BINF2111 – Introduction to Bioinformatics Computing

UNIX 101 part deux (Grep and regular exp)

Richard Allen White III, PhD

RAW Lab

Lecture 3 - Tuesday Aug 30th, 2022

Learning Objectives

- Review quiz and lab

- Grep

- Regular expressions in grep

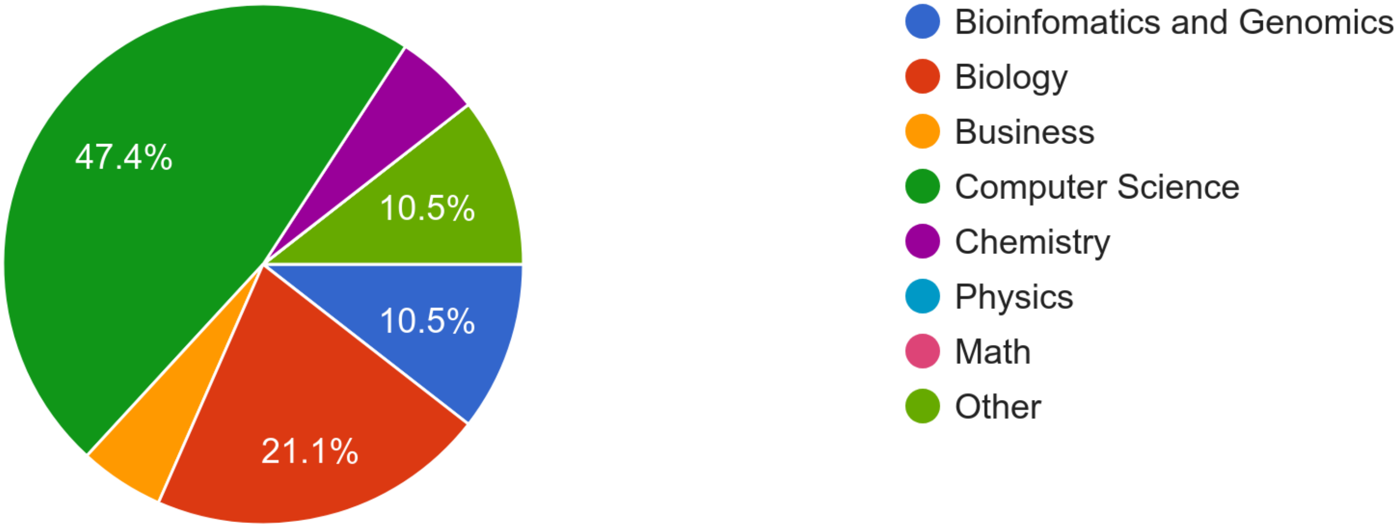
- Count nucleotide strings in grep

- Quiz 3

What is your major department ?

Which department?

19 responses



Bonus 1

- Create a file using a one-line command that prints “Hello World” six times?

Bonus 1

- Create a file using a one-line command that prints “Hello World” six times?

```
for i in {1..6}; do echo “Hello World”; done >>file.txt
```

What if I do this? What happens?

```
for i in {1..6}; do echo Hello World; done >file.txt
```

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

File is on the github
https://github.com/raw-lab/BINF2111/blob/main/data/example.fasta

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

```
cat example.fasta | grep “>” | wc -l
```

Can we do this better?

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

```
cat example.fasta | grep “>” | wc -l
```

Can we do this better?

```
grep “>” example.fasta | wc -l (better)
```

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

```
cat example.fasta | grep “>” | wc -l
```

Can we do this better?

```
grep “>” example.fasta | wc -l (better)
```

Even better?

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

```
cat example.fasta | grep “>” | wc -l
```

Can we do this better?

`grep “>” example.fasta | wc -l` (better)

Even better?

`grep -c “>” example.fasta` (**BEST**)

Quick exercise 1

- Write a single line UNIX to count the number of “>” in file

grep -c “>” example.fasta (**BEST**)

Whats the answer?

Count the number of “T’s” in the file?

Grep vs. Python

- Linux terminal (bash) commands:

```
grep '>' one.fasta | wc
–l
Or:
grep -c '>' one.fasta
```

- Python Script:

```
#!/usr/bin/env python

import sys

count = 0
with open(sys.argv[1]) as reader:
    for line in reader:
        if line.startswith('>'):
            count += 1
print(count)
```

- Python – One Line

```
#!/usr/bin/env python
import sys
print( len( [ x for x in open(sys.argv[1]) if x.startswith('>') ] )
 )
```

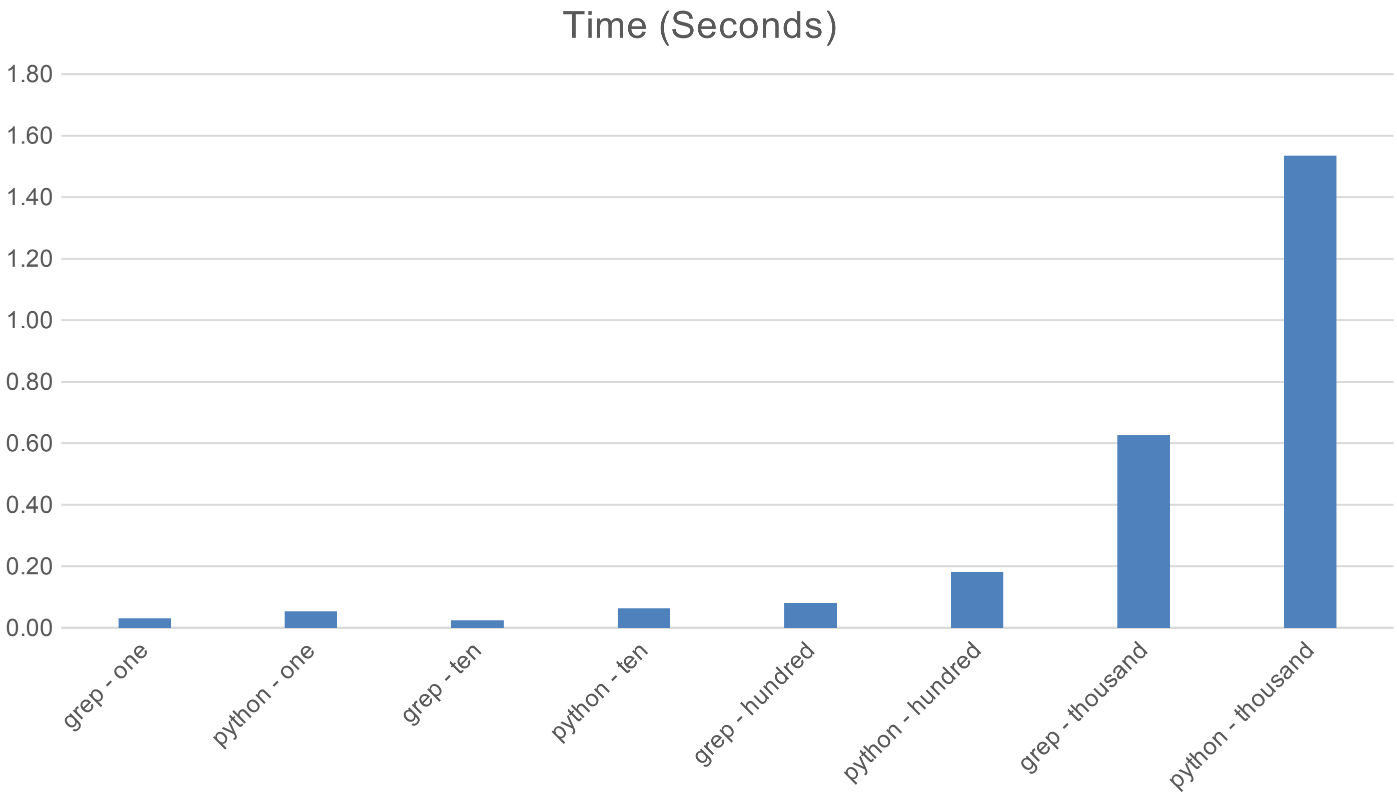
- Test script to time everything:

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

time grep -c '>' one.fasta
time grep -c '>' ten.fasta
time grep -c '>' hundred.fasta
time grep -c '>' thousand.fasta

time ./count.py one.fasta
time ./count.py ten.fasta
time ./count.py hundred.fasta
time ./count.py thousand.fasta
```

Grep vs. Python

Filename	Count
one.fasta	1041
ten.fasta	10604
hundred.fasta	131349
thousand.fasta	1857307

Test Name	Time (s)
grep - one	0.0301667
grep - ten	0.0243333
grep - hundred	0.0813333
grep - thousand	0.6248333
python - one	0.0525000
python - ten	0.0626667
python - hundred	0.1816667
python - thousand	1.5358333



grep – master command of UNIX

Today, I will show you how to use grep -‘*the hands of the UNIX gods*’

Chris Grassa Ph.D.
2010

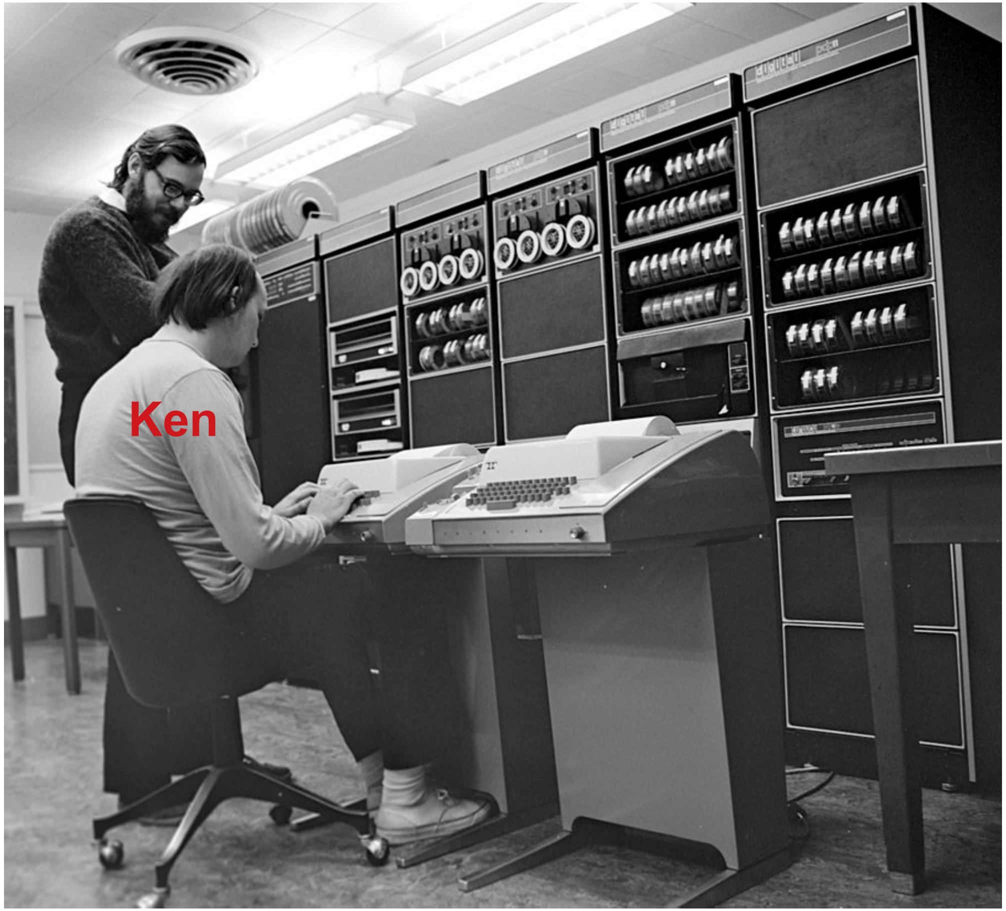
grep – master command of UNIX

grep =

grep – master command of UNIX

grep =

Ken Thompson
AT&T Bell Laboratories
Initial release - November 1973 (47 years ago)



grep – master command of UNIX

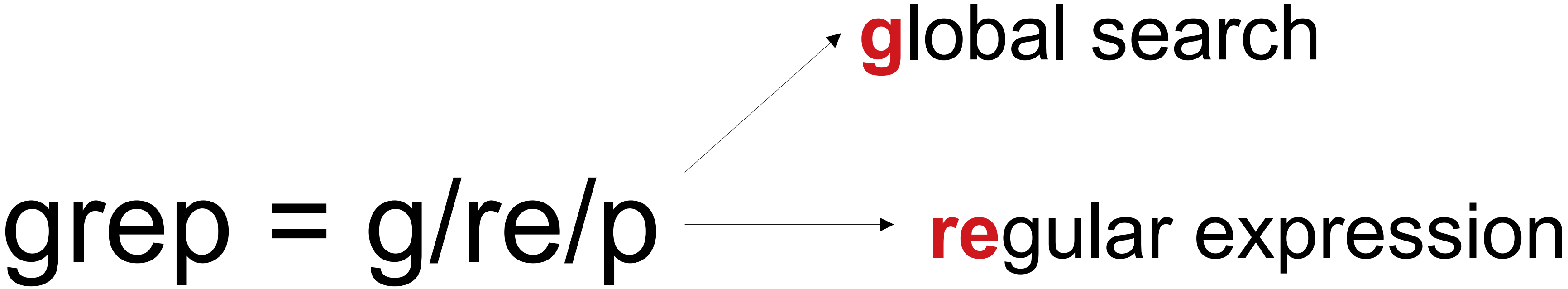
grep = g/re/p

grep – master command of UNIX

global search

grep = g/re/p

grep – master command of UNIX

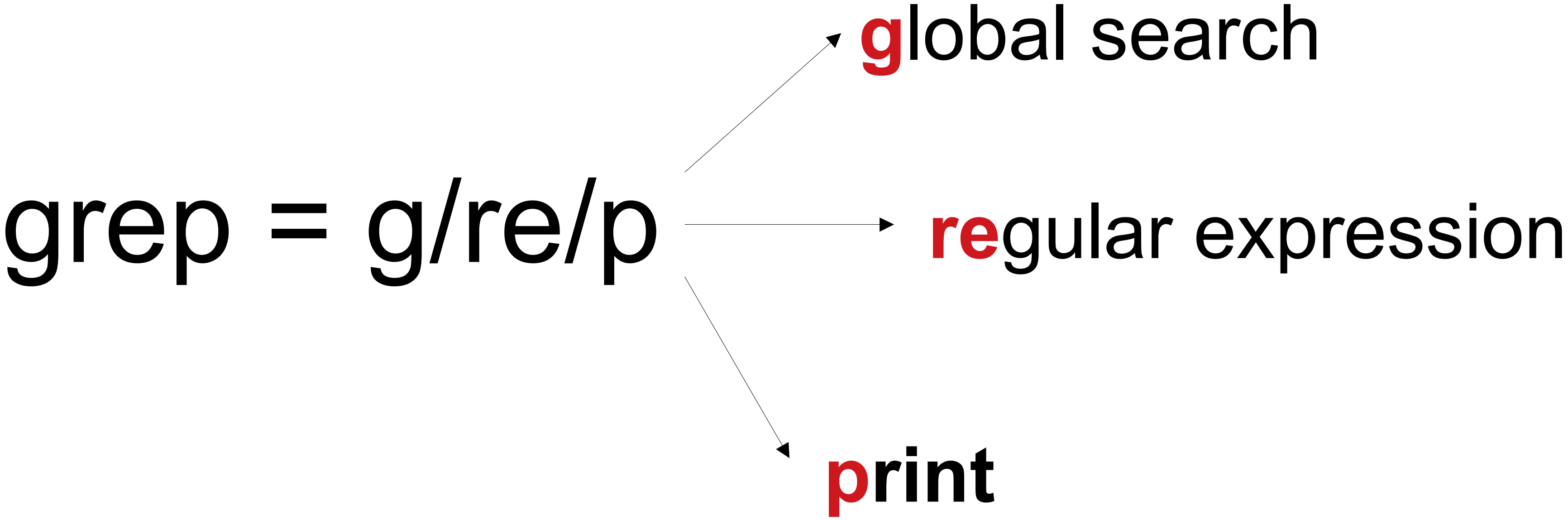


grep = g/re/p

global search

regular expression

grep – master command of UNIX



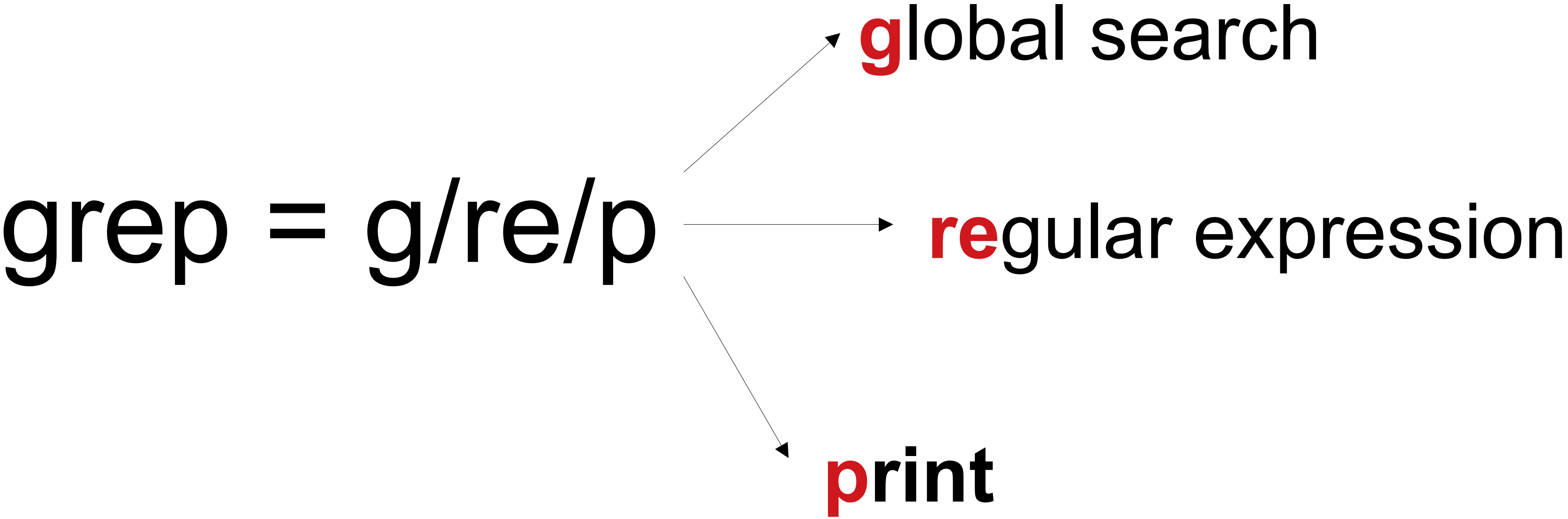
grep = g/re/p

global search

regular expression

print

