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



Richard Allen White III, PhD RAW Lab Lecture 4 - Thursday Sep 2nd, 2021

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?

New question

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

New question

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

Bonus 2

- 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?

Bonus 2

- 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 2 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 2 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

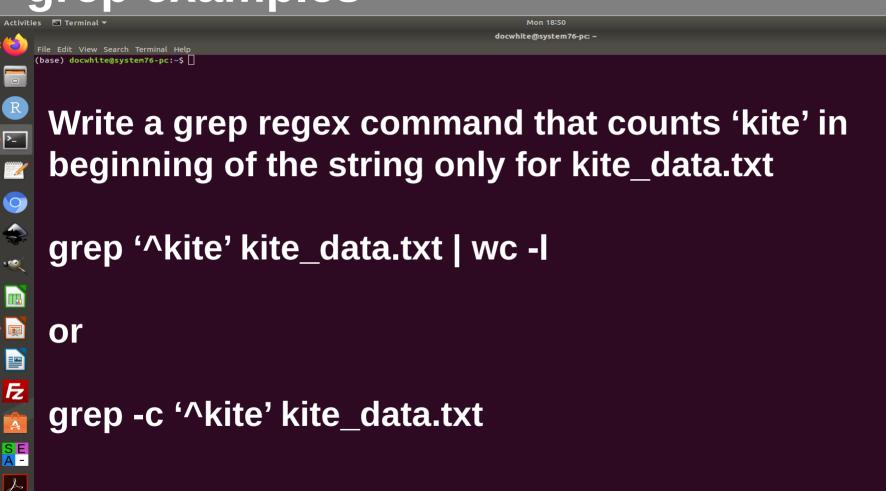
grep – syntax to hands of UNIX

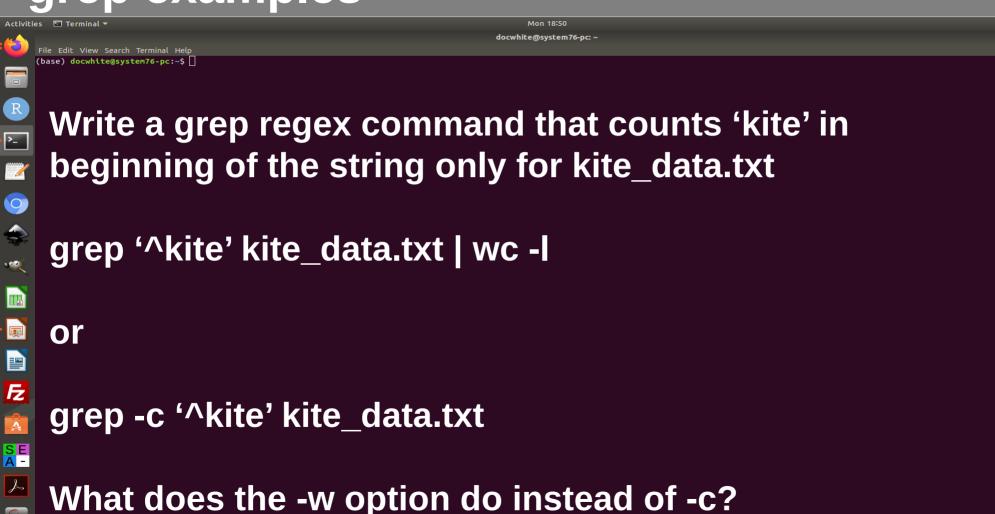
grep [option] pattern file

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.

