# BINF2111 - Introduction to Bioinformatics Computing UNIX 101 part trois (grep regrex and sed)



Richard Allen White III, PhD RAW Lab Lecture 4 - Thursday Sep 1<sup>st</sup>, 2022

# **Learning Objectives**

- Review quiz and bonus
- Regular expressions in grep
- Sed
- Regular expressions in sed
- Quiz 4

### Quiz 3 answers

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### Quiz 3 answers

- grep -c "AT" example.fasta command is able to count all AT's in this line?

- grep -c "AT" example.fasta command is able to count all AT's in this line?

- grep -c "AT" example.fasta command is able to count all AT's in this line?

- grep -c "AT" example.fasta command is able to count all AT's in this line?

How would you count it?

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -c "AT" example.fasta

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -o -c "AT" example.fasta or grep -oc "AT" example.fasta

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -Eo "AT" example.fasta --color grep -o "AT" example.fasta

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -Eo "AT" example.fasta | wc -l grep -o "AT" example.fasta | wc -l

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -o "^AT|AT\$" example.fasta

- grep -c "AT" example.fasta command is able to count all AT's in this line?

### 

What this work? grep -Eo "^AT|AT\$" example.fasta | wc -l egrep -o "^AT|AT\$" example.fasta | wc -l

### Bonus 3

- For the example.fasta file count both the number of AT and GC in one grep command and in another command print the line number which they appear?

Command 1 grep -Eo "GC|AT" example.fasta | wc -l egrep -o "GC|AT" example.fasta | wc -l

Command 2 grep -nEo "GC|AT" example.fasta --color >>file.txt egrep -no "GC|AT" example.fasta --color >>file.txt

## Bonus 3 extra questions

- For the example.fasta file count both the number of AT and GC in one grep command and in another command print the line number which they appear?

How many GC and AT's are present count? Number here: command:

And, how many are GC and AT's individually? Commands here:
Number here:

## Bonus 3 extra questions

How many GC and AT's are present count?

Number here: GC = 8 and AT = 7

Number here: 15

- For the example.fasta file count both the number of AT and GC in one grep command and in another command print the line number which they appear?

command: egrep -o "GC|AT" example.fasta | wc -l
And, how many are GC and AT's individually?
Commands here: egrep -o "GC" example.fasta | wc -l
egrep -o "AT" example.fasta | wc -l

