Lecture 09
regular expressions (intro)

Course: Practical Bioinformatics (BIOL 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 09 outline

Last time: sequence alignment

This time: regular expressions

- uses of regex
- basic syntax
- examples

Regular expressions

A ***regular expressions*** (or *regex*) defines a pattern to search (and capture) text

Regular expressions are useful to

- ***collect*** information
- ***navigate*** and ***parse*** text
- ***search*** and ***replace*** complex text patterns
- ***condense*** your code

Regex use

Regex statements have a pre-defined but flexible grammatical structure, and make use of special characters to define text patterns

A regex statement is applied to a body of text, then ***matches*** text that fits the regex pattern.

Example:

regex	text
[br]at	bat (match)
	hat (skip)
	rat (match)

Using regex with *grep*

global ***r***egular ***e***xpression ***p***rint

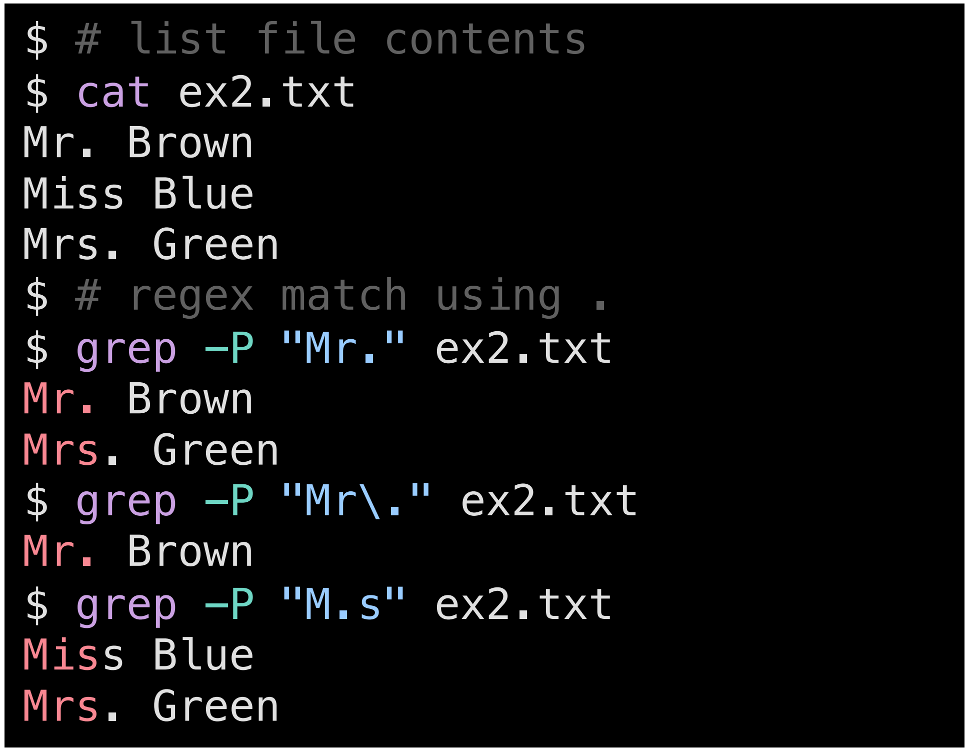
```
$ # list file contents
$ cat ex1.txt
bat
brat
chat
yacht
$ # regex for a simple pattern match
$ # -P : use Perl-Compatible Regex
$ grep -P "chat" ex1.txt
chat
```

Regex is supported for many programming languages, but how they implement regex may differ.

We'll use ***Perl-Compatible Regular Expressions*** (PCRE)

Any character, .