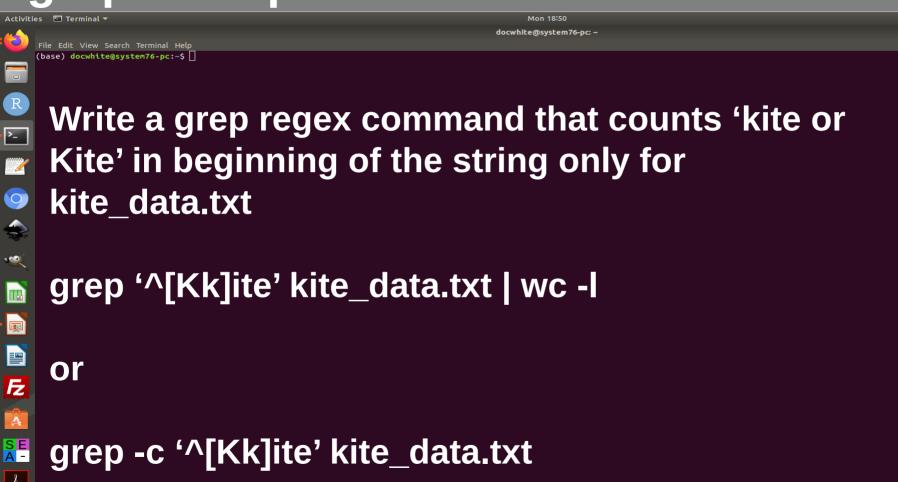




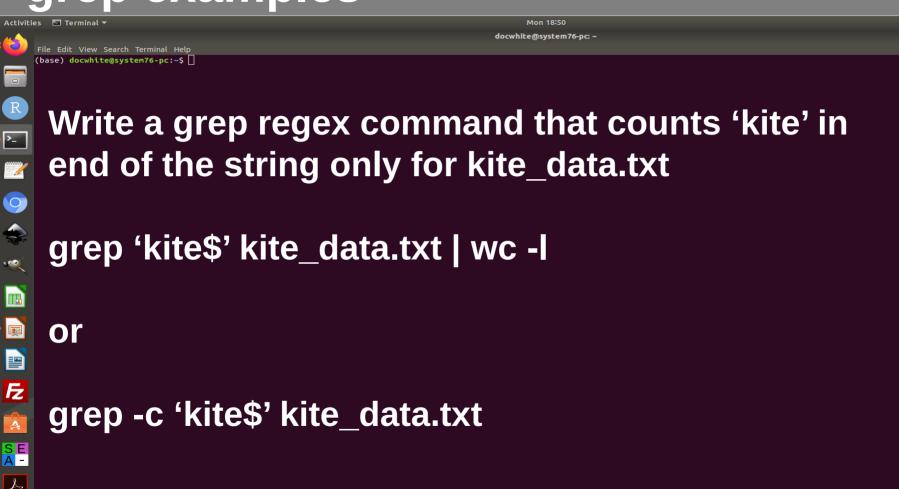


Æ



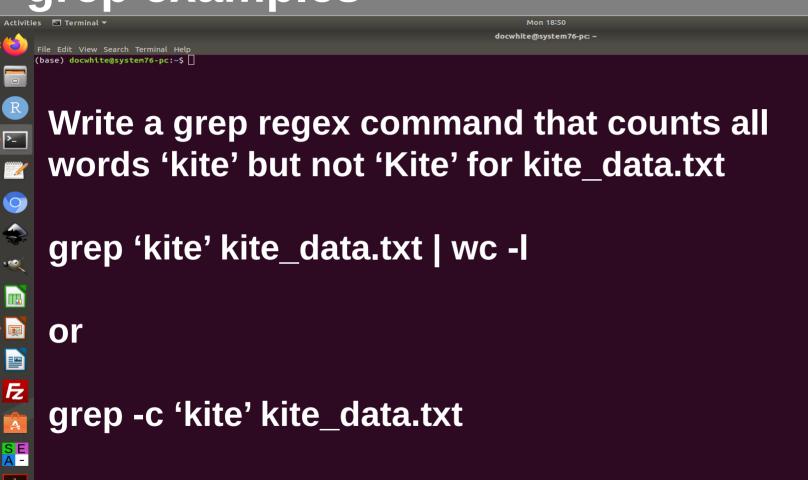




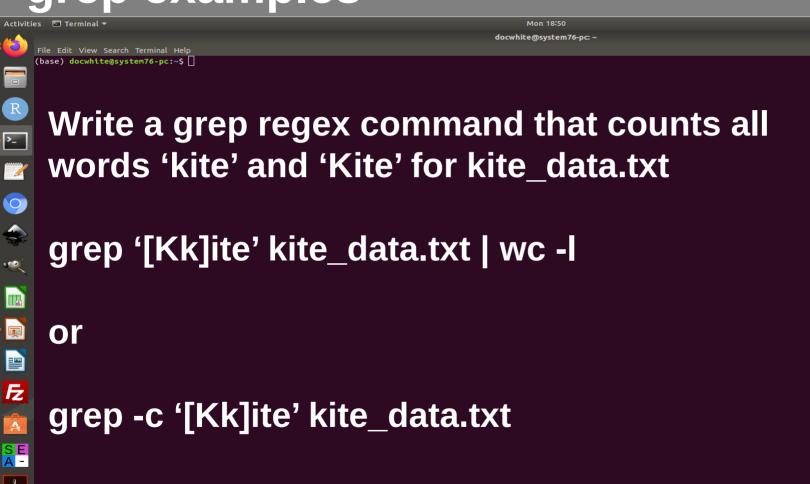


Æ



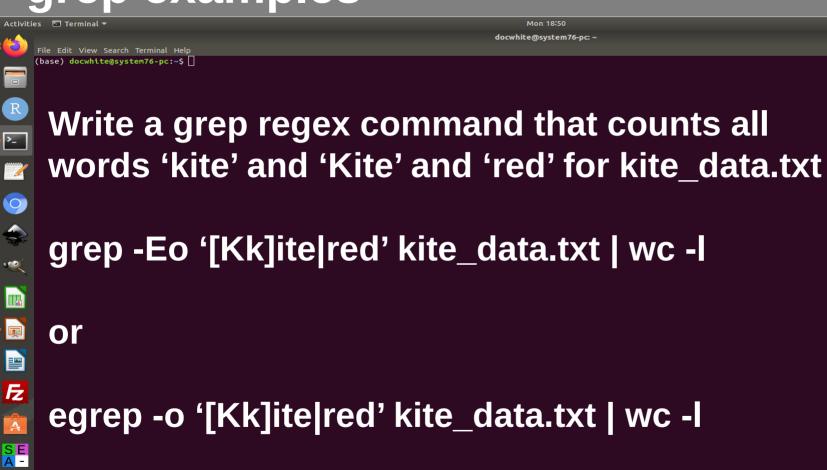




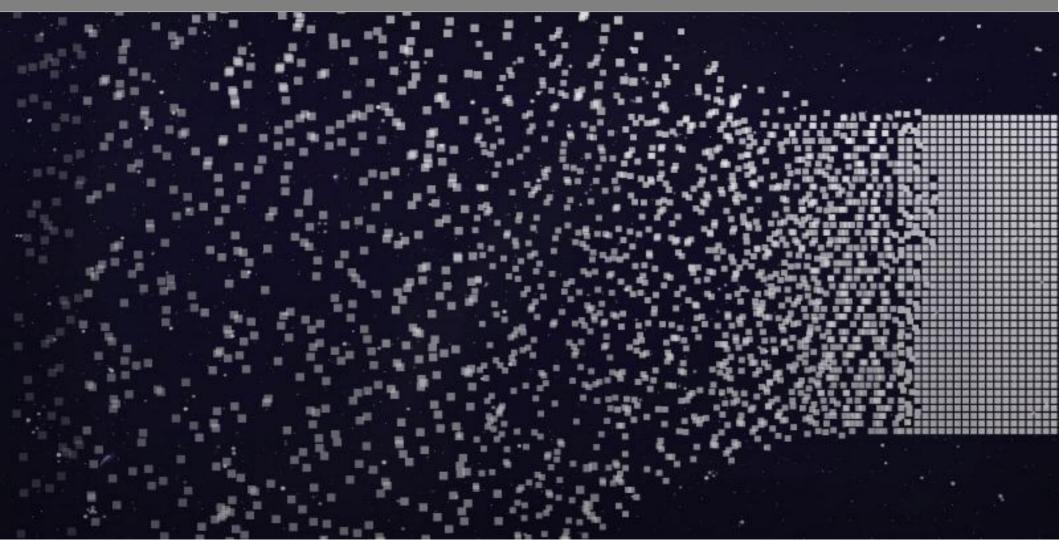


Æ





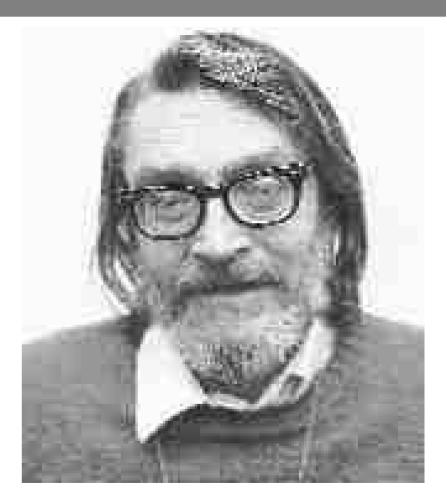
Data science



Data science

90% of data science is DATA WRANGLING

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

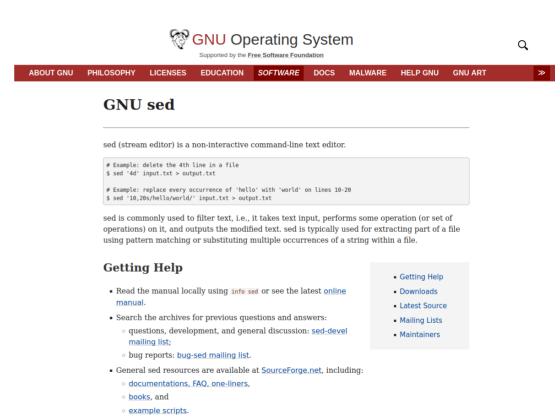
sed ("stream editor") is sed was one of the earliest tools to support regular expressions, and remains in use for text processing, most notably with the substitution command. Written in C