# grep – syntax to hands of UNIX

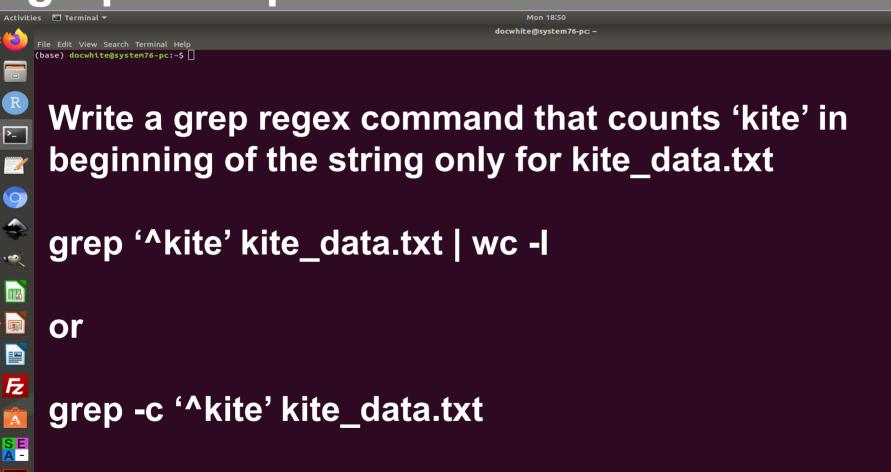
# grep [option] pattern file

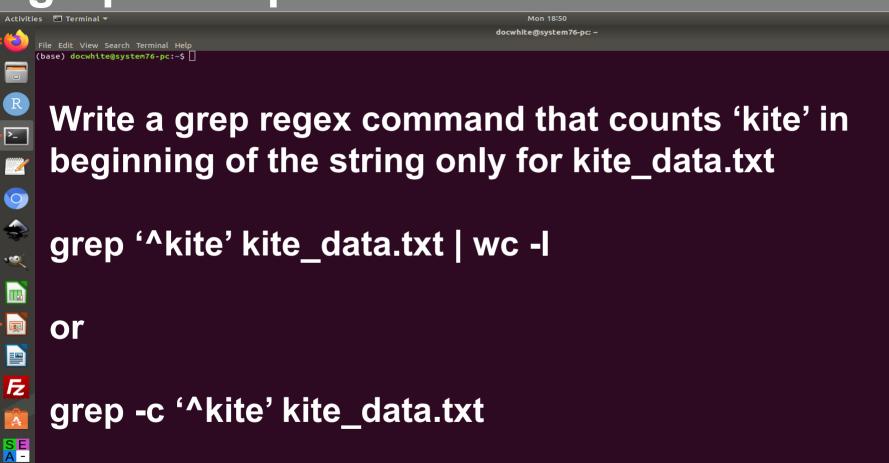
**Understanding Regular Expressions:** 

- ^ (Caret) match expression at the start of a line, as in ^A.
- \$ (Question) match expression at the end of a line, as in A\$.
- \ (Back Slash) turn off the special meaning of the next character, as in \^. To look for a Caret "^" at the start of a line, the expression is ^\^.
- [] (Brackets) match any one of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9].
- [^] match any one character except those enclosed in [], as in [^0-9].
- . (Period) match a single character of any value, except end of line. So b.b will match "bob", "bib", "b-b", etc.
- \* (Asterisk) match zero or more of the preceding character or expression. An asterisk matches zero or more of what precedes it. Thus [A-Z]\* matches any number of upper-case letters, including none, while [A-Z][A-Z]\* matches one or more upper-case letters.