. will match exactly *one* character with any value



```
$ # list file contents
$ cat ex2.txt
Mr. Brown
Miss Blue
Mrs. Green
$ # regex match using .
$ grep -P "Mr." ex2.txt
Mr. Brown
Mrs. Green
$ grep -P "Mr\." ex2.txt
Mr. Brown
$ grep -P "M.s" ex2.txt
Miss Blue
Mrs. Green
```

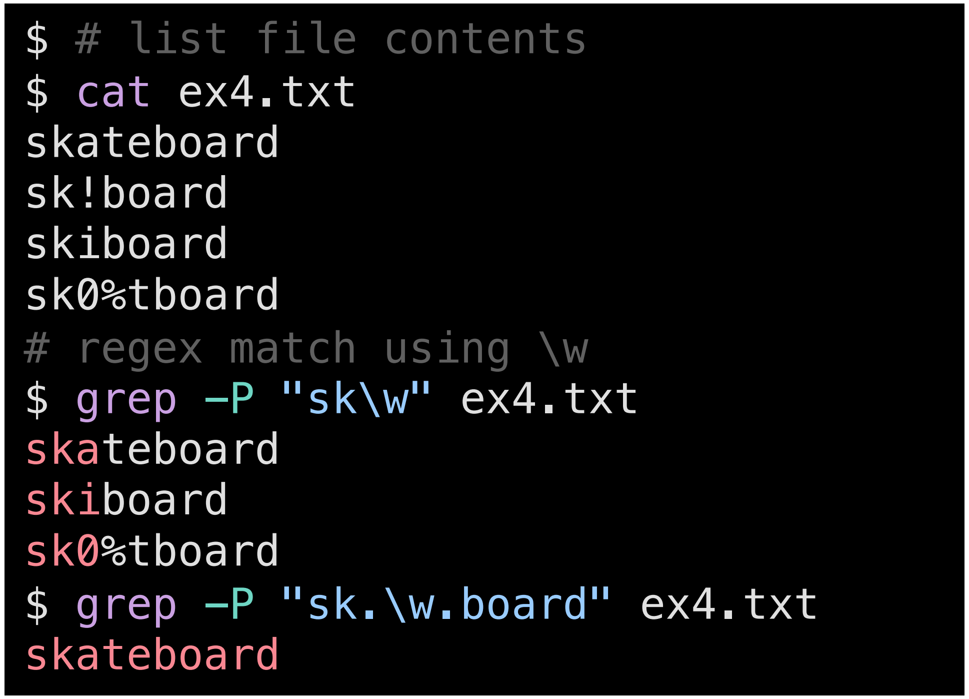
Any digit, \d

\d will match exactly *one* character with any numerical value in 0..9

```
$ # list file contents
$ cat ex3.txt
King Richard III
King Henry the 8th
King Susie the 72nd
# regex match using \d
$ grep -P "\d" ex3.txt
King Henry the 8th
King Susie the 72nd
$ grep -P "\d\d" ex3.txt
King Susie the 72nd
```

Any alphanumeric, \w

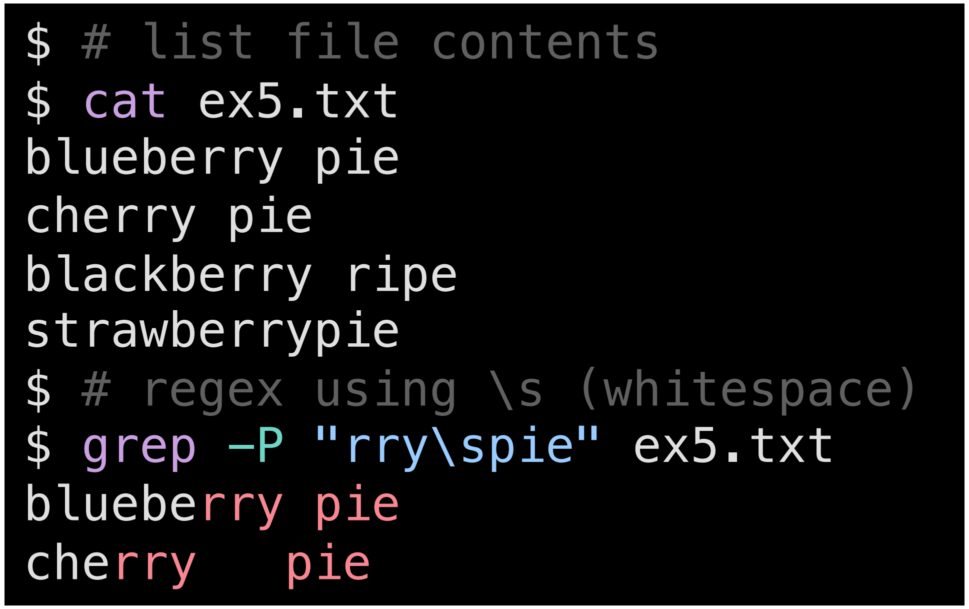
\w will match exactly *one* character with any numerical value from A..Z, a..z, or 0..9



```
$ # list file contents
$ cat ex4.txt
skateboard
sk!board
skiboard
sk0%tboard
# regex match using \w
$ grep -P "sk\w" ex4.txt
skateboard
skiboard
sk0%tboard
$ grep -P "sk.\w.board" ex4.txt
skateboard
```

