01:00:00  Q       --
2498067.moab.rc.     wyldstyle    janus-sm 1_Poly2_2_2_38     1118     1     12    --   15:00:00  R  00:25:37
```

If my username is "emmet", I can look at my jobs via

```
$ qstat -a | grep emmet

2514880.moab.rc.     emmet       janus-lo q0.3e0.8          13787     1     12    --  100:00:00  Q       --
2514881.moab.rc.     emmet       janus-lo alpha0.01rout04   13875     1     12    --  168:00:00  R  00:16:43
2514882.moab.rc.     emmet       janus-lo alpha0.01rout16   13611     1     12    --  168:00:00  R  00:16:43
2514883.moab.rc.     emmet       janus-lo alpha0.01rout64   14270     1     12    --  168:00:00  R  00:16:43
2514884.moab.rc.     emmet       janus-lo alpha0.05rout04   13594     1     12    --  168:00:00  R  00:16:43
2514886.moab.rc.     emmet       janus-lo alpha0.05rout64   14674     1     12    --  168:00:00  H  00:00:00
2514887.moab.rc.     emmet       janus-lo alpha0.25rout04    3856     1     12    --  168:00:00  R  00:31:40
2514888.moab.rc.     emmet       janus-lo alpha0.25rout16    2749     1     12    --  168:00:00  R  00:21:54
2514889.moab.rc.     emmet       janus-lo alpha0.25rout64     --      1     12    --  168:00:00  Q       --
2517835.moab.rc.     emmet       janus-sh LT                27189     1      1    --   04:00:00  C       --
2517836.moab.rc.     emmet       janus-sh LT                27253     1      1    --   04:00:00  C       --
```

(Caution, this might return another user's job whose Jobname includes the string "emmet")

Get list of all my Job IDs:

```
$ qstat -a | awk '$2=="emmet" {print $1}' |cut -d. -f1

2514880
2514881
2514882
2514883
2514884
2514886
2514887
2514888
2514889
2517835
2517836
```

Get a list of all my Job IDs for jobs in the "Q" state

```
$ qstat -a | awk '$2=="emmet" && $10=="Q" {print $1}' |cut -d. -f1 | xargs qdel
```

Could also do this via a for-loop:

```
$ for j in `qstat -a | awk '$2=="emmet" && $10=="Q" {print $1}' |cut -d. -f1`; do
> echo $j
> qdel $j
> done
```

Numerical computing and plotting

Celsius to Fahrenheit converter shell script:

```
#!/bin/bash
# Calculate Fahrenheit equivalents of Celsius temperatures
#  from -40 to 40
c=-40
while [ $c -le 40 ]; do
  echo $c `echo "scale=3; (9/5)*$c+32" |bc`  # use "bc" for non-integer math
  c=$((c+1))  # increment c by 1
done
```

Shell script to plot sine function from -10 to 10 radians:

```
#!/bin/bash
# first, generate a file containing x-y coordinates of a sine function
#  from -10 to 10 radians
i=-100
while [ $i -le 100 ]; do
  # since we can only increment on integers in bash, have to divide in order
  # to get enough data points on x-axis
  x=`echo "scale=3; $i/10" | bc`
  y=`echo "scale=3; s($x)" | bc -l`   # need -l to enable trig functions
  echo "$x $y" > datafile.txt
  i=$((i+1))
done
# now plot to postscript file using gnuplot
gnuplot << EOF
set terminal postscript
set output "datafile.ps"
plot "datafile.txt" with linespoints
EOF
```

Text manipulation

Find and Replace:

```
$ sed 's/original/replacement/g' < input.txt > output.txt
```

(remember; the "s" means "substitute". This only works if "original" is all on one line)

Remove blank lines:

```
$ sed '/^$/d' < input.txt > output.txt
```

What about lines that only have white space in them?

```
$ sed '/^\s*$/d' file
```

Hmm, that's only spaces; what about tabs and other "blank" characters.
Use a character class:

```
$ sed '/^[[:space:]]*$/d'
```

(for more details see "man 7 regex" and "man isspace")

De-htmlize (note that regex doesn't handle all possible tag syntax perfectly) :

```
$ sed 's/<[^>]*>//g'
```

If the tags span lines then use

```
$ sed -e :a -e 's/<[^>]*>//g;/</N;//ba' test.html
```

(lots of good sed one-liners at
http://www.catonmat.net/blog/wp-content/uploads/2008/09/sed1line.txt)

What about acting on multiple files at once?

```
$ for f in `ls -1 *.txt`; do
> sed 's/original/replacement/g' < $f > $f.new
> done
```

What if the files are in several subdirectories?

```
$ for f in `find . -name "*.txt"`; do
> sed 's/original/replacement/g' < $f > $f.new   # compare "sed -i"
> done
```