Æ

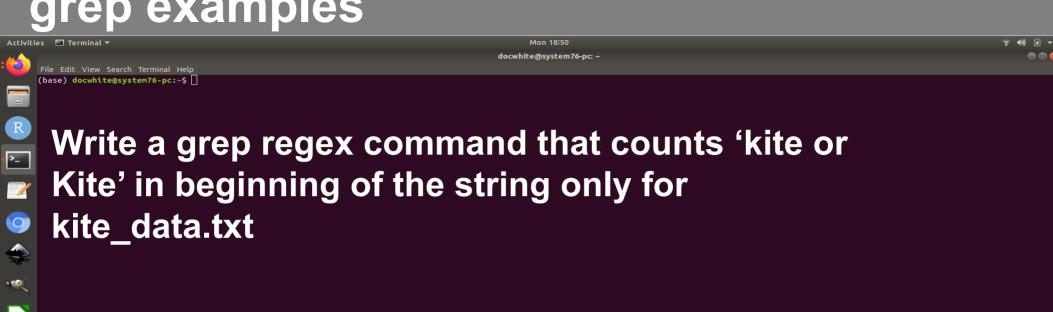


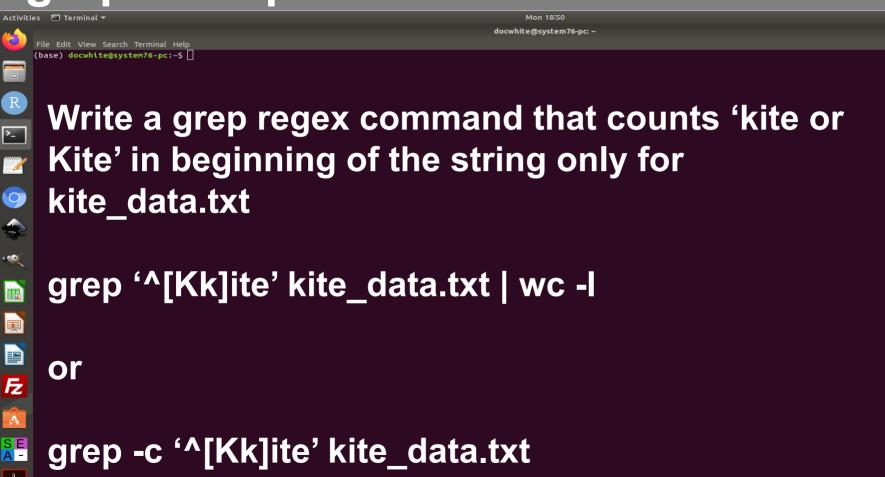




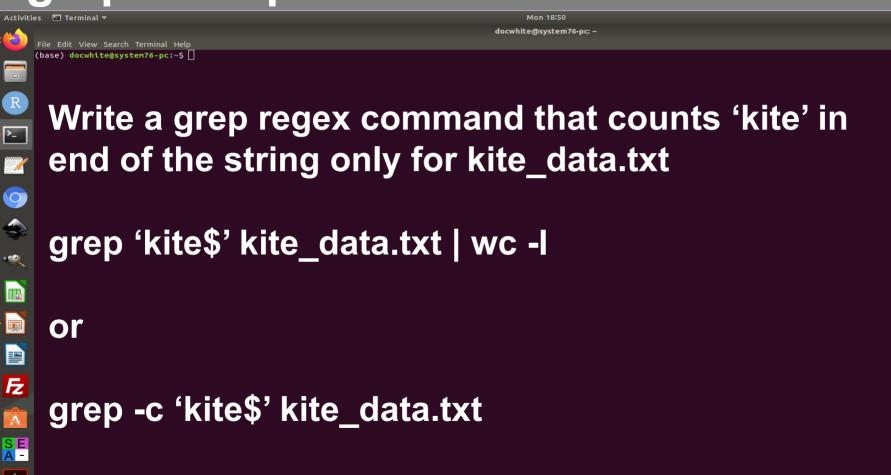
What does the -w option do instead of -c?