Any whitespace, *s*

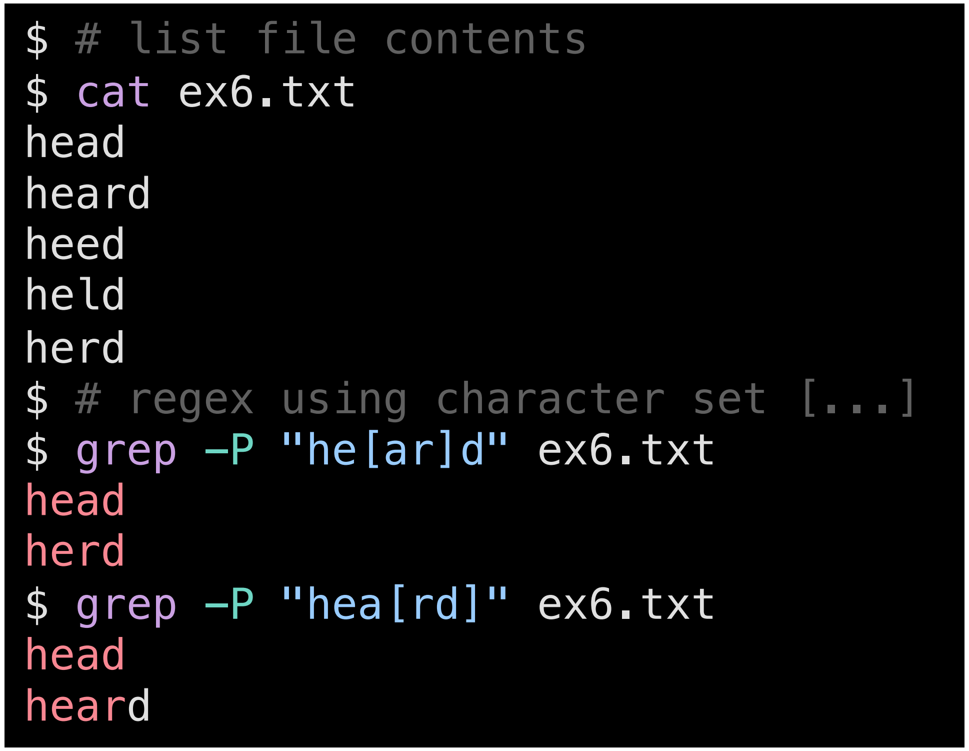
s will match exactly *one* character with any whitespace value: [space], [tab], [newline]



```
$ # list file contents
$ cat ex5.txt
blueberry pie
cherry pie
blackberry ripe
strawberrypie
$ # regex using \s (whitespace)
$ grep -P "rry\spie" ex5.txt
blueberry pie
cherry   pie
```