www.gnu.org/software/sed/

GNU Operating System Q SOFTWARE GNU sed sed (stream editor) is a non-interactive command-line text editor # Example: delete the 4th line in a file # Example: replace every occurrence of 'hello' with 'world' on lines 10-20 \$ sed '10.20s/hello/world/' input.txt > output.txt sed is commonly used to filter text, i.e., it takes text input, performs some operation (or set of operations) on it, and outputs the modified text, sed is typically used for extracting part of a file using pattern matching or substituting multiple occurrences of a string within a file. Getting Help Getting Help Read the manual locally using info sed or see the latest online Downloads manual. Latest Source • Search the archives for previous questions and answers: Mailing Lists questions, development, and general discussion: sed-devel Maintainers bug reports: bug-sed mailing list. General sed resources are available at SourceForge.net, including: o documentations, FAO, one-liners, books, and

example scripts

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



s/he/she/g

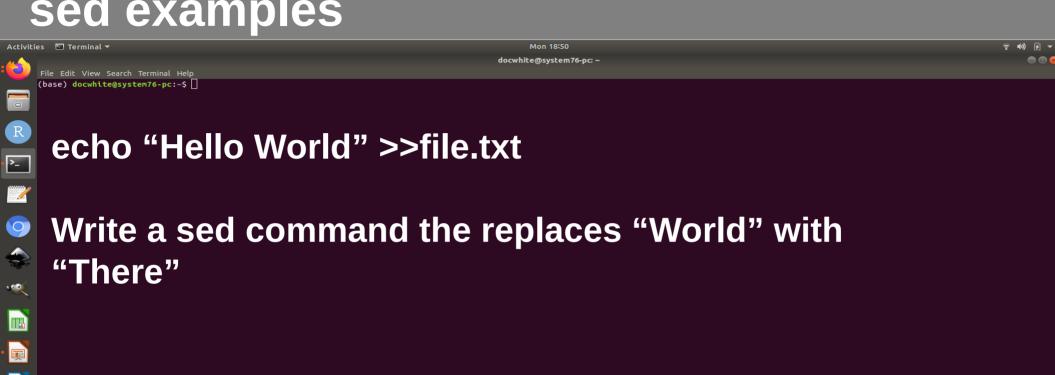
sed OPTIONS... [SCRIPT] [INPUTFILE...]