Æ



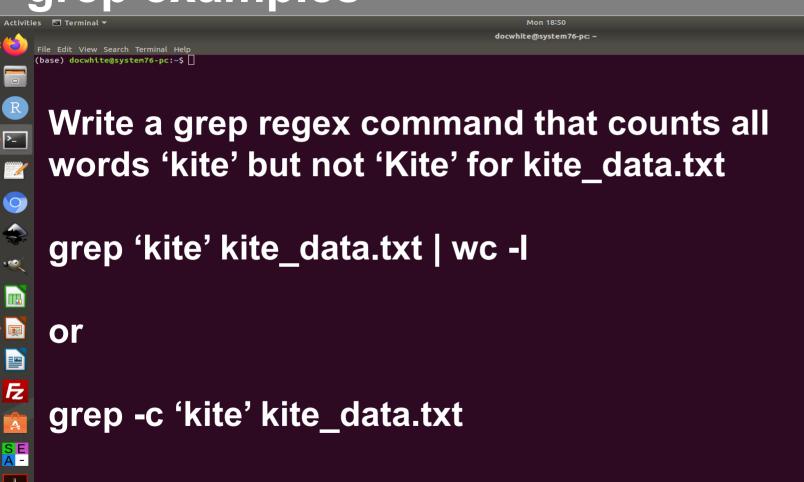




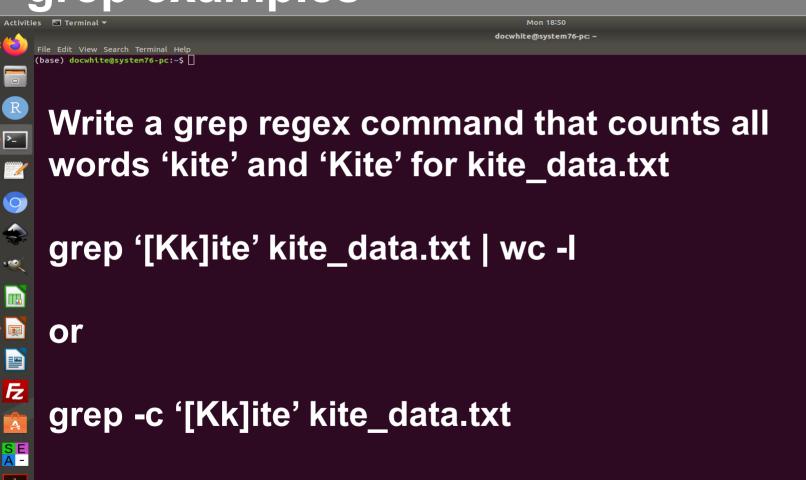


Æ



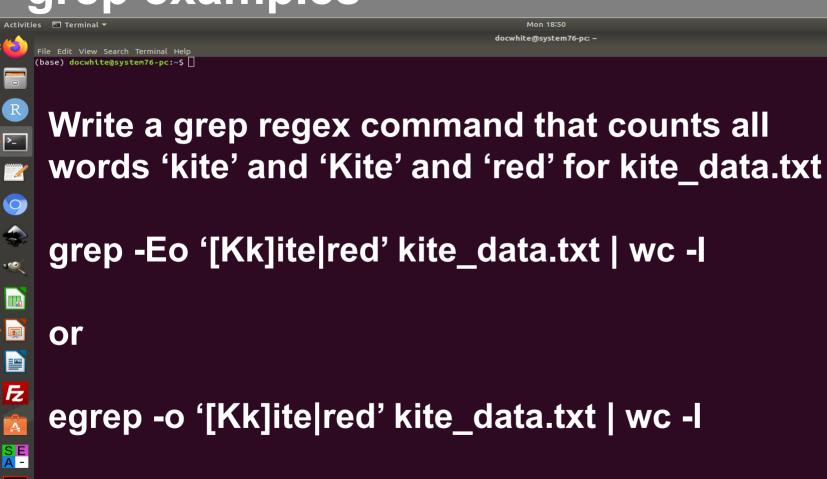




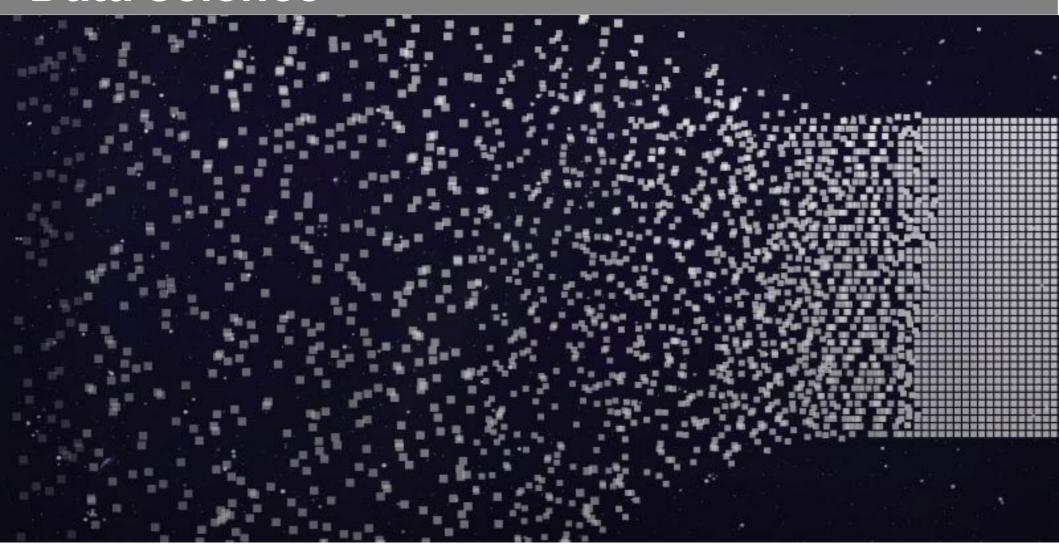


Æ





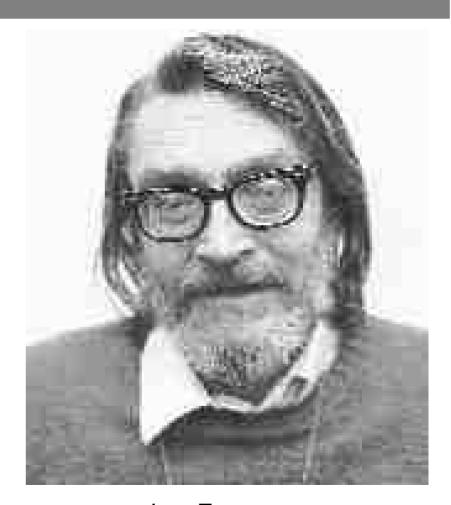
# Data science



# **Data science**



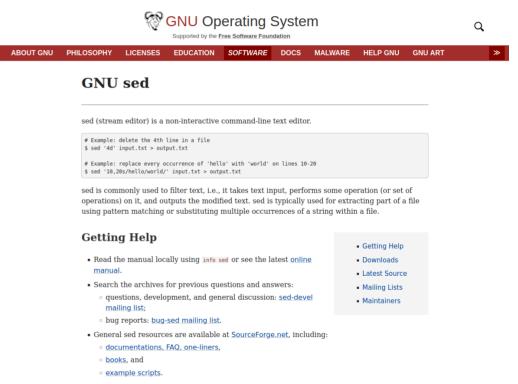
sed ("stream editor") is a Unix utility that parses and transforms text, using a simple, compact programming language. sed was developed from 1973-1974 by Lee E. McMahon of Bell Labs



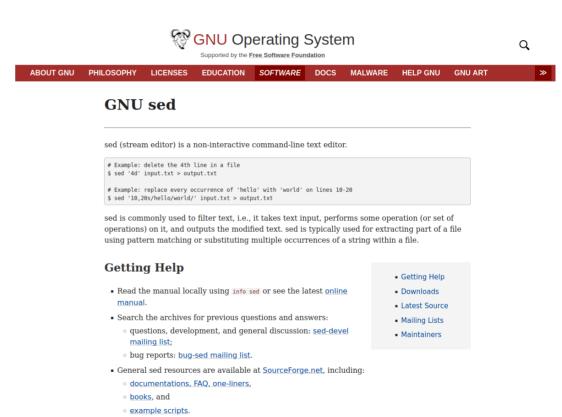
Lee E. McMahon

sed ("stream editor") is sed was one of the earliest tools to support regular expressions, and remains in use for text processing, most notably with the substitution command. Written in C

www.gnu.org/software/sed/



Popular alternative tools for plaintext string manipulation and "stream editing" include AWK and Perl.