Character-set, *[ab]*

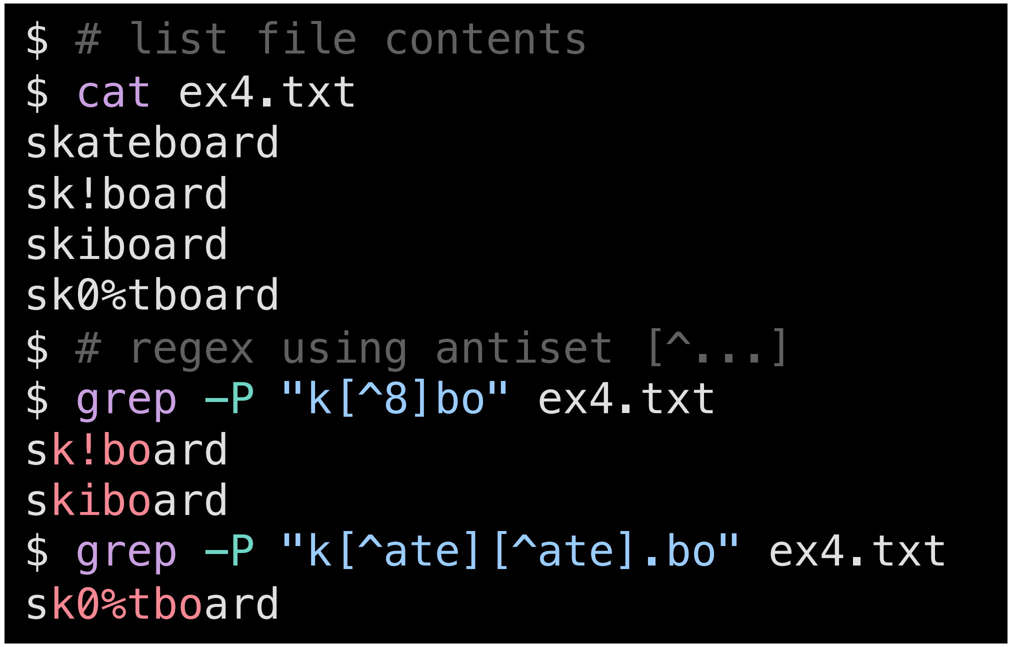
[ab] will match with *one* character that is a member of the set *a* or *b*



```
$ # list file contents
$ cat ex6.txt
head
heard
heed
held
herd
$ # regex using character set [...]
$ grep -P "he[ar]d" ex6.txt
head
herd
$ grep -P "hea[rd]" ex6.txt
head
heard
```

Anti-set, *[^ab]*

[^ab] will match with *one* character that is ***not*** a member of the set *a* or *b*



```
$ # list file contents
$ cat ex4.txt
skateboard
sk!board
skiboard
sk0%tboard
$ # regex using antiset [^...]
$ grep -P "k[^8]bo" ex4.txt
sk!board
skiboard
$ grep -P "k[^ate][^ate].bo" ex4.txt
sk0%tboard
```

Character ranges, *[m-z]*

[m-z] will match with *one* character that is in the character range *m..z*

```
$ # list file contents
$ cat ex7.txt
arm
chin
hand
foot
knee
$ # regex match using char range, [a-g]
$ grep -P "[a-g]..." ex7.txt
chin
foot
$ grep -P "..[a-m]." ex7.txt
chin
knee
```