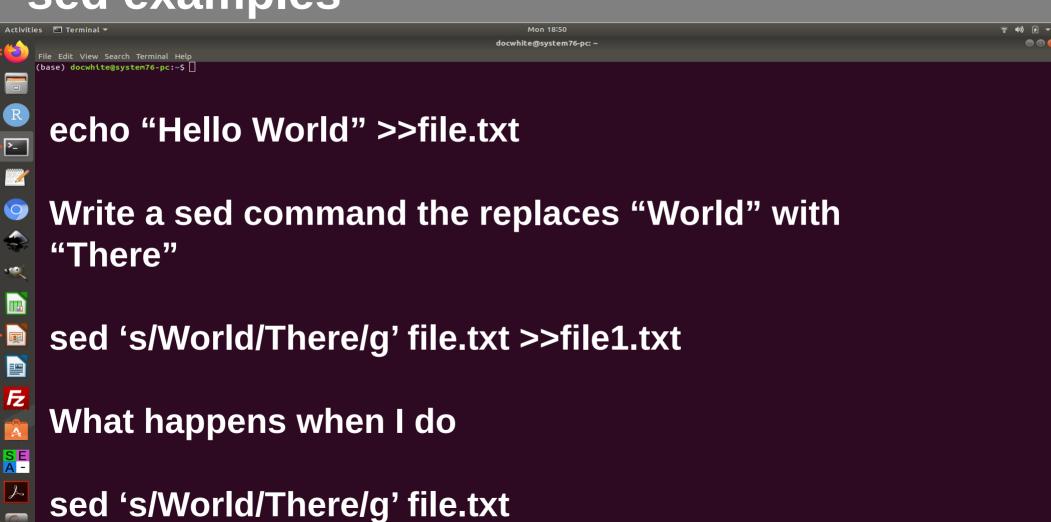
- --version (Print out the version of sed)
- --help (help page)
- -n, --quiet, --silent (suppress automatic printing of pattern space)
- --debug
- -e script, --expression=script (Add the commands in script to the set of commands to be run)
- -i [SUFFIX], –in-place[=SUFFIX] (This option specifies that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

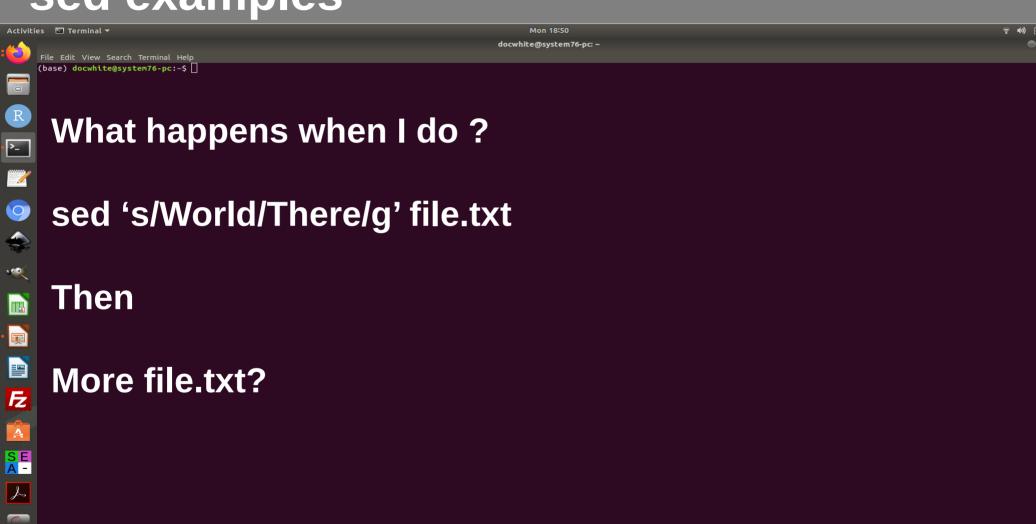
Substitution command S command swiss army knife sed 's/regexp/replacement/g' inputFileName > outputFileName