s/he/she/g

#### **Sed - Stream Editor**

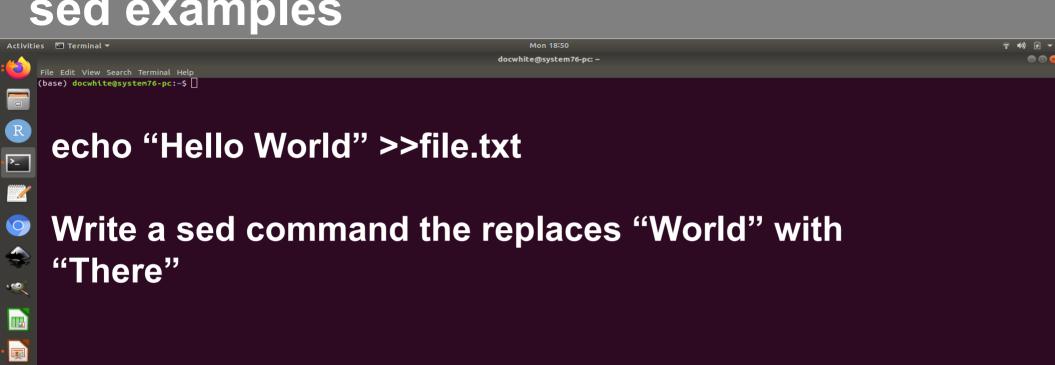
#### sed OPTIONS... [SCRIPT] [INPUTFILE...]

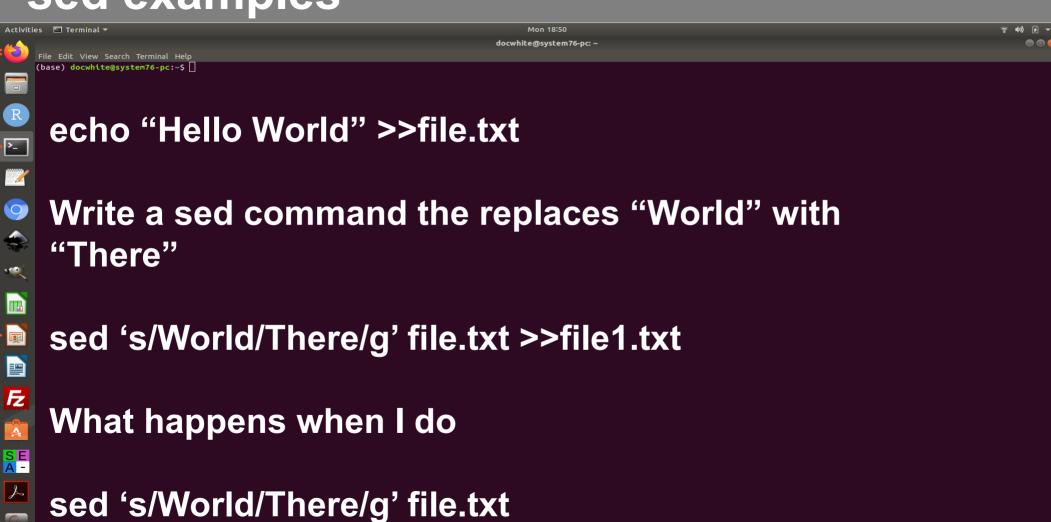
- --version (Print out the version of sed)
- --help (help page)
- -n, --quiet, --silent (suppress automatic printing of pattern space)
- --debug
- -e script, --expression=script (Add the commands in script to the set of commands to be run)
- -i [SUFFIX], –in-place[=SUFFIX] (This option specifies that files are to be edited in-place)