grep – master command of UNIX



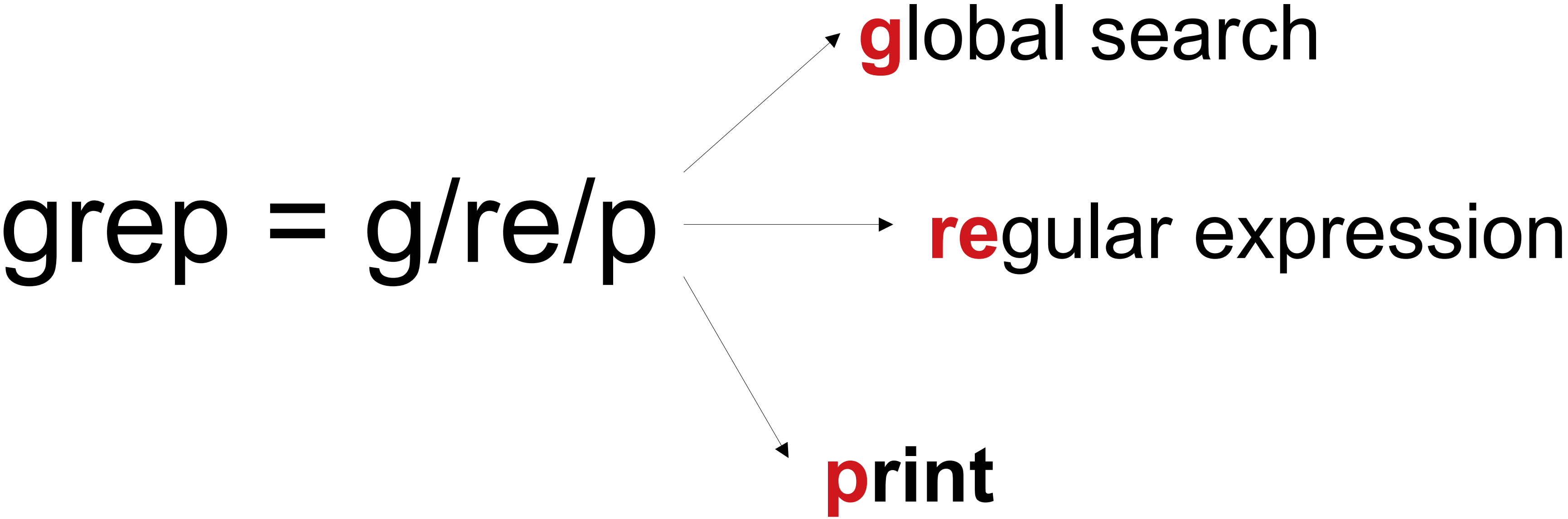
grep = g/re/p

global search

regular expression

print

grep – master command of UNIX



grep = g/re/p

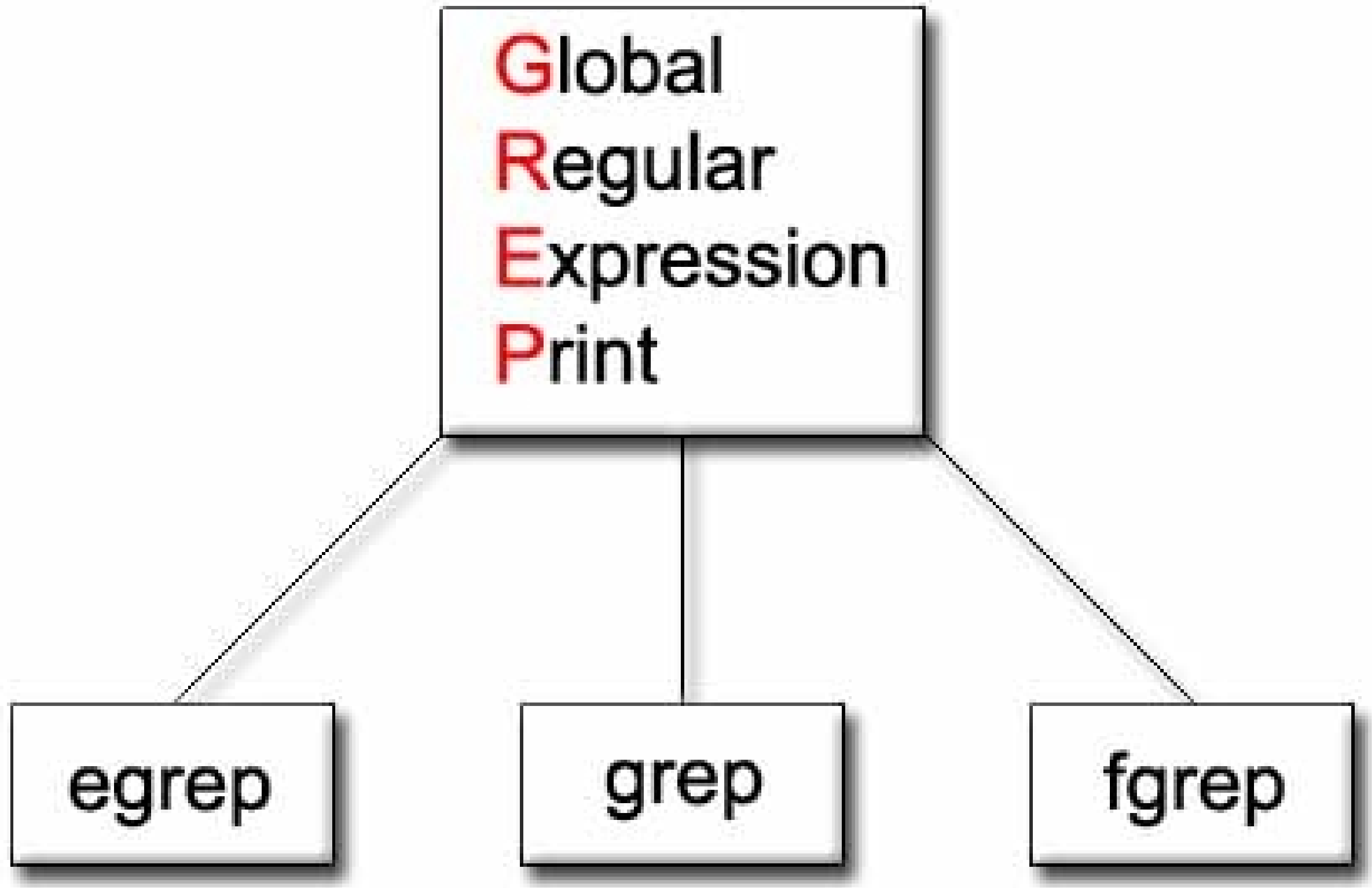
global search

regular expression

print

grep = **g**lobal search for **re**gular expression and **p**rint the result

grep – master command of UNIX



Global
Regular
Expression
Print

egrep

grep

fgrep

grep – command options

grep command options

-c Print only a count of the lines that contain the pattern.

-i Ignore upper/lower case distinction during comparisons.

-l Print only the names of file.txt with matching lines,separated by NEWLINE characters.
Does not repeat the names of file.txt when the pattern is found more than once.

-n Precede each line by its line number in the file (first line is 1).

-v Print all lines except those that contain the pattern.

-r It recursively search the pattern in all the file.txt in the current directory and all it’s subdirectory.

-w It searches the exact word

--color colors the matched text for easy visualization

-F interprets the pattern as a literal string

-H,-h print, don’t print the matched filename

-o only print the matching pattern

-x forces patterns to match the whole line

-E or -e provides extended functions to egrep (multiple exact matches)

grep – syntax to hands of UNIX