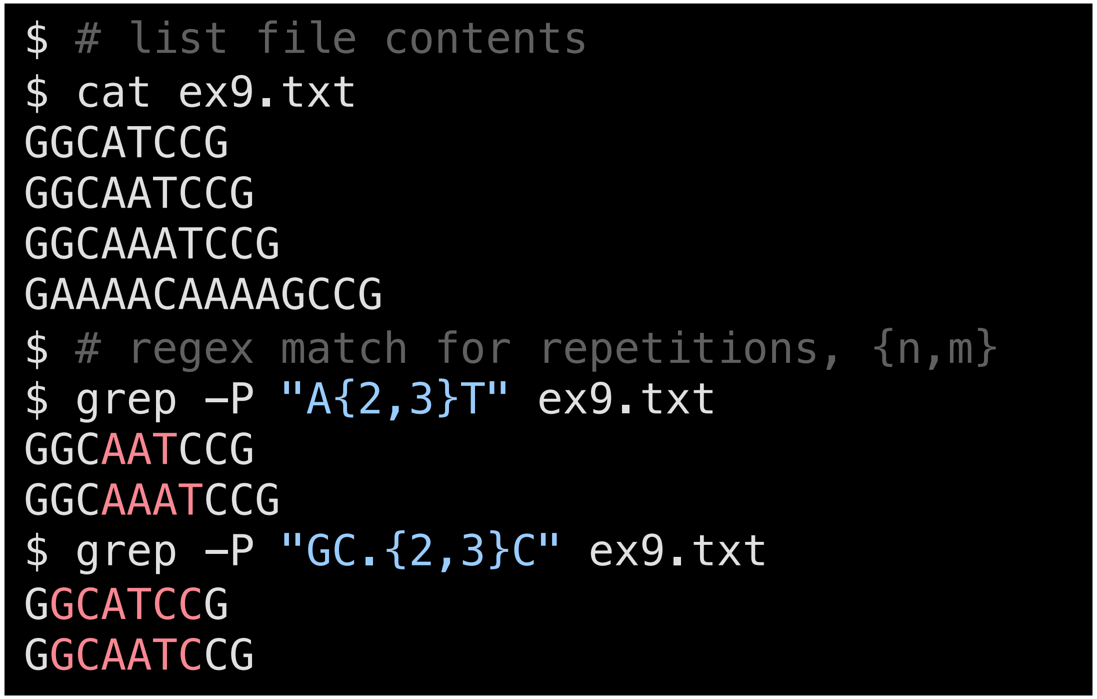
Repetitions, {*m*}

{*m*} will match the preceding pattern
if it appears *exactly m times* in the text

```
$ # list file contents
$ cat ex8.txt
GATACAT
GATAACAT
GATAAACAT
GATAAAACAT
$ # regex match for repetitions, {n}
$ grep -P "A{3}" ex8.txt
GATAAACAT
GATAAAACAT
$ grep -P "TA{2}C" ex8.txt
GATAACAT
```

Repetition range, {*m*,*n*}

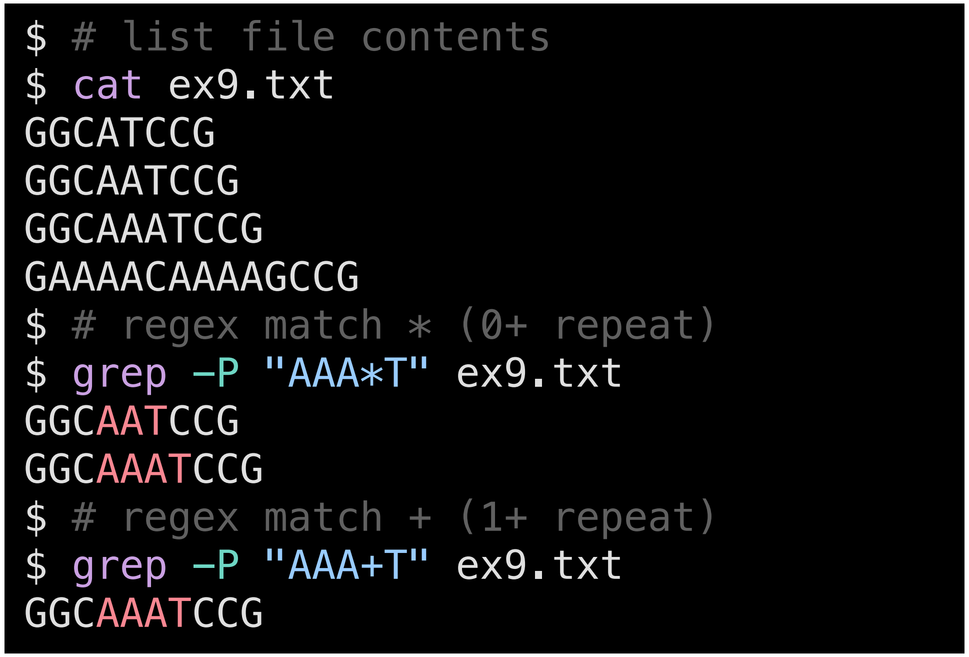
{*m*,*n*} will match the preceding pattern
if it appears *between m and n times* in the text



```
$ # list file contents
$ cat ex9.txt
GGCATCCG
GGCAATCCG
GGCAAATCCG
GAAAACAAAAGCCG
$ # regex match for repetitions, {n,m}
$ grep -P "A{2,3}T" ex9.txt
GGCAATCCG
GGCAAATCCG
$ grep -P "GC.{2,3}C" ex9.txt
GGCATCCG
GGCAATCCG
```

