REGULAR EXPRESSIONS AND AWK

WEEK 2 – 09/04/2019

GENERAL INFORMATION

- Emailing decorum
  - Subject should look like "CSCI 3308 – [Course Material/Information] – [Problem - *Optional*]"
    - [Course Material/Information] – Lecture, Recitation, Homework, Exam, General Information, Accommodations.
    - [Problem] – Summary of the issue being addressed in the email

- Reminder to fill out the team survey
  - Deadline: September 4, 2019 11:59 p.m.
  - If you choose to NOT complete this survey before the deadline, you will be randomly assigned to a group, and that assignment may impair your ability to work with your team due to schedule/personality/skills conflicts.
  - No changes to teams once they're formed

REGEX

- RegEx: A regular expression is a special text string for describing a search pattern.
- In Windows, a wildcard notation such as *.txt finds all text files in a directory.
- The regex equivalent is ^.*\.txt$.
- Why RegEx?
  - Programmatically matching data in records to a pattern
  - Validating user data input

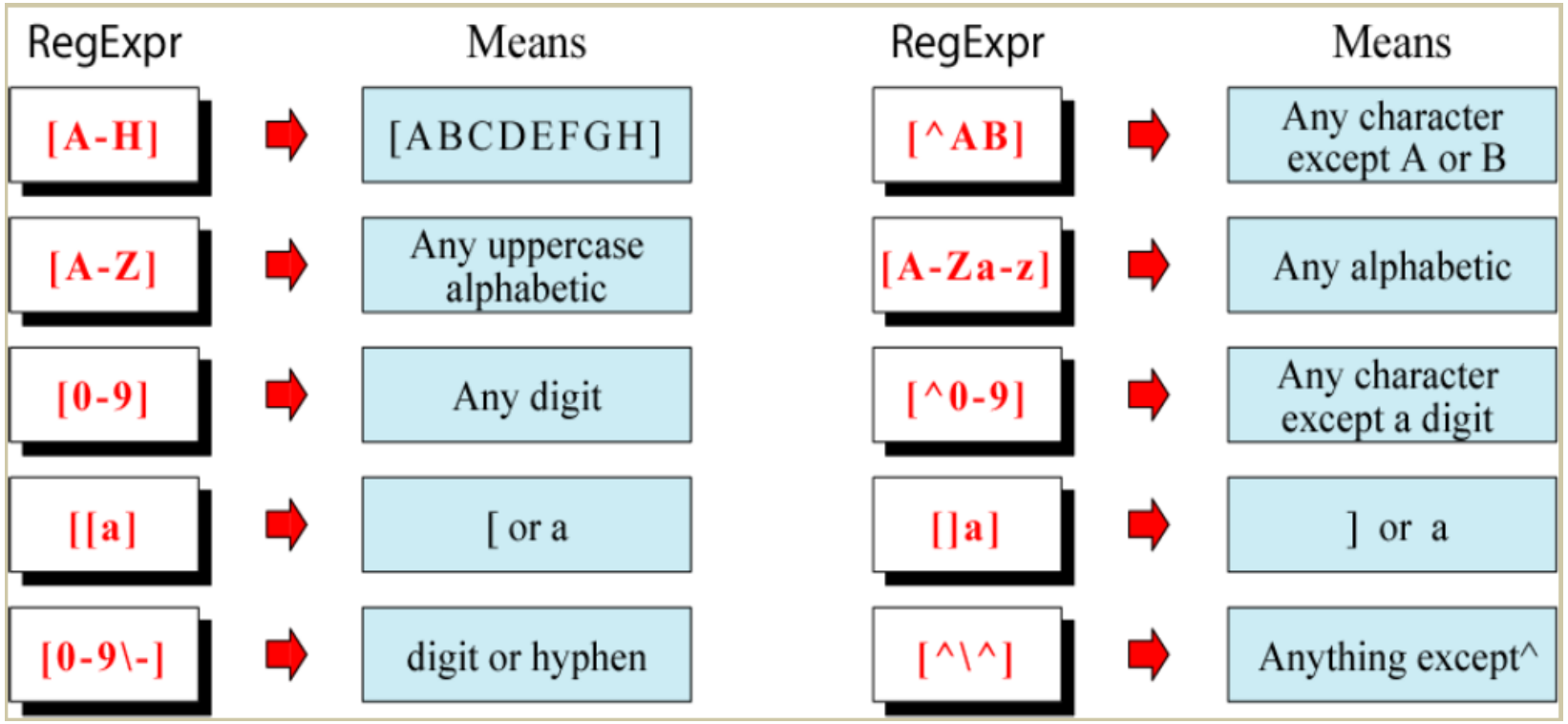
METACHARACTERS

- Metacharacters are the building blocks of regular expressions.
- Characters in RegEx are understood to be either a metacharacter with a special meaning or a regular character with a literal meaning.

METACHARACTERS

Metacharacter	Usage
.	Any one character
[]	Any enclosed character
*	Zero or more of the preceding character
?	Zero or one of the preceding character
+	One or more of the preceding character
^	Anchor - The beginning of a string
$	Anchor – The end of the string
\	Escape character
|	Boolean OR
{m,n}	The preceding character appears **m** to **n** times
\b	Anchor – The beginning of a word
[[:blank:]]	Space or Tab

REGEX


RegExpr
Means
RegExpr
Means
[A-H]
[ABCDEFGH]
[^AB]
Any character except A or B
[A-Z]
Any uppercase alphabetic
[A-Za-z]
Any alphabetic
[0-9]
Any digit
[^0-9]
Any character except a digit
[[a]
[or a
[]a]
] or a
[0-9\-]
digit or hyphen
[^\^]
Anything except^

EXAMPLES

- \"\(8
- 5{2}
- [Jj]a
- [Jj]a.{1,10}
- \b[Mm]|\b[Ff]

EXAMPLES

^The	matches any string that starts with "The".
of despair$	matches a string that ends in with "of despair".
^abc$	a string that starts and ends with "abc" - effectively an exact match comparison.
notice	a string that has the text "notice" in it.
ab*	matches a string that has an a followed by zero or more b's ("ac", "abc", "abbc", etc.)
ab+	same, but there's at least one b ("abc", "abbc", etc., but not "ac")
ab?	there might be a single b or not ("ac", "abc" but not "abbc").
a?b+$	a possible 'a' followed by one or more 'b's at the end of the string: Matches any string ending with "ab", "abb", "abbb" etc. or "b", "bb" etc. but not "aab", "aabb" etc.

EXAMPLES

"[ab]"	matches a string that has either an a or a b (that's the same as "a|b")
"[a-d]"	a string that has lowercase letters 'a' through 'd' (that's equal to "a|b|c|d" and even "[abcd]")
"^[a-zA-Z]"	a string that starts with a letter
"[0-9]%"	a string that has a single digit before a percent sign
",[a-zA-Z0- 9]$"	a string that ends in a comma followed by an alphanumeric character
"ab{2}"	matches a string that has an a followed by exactly two b's ("abb")
"ab{2,}"	a followed by at least two