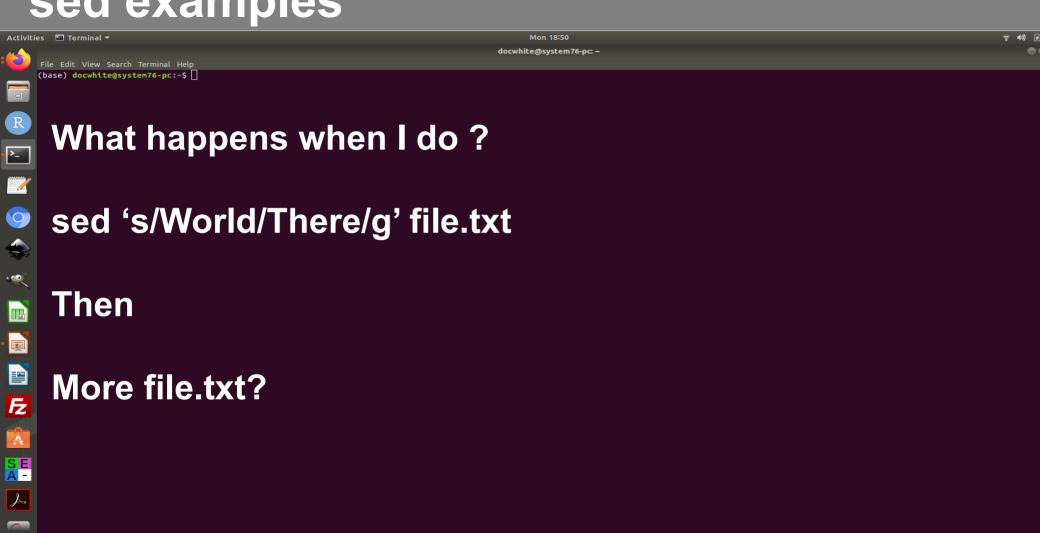
#### **Sed - Stream Editor**

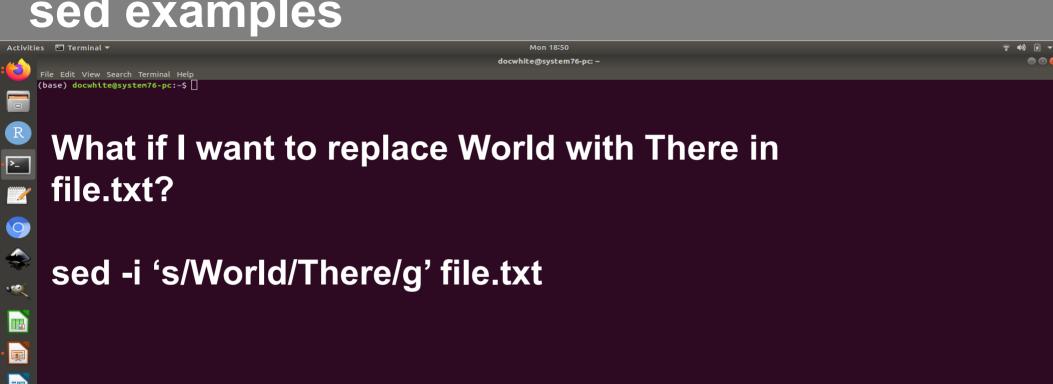
Substitution command
S command swiss army knife
sed 's/regexp/replacement/g' inputFileName >
outputFileName

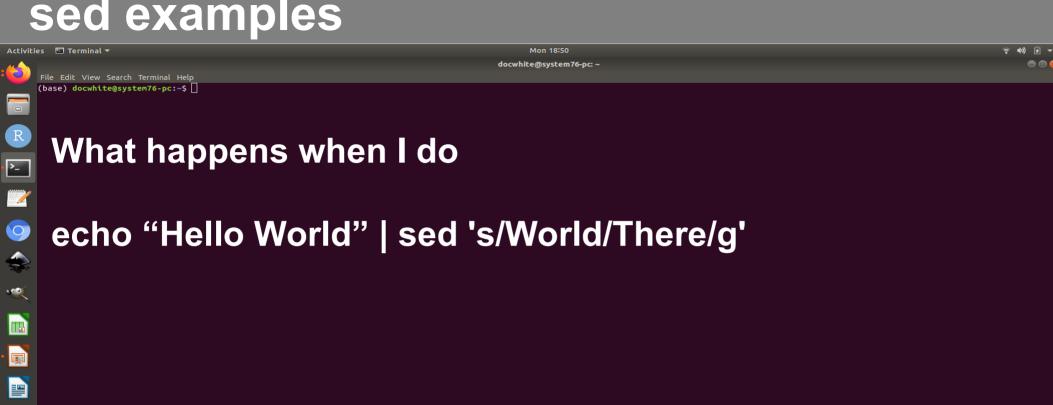
The s stands for substitute, while the g stands for global, which means that all matching occurrences in the line would be replaced.