Kleene repetitions, * *and* +

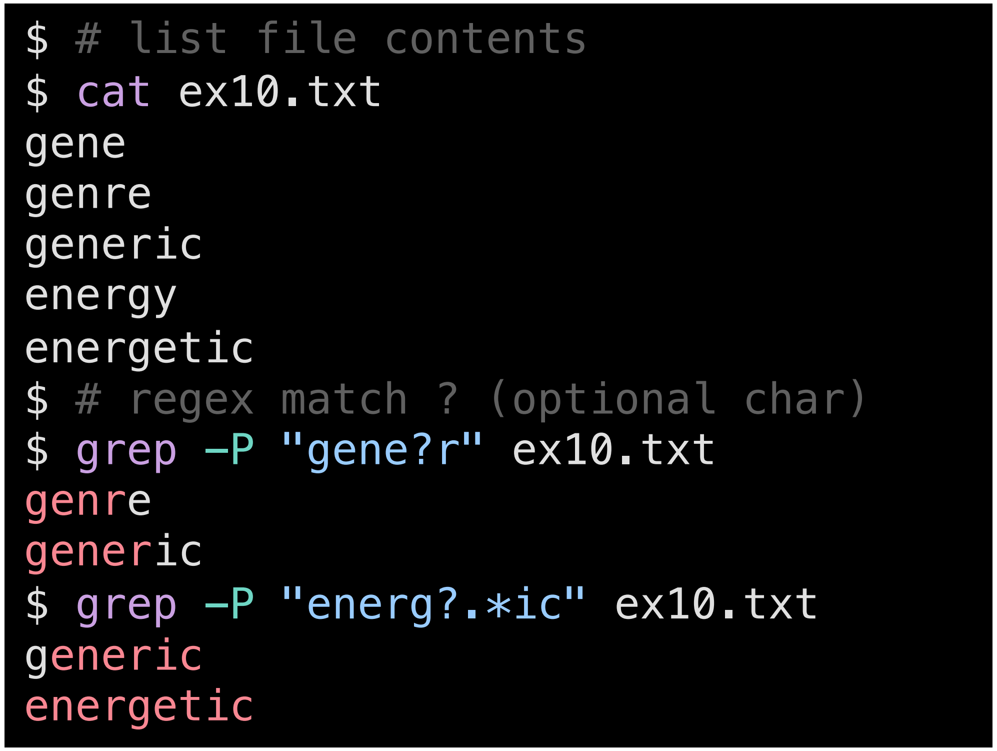
* will match the preceding pattern 0+ times
+ will match the preceding pattern 1+ times



```
$ # list file contents
$ cat ex9.txt
GGCATCCG
GGCAATCCG
GGCAAATCCG
GAAAACAAAAGCCG
$ # regex match * (0+ repeat)
$ grep -P "AAA*T" ex9.txt
GGCAATCCG
GGCAAATCCG
$ # regex match + (1+ repeat)
$ grep -P "AAA+T" ex9.txt
GGCAAATCCG
```