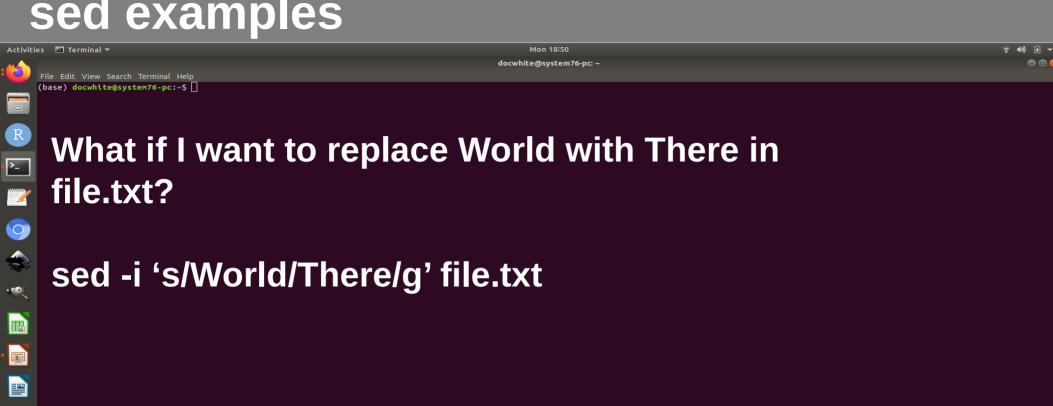
The **s** stands for substitute, while the **g** stands for global, which means that all matching occurrences in the line would be replaced.