```
grep [option] pattern file
```

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.

grep – syntax to hands of UNIX

grep [option] pattern file

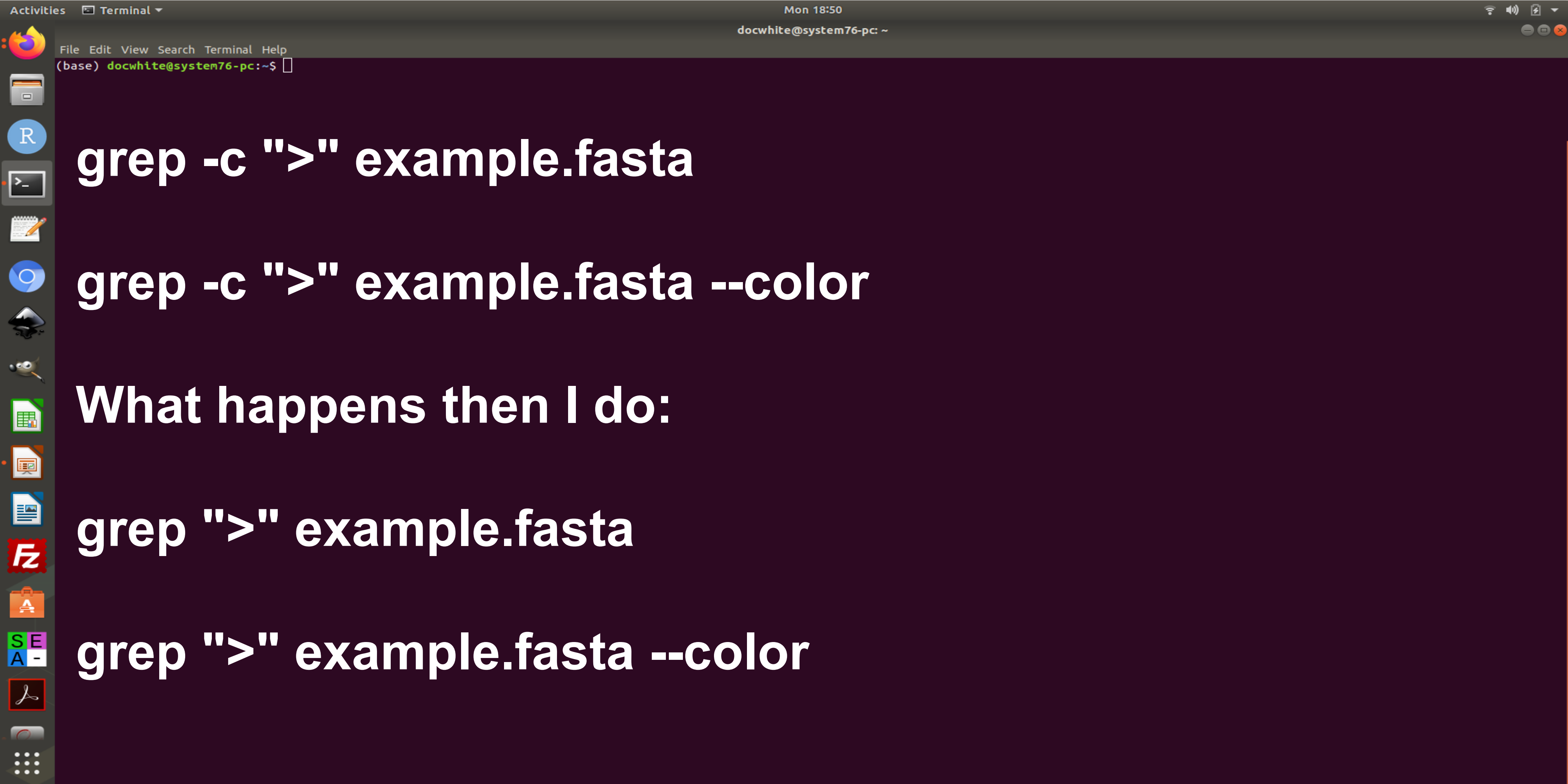
Kite rats kite cash REd kite
kite rats kite red caSh rats
kite rats kite caSh red green

grep ‘^kite’ file.txt | wc -l **(front of the line)**

grep ‘kite$’ file.txt | wc -l **(end of the line)**

grep ‘[Kk]ite’ file.txt | wc -l **(match all)**

grep examples