Optional character, *?*

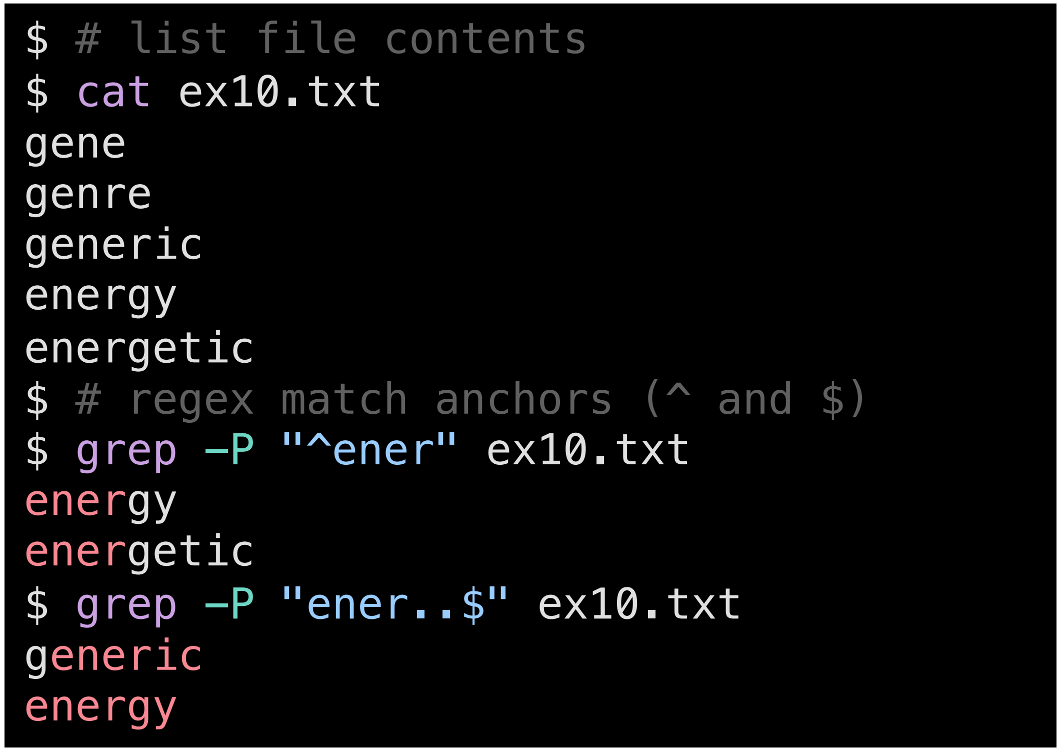
? will match the preceding pattern
either exactly 0 or 1 time



```
$ # list file contents
$ cat ex10.txt
gene
genre
generic
energy
energetic
$ # regex match ? (optional char)
$ grep -P "gene?r" ex10.txt
genre
generic
$ grep -P "energ?.*ic" ex10.txt
generic
energetic
```

Anchors, ^ and $

^ indicates the *start* of the matched string
$ indicates the *end* of the matched string



```
$ # list file contents
$ cat ex10.txt
gene
genre
generic
energy
energetic
$ # regex match anchors (^ and $)
$ grep -P "^ener" ex10.txt
energy
energetic
$ grep -P "ener..$" ex10.txt
generic
energy
```

Managing *ssh* sessions

Each *ssh* will generally terminate all processes it initiated when it ends

Terminal multiplexers create shell sessions that persist after log-off and that can be resumed from other log-ons

Popular tools include *tmux*, *screen*

Using *tmux*

list *tmux* sessions (none found)

create and enter a new session, do work, and then detach from session

```
$ # original SSH session
$ tmux ls
no server running on /tmp/tmux-1001/default
$ tmux new -s my_job

... create and enter new tmux session ...
... work within `my_job` session ...
... detach from session with `ctrl-B D` ...
... return to original ssh session ...

[detached (from session my_job)]
$ exit
```

list *tmux* sessions (*my_job* found)

attach to *my_job* session, complete work then close *my_job*

list *tmux* sessions (none found)

```
$ # later SSH session
$ tmux ls
my_job: 1 windows (created Thu Sep 22 10:29:41 2022)

$ tmux -a -t my_job
... re-attach to tmux session `my_job` ...
... continue working within `my_job` ...
... call `exit` in `my_job` to end session ...

[exited]
$ tmux ls
no server running on /tmp/tmux-1001/default
```

Overview for Lab 09