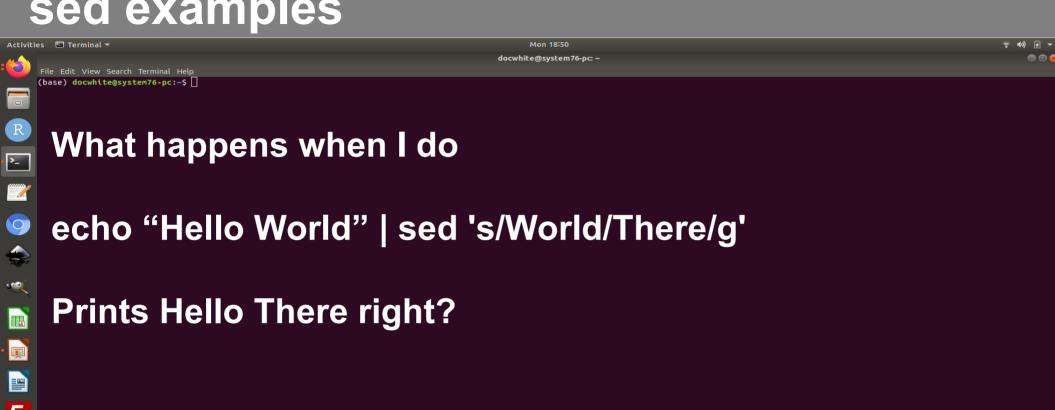








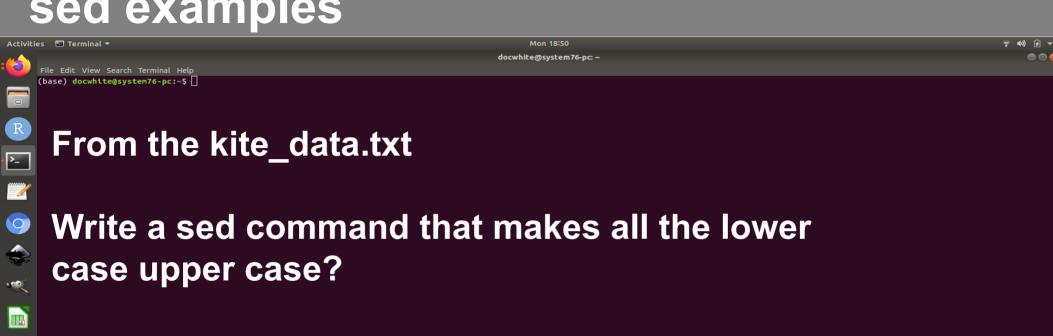
SE A-

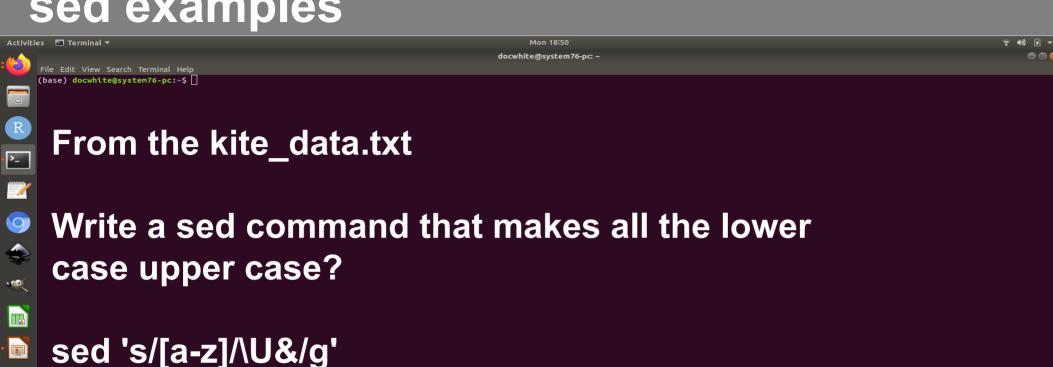


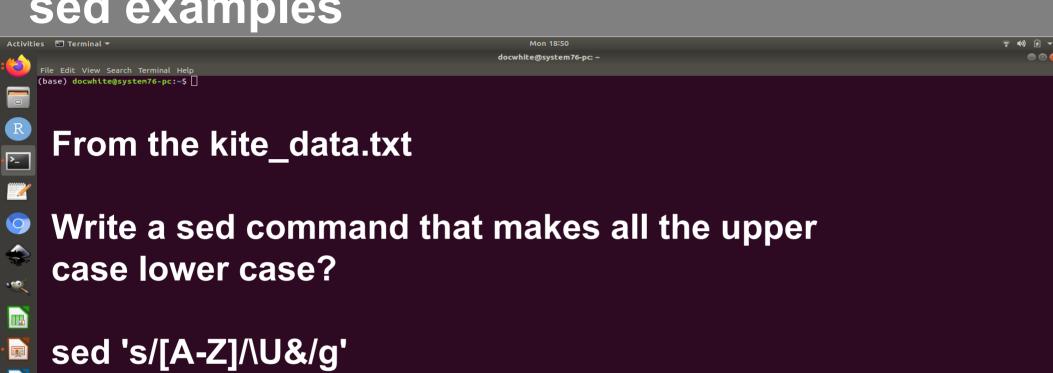
#### Sed – S swiss army knife

# sed 's/regexp/replacement/flags'.

- \L Turn the replacement to lowercase until a \U or \E is found
- \I Turn the next character to lowercase,
- \U Turn the replacement to uppercase until a \L or \E is found,
- \u -Turn the next character to uppercase,
- \E Stop case conversion started by \L or \U.
- g Apply the replacement to all matches to the regexp, not just the first.
- d Delete the pattern space; immediately start next cycle.
- # a comment, until the next newline.



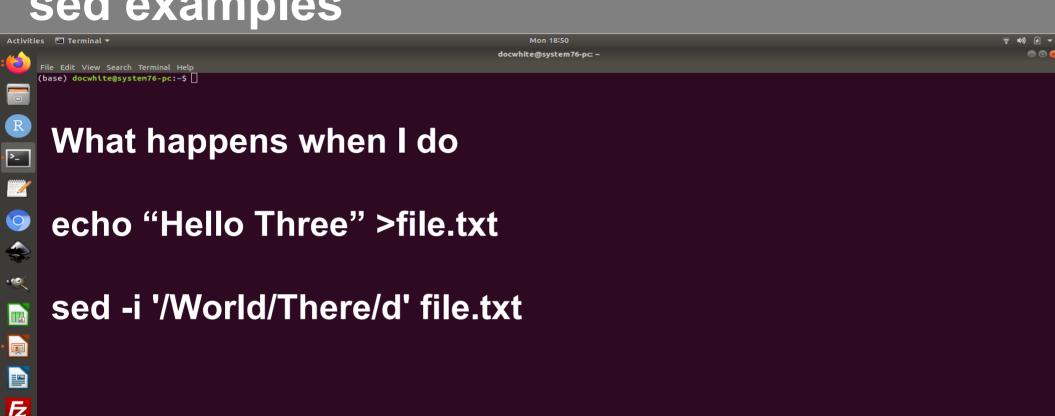