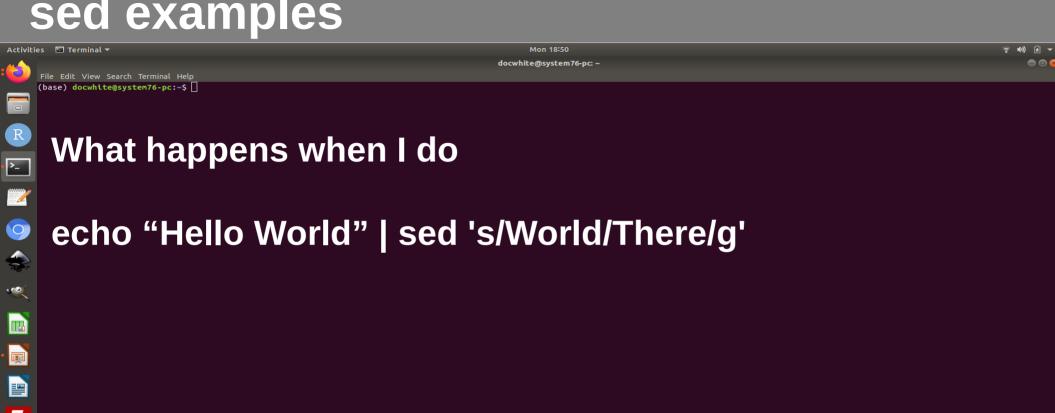
Æ

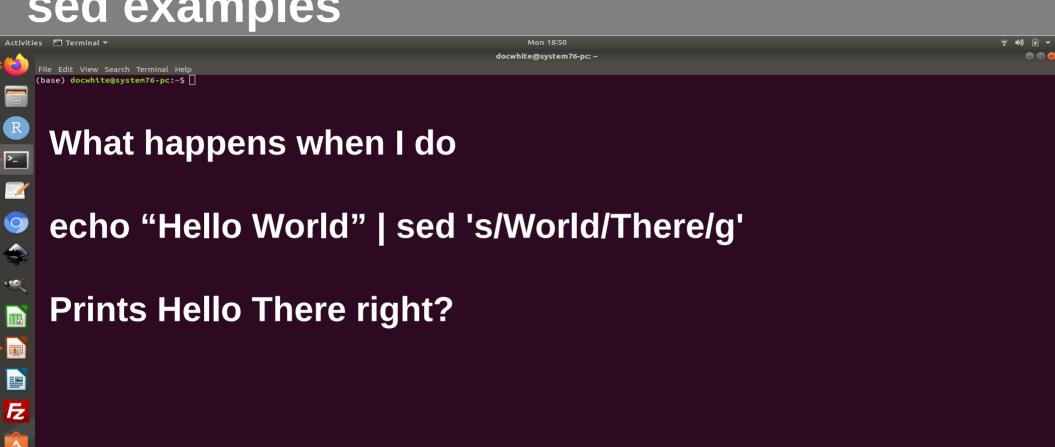








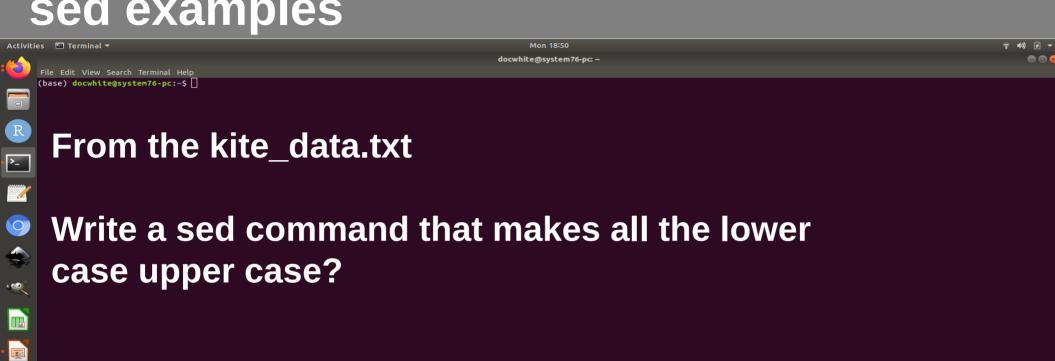




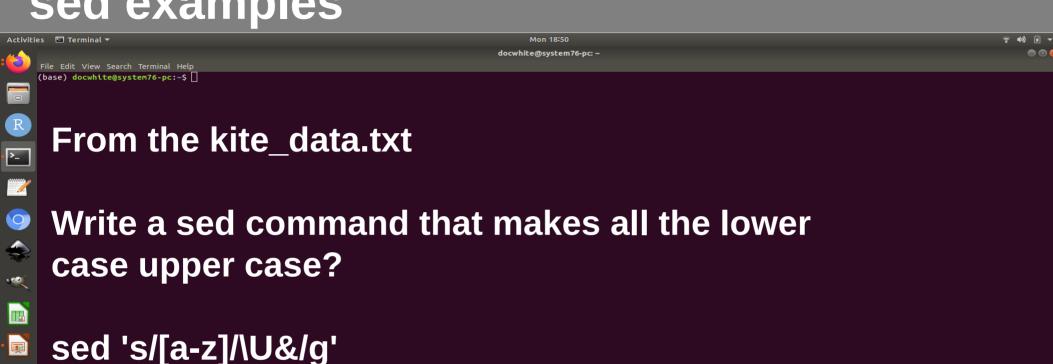
Sed – S swiss army knife

sed 's/regexp/replacement/flags'.

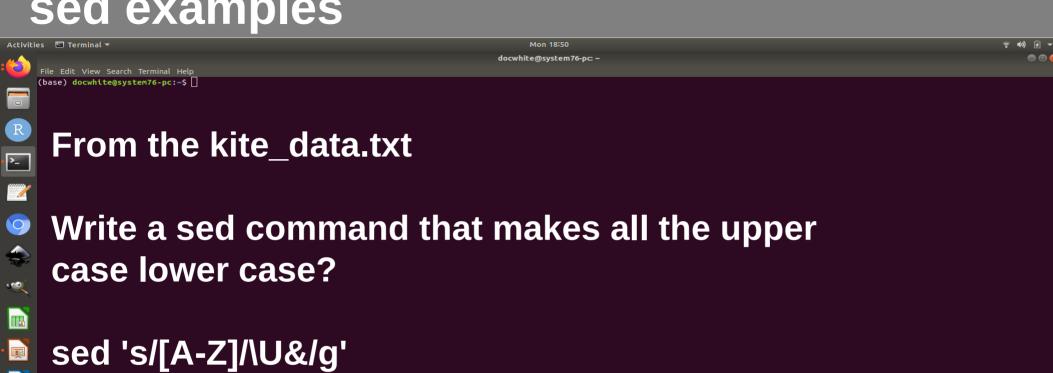
- \L Turn the replacement to lowercase until a \U or \E is found
- \I Turn the next character to lowercase,
- \U Turn the replacement to uppercase until a \L or \E is found,
- \u -Turn the next character to uppercase,
- \E Stop case conversion started by \L or \U.
- g Apply the replacement to all matches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