```
(base) docwhite@system76-pc:~$ 
```

grep -c ">" example.fasta

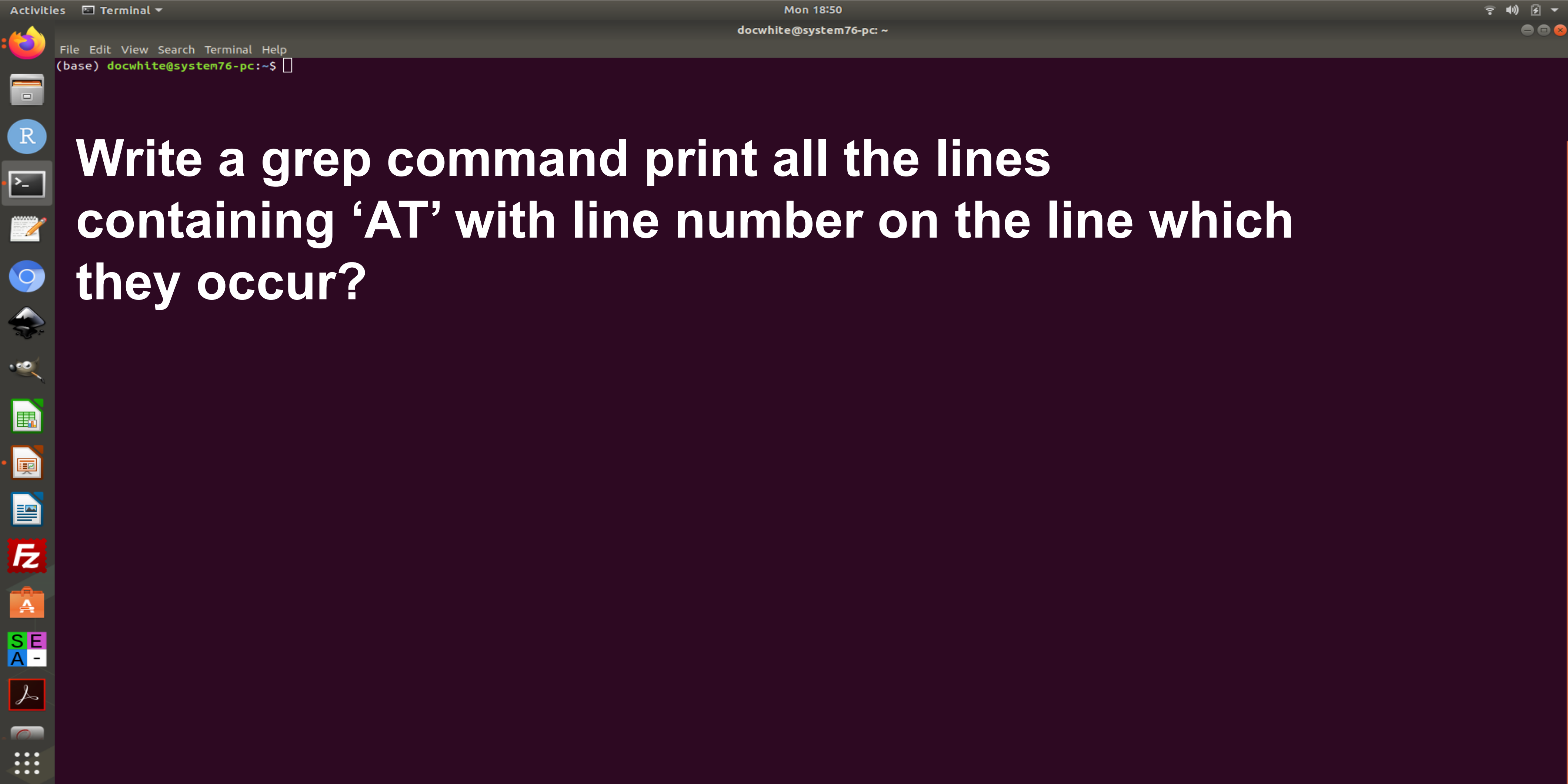
grep -c ">" example.fasta --color

What happens then I do:

grep ">" example.fasta

grep ">" example.fasta --color

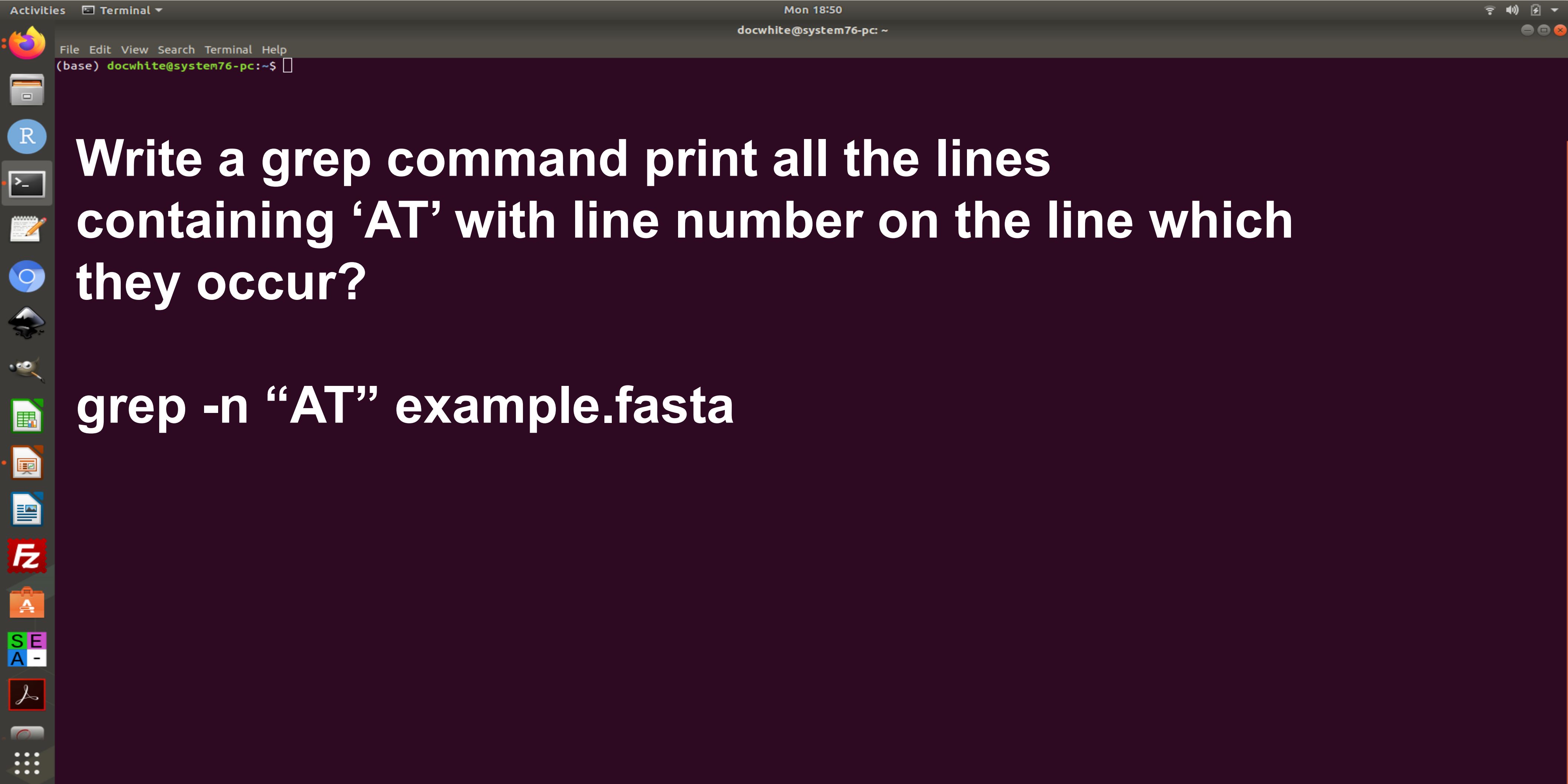
grep examples



```
(base) docwhite@system76-pc:~$ 
```

Write a grep command print all the lines containing ‘AT’ with line number on the line which they occur?

grep examples

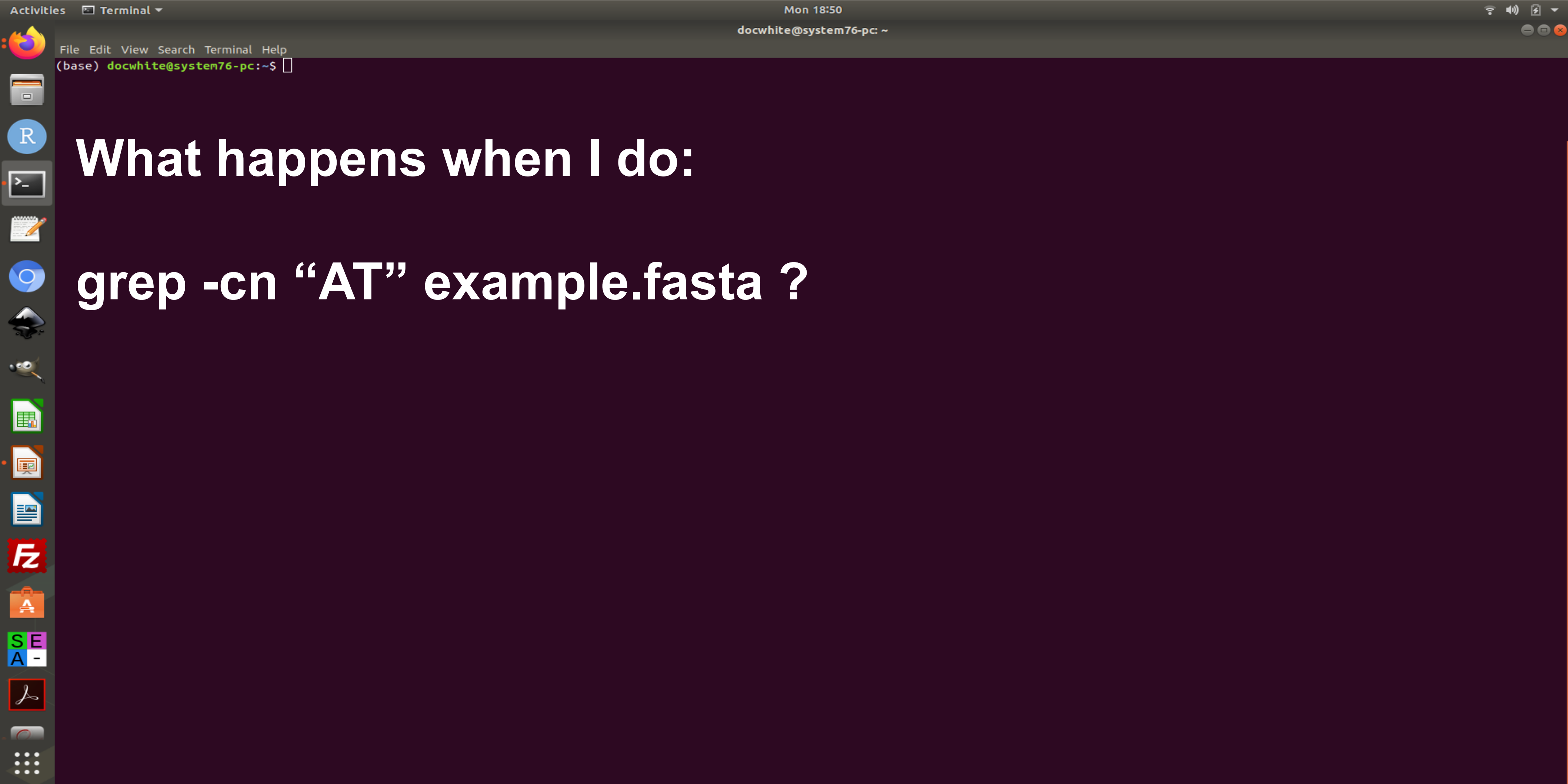


