# Introduction to Regular Expressions

Demitri Muna

5 August 2016

# What Are Regular Expressions?

- Regular expressions are used for textual pattern matching.
- It's like half a computer language.
- Built into most modern programming languages and all good text editors
- Functionality provided on the command line though grep.
- Can be linked as a library into any code, e.g. C.
- Best learned through examples.

## grep

Open a terminal window in the same directory as the sample file student\_data.txt.

#### Usage:

% grep <pattern> <files to search>

#### Simple searches:

# Square Brackets

Square brackets match (i.e. find) a single character.

#### [bc]at

will match 'bat' and 'cat' but not 'mat' or 'hat'. You can place a range of letters or numbers inside the brackets.

On the command line:

$$% ls -d .[a-zA-Z]*$$

\* = any length of characters: list any hidden (starting with a '.') file or directory beginning with a letter

## Shortcuts

These are shortcuts that define frequently used patterns.

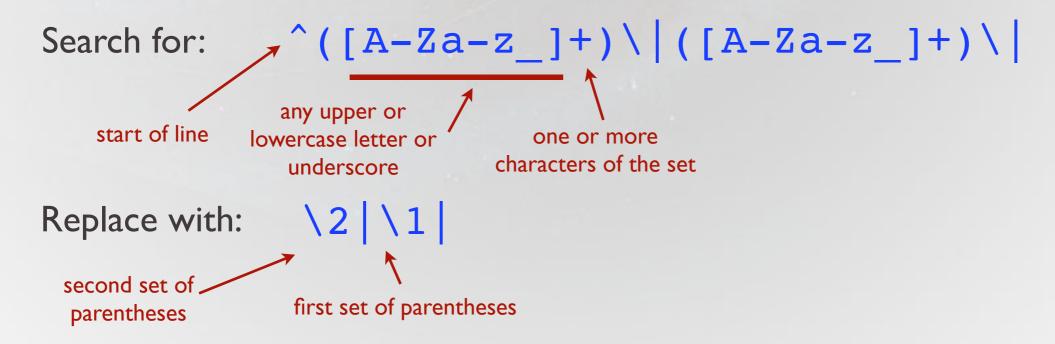
Shortcut	Matches
\s	any whitespace, e.g. space, tab, end of line (\n, \r), etc.
\S	any character except whitespace
/d	any digit : [0-9]
\D	any character excluding numbers
\w	effectively, a word : [0-9a-zA-Z]
\W	not a word : [^0-9a-zA-Z] (i.e. inverse of \w)

# Grouping

You can identify groups of characters by using parentheses. This is useful for extracting text that matches a pattern or search and replace operations.

Example:
Swap the columns
'first\_name' and 'last\_name'.

first\_name|last\_name|city|supervisors|club|status Cara|Rogers|New Britain|Tennant/Room 101|Sophomore|Che Ori|Mejia|Lakeland||Senior|Debate Leandra|Stevens|Rockford||Freshman| Danielle|Moody|Oro Valley|Baker/Room 315, Eccleston/Ro Josiah|Barber|Rancho Cordova||Sophomore| Wing|Gordon|Reedsport|Baker/Room 315|Freshman|Rugby, C



Note: what represents the groups differs between Python, Perl, text editors, etc.

# Grouping

Grouping can also let you specify more complex queries:

([ea][^r])

Matches two characters beginning with either 'e' or 'a', but not when followed by 'r'.

Imagine data files that look like this, and you want to extract the information from the filename.

```
data-7542-55726-01f.par
data-5468-55777-02f.par
data-9875-55728-01x.par
```

```
55726
   data
([a-z]+)-([0-9]+)-([0-9]+)-([0-9][0-9])[a-z].par
```

between I and 7 lowercase letters

exactly 4 digits

exactly 5 digits two or three digits single letter

$$([a-z]{1,7})-([0-9]{4})-([0-9]{5})-([0-9]{2,3})[a-z] \cdot par$$

'actually matches any character, so it should be escaped with a backslash to match a period

# Special Characters

Character	Meaning
\n, \t, \r	new line, tab, linefeed
•	any single character except a line break (\n, \r)
?	last item optional, e.g. [a-z][0-9]? matches 'a9' and 'x'
+	one or more of the last item, e.g. [a-z]+ one or more lowercase letters
1	escape character, e.g. \+ is an actual plus
٨	start of line
\$	end of line

Example: ^.+ matches any full line.

### Command Line Tricks

Useful for deleting certain ranges of files:

$$% rm */.[a-zA-Z0-9]*$$

Deletes files from January, June, and July.

Deletes hidden files in all subdirectories.

# RegExp in Python

The match method searches from the beginning of the string.

```
import re
                Import the regular expression module.
d = "The date is Jan-04-2011 today."
m = re.match(".+([A-Za-z]{3}))-([0-9]+)-([0-9]+).+", d)
assert m is not None, "The pattern was not matched."
month = m.group(1)
                         Three groups of parentheses for three
day = m.group(2)
                         matches, starting from index 1.
year = m.group(3)
                         index 0 returns the original string
print m.group(0)
print m.groups() <</pre>
                         returns an array of all the grouped items
m = re.match("Jan", d) m is None (doesn't match start of line)
m = re.match("The", d)
# typical usage
if (re.match(...)):
    # do stuff
```

If the string does not match the regular expression, match returns None.

The search method works in exactly the same way, but searches anywhere in the string (not strictly from the start).