Æ



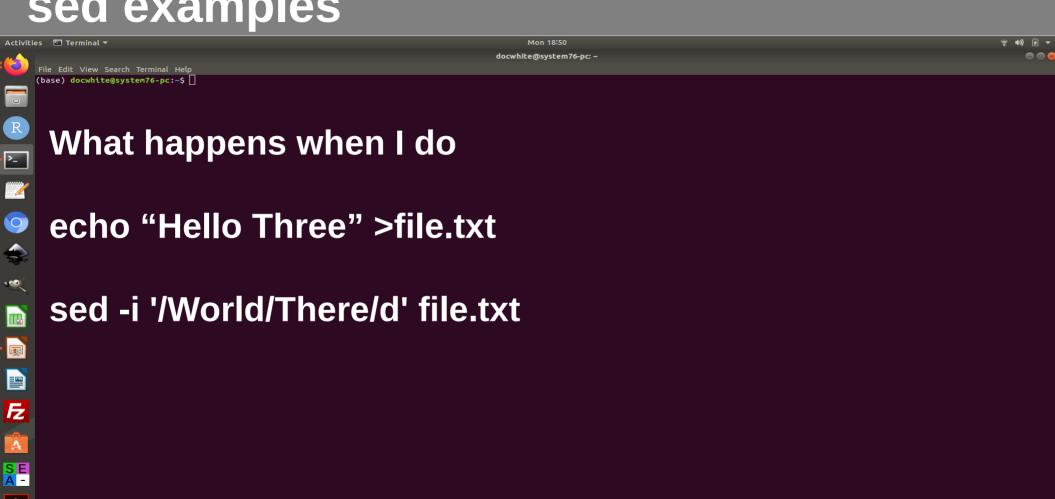
Æ



Sed – delete

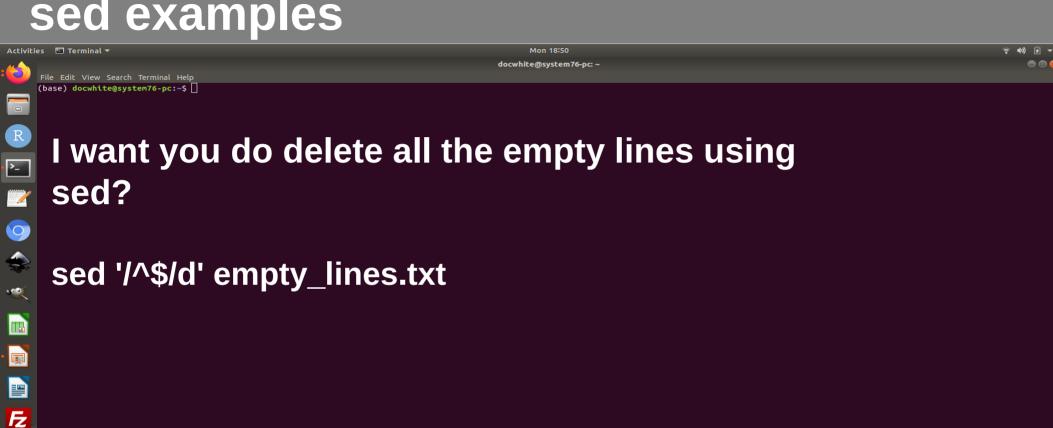
sed '/^ *\$/d' inputFileName.

- 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.





SE A-



Bonus 3

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

Also, in python (think Pandas)

Quiz 4

- On canvas now