```
(base) docwhite@system76-pc:~$
```

Write a grep command print all the lines containing ‘AT’ with line number on the line which they occur?

grep -n “AT” example.fasta

grep examples

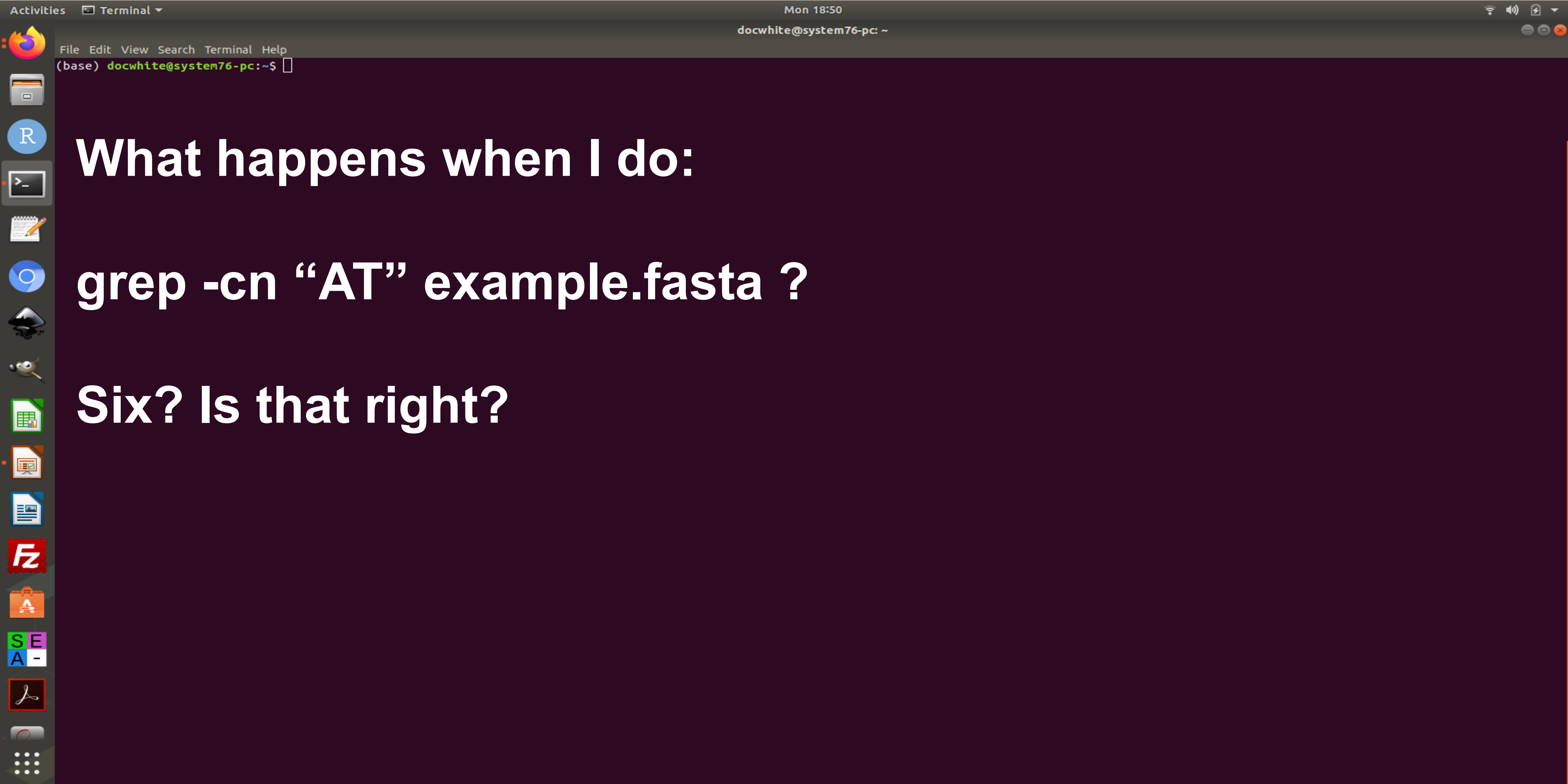