#### Sed – delete

#### sed '/^ \*\$/d' inputFileName.

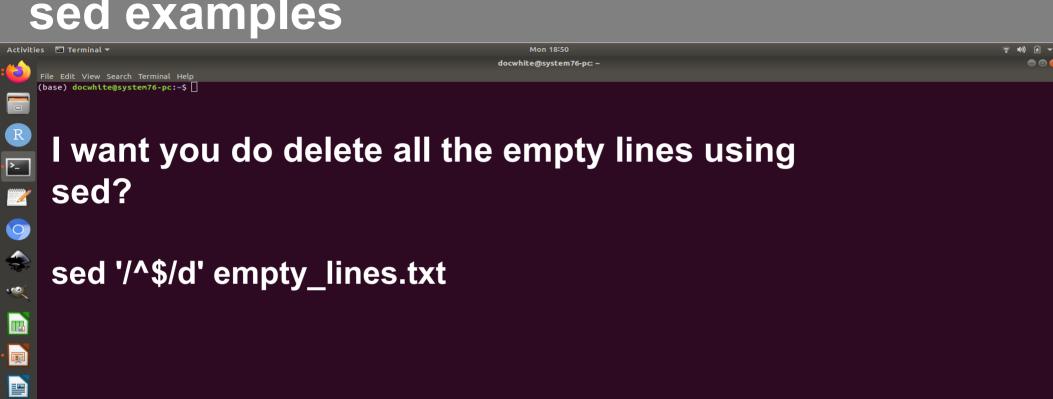
- The caret (^) matches the beginning of the line.
- The dollar sign (\$) matches the end of the line.
- The asterisk (\*) matches zero or more occurrences of the previous character.
- The plus (+) matches one or more occurrence(s) of the previous character.
- The question mark (?) matches zero or one occurrence of the previous character.
- The dot (.) matches exactly one character.





Æ

SE A-



#### Bonus 4

- Delete all the empty lines in the empty lines file with
- → grep
- $\rightarrow$  awk
- Delete all the 'all white space' with grep
- $\rightarrow$  grep
- $\rightarrow$  awk

Also, in python (think Pandas)

# Quiz 4

- On canvas now