```
(base) docwhite@system76-pc:~$ 
```

What happens when I do:

grep -cn “AT” example.fasta ?

grep examples



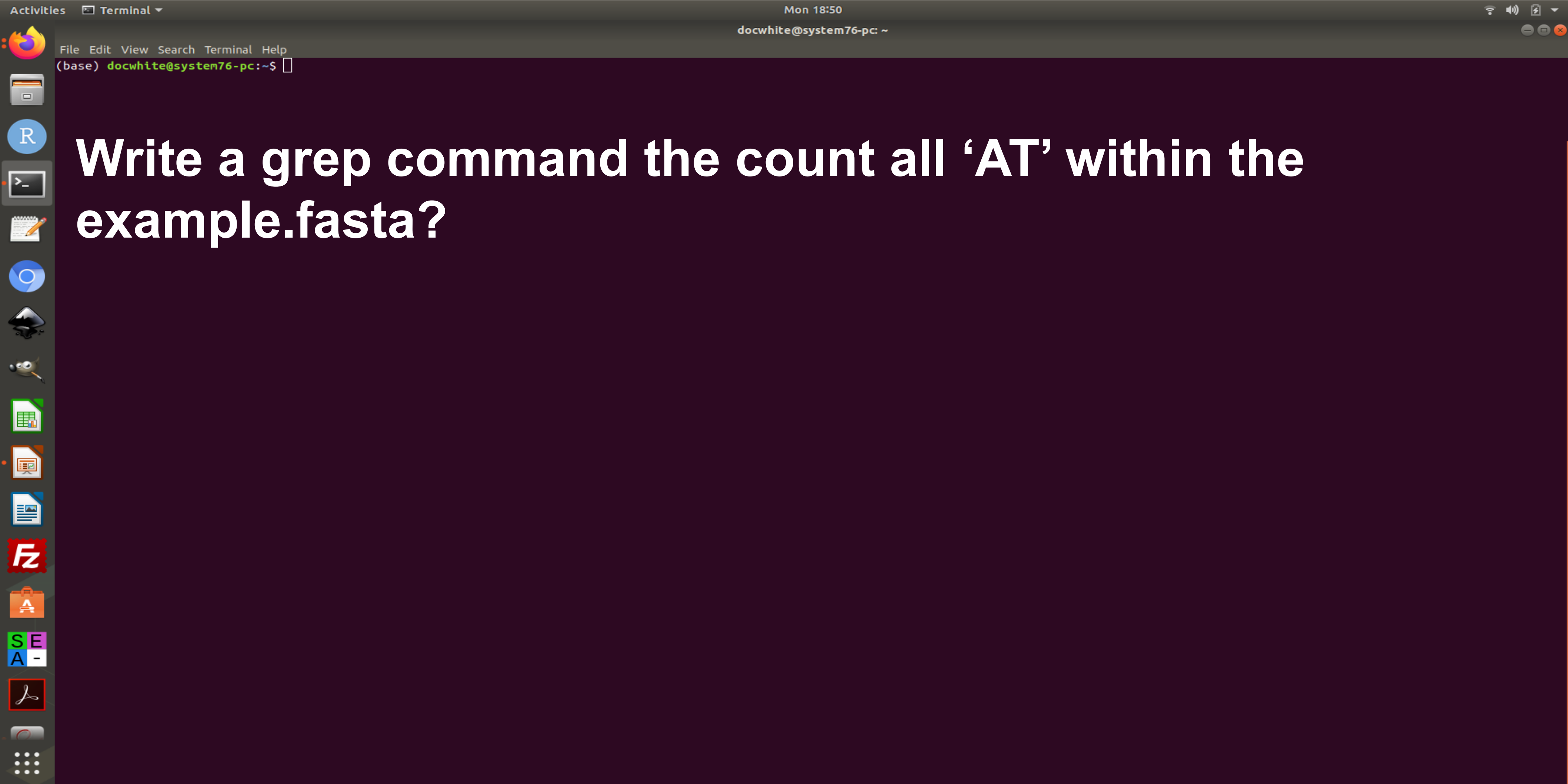
```
(base) docwhite@system76-pc:~$
```

What happens when I do:

grep -cn “AT” example.fasta ?

Six? Is that right?

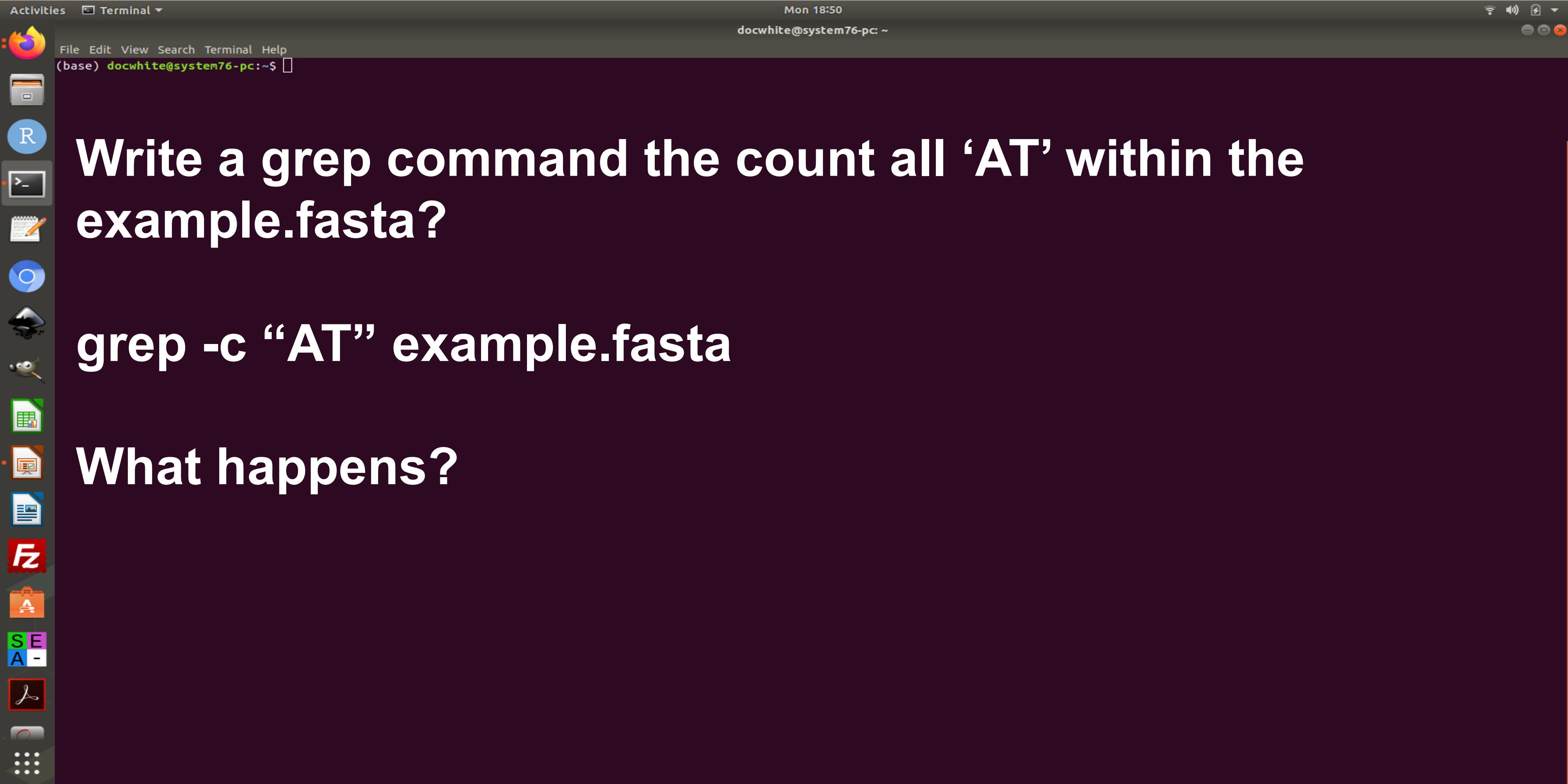
grep examples



```
(base) docwhite@system76-pc:~$ 
```

Write a grep command the count all ‘AT’ within the example.fasta?

grep examples



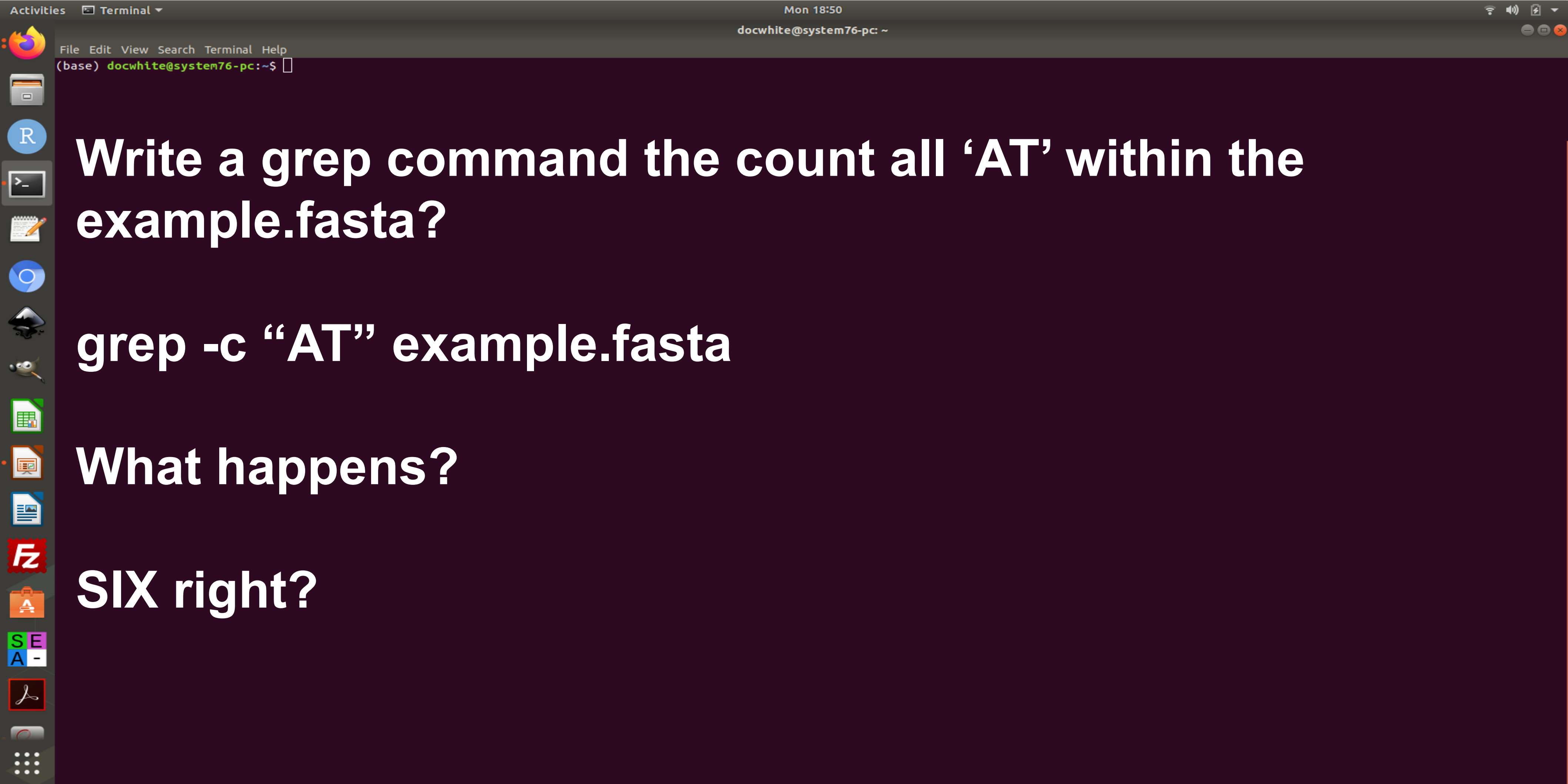
```
(base) docwhite@system76-pc:~$
```

Write a grep command the count all ‘AT’ within the example.fasta?

grep -c “AT” example.fasta

What happens?

grep examples



```
(base) docwhite@system76-pc:~$ 
```

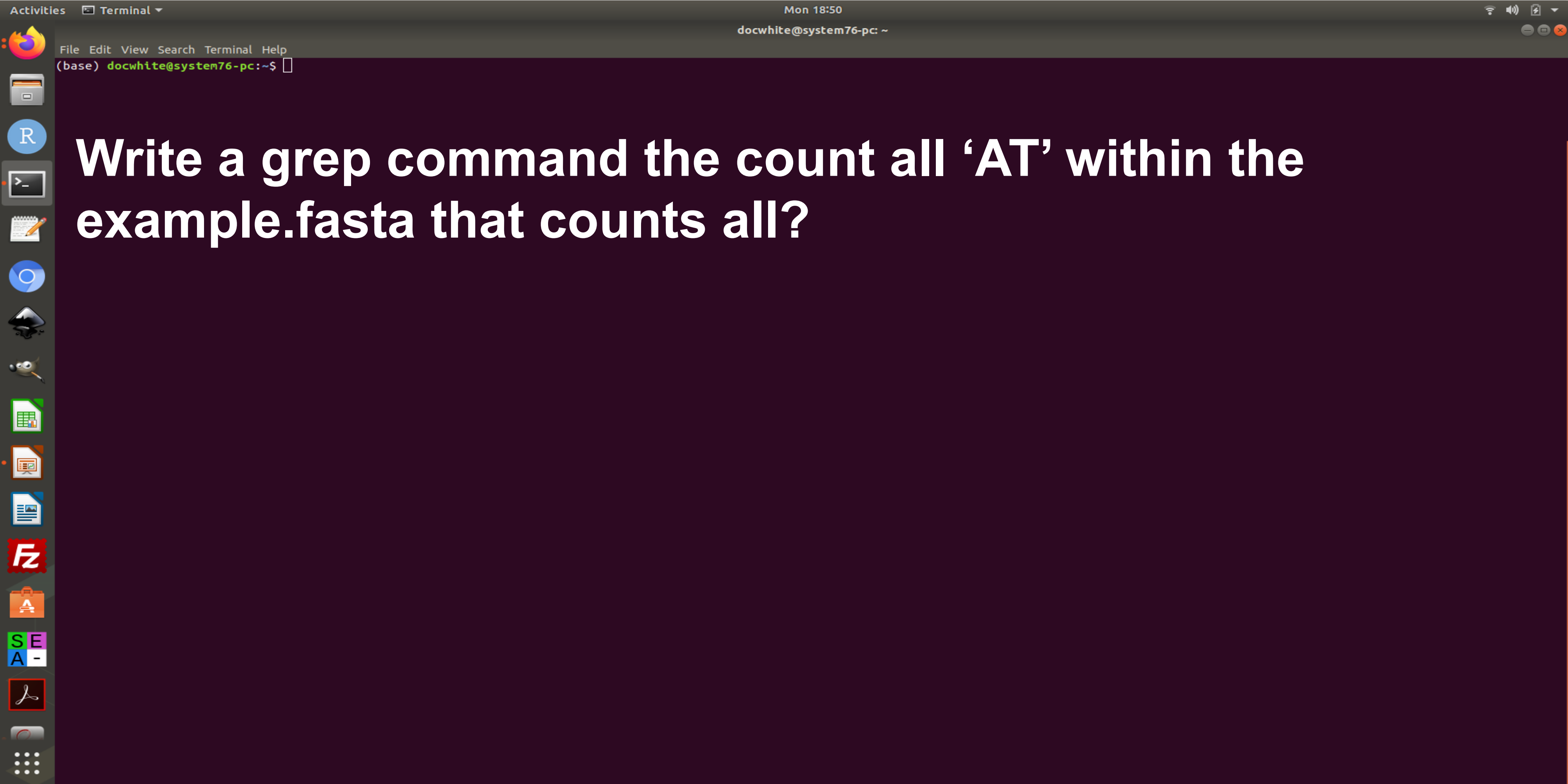
Write a grep command the count all ‘AT’ within the example.fasta?

grep -c “AT” example.fasta

What happens?

SIX right?

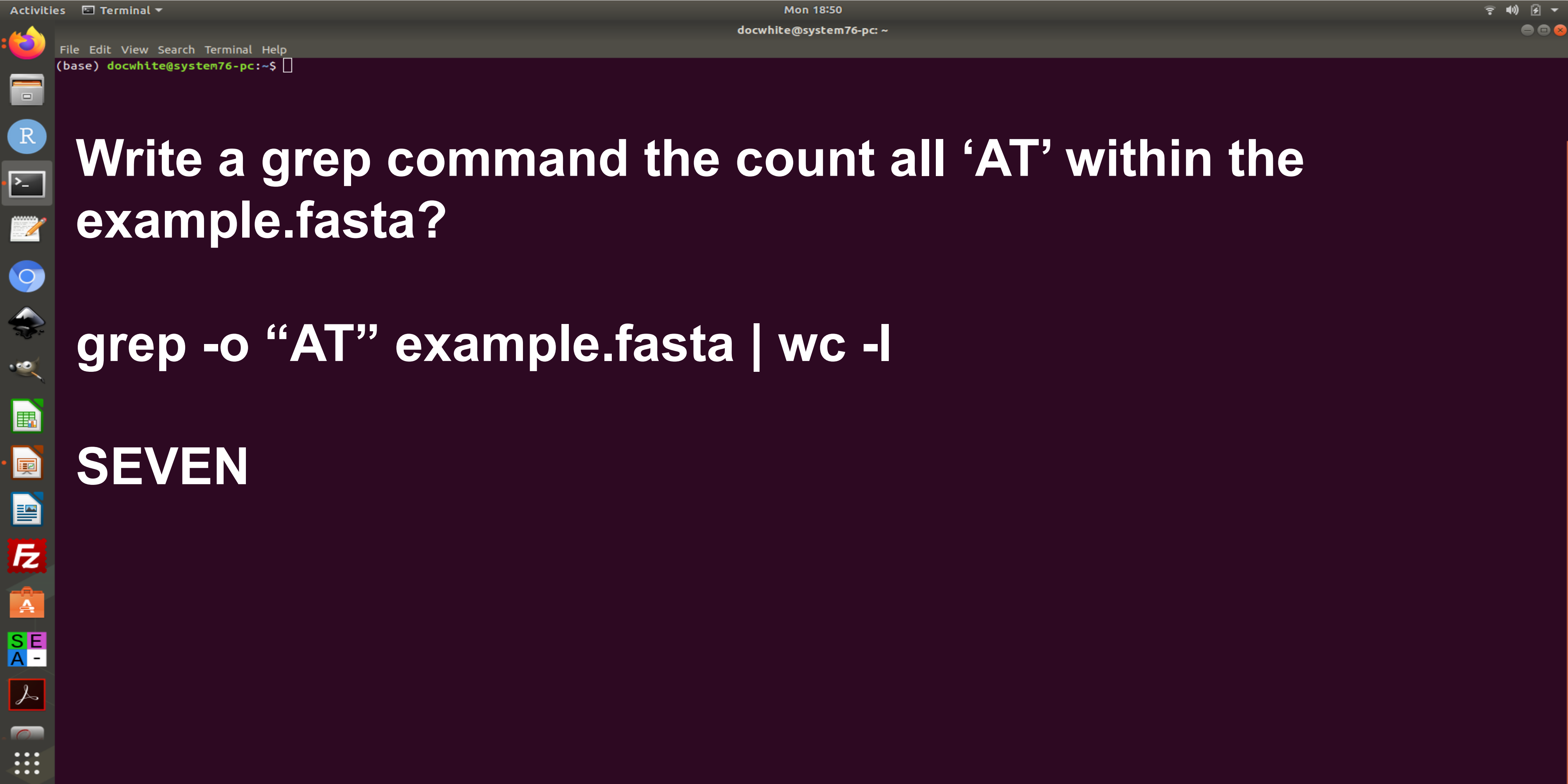
grep examples



```
(base) docwhite@system76-pc:~$ 
```

Write a grep command the count all ‘AT’ within the example.fasta that counts all?

grep examples



```
(base) docwhite@system76-pc:~$
```

Write a grep command the count all ‘AT’ within the example.fasta?

grep -o “AT” example.fasta | wc -l

SEVEN

Bonus 3

- Count both the number of AT and GC in one grep command and in another command print the line number which they appear?

Quiz 3

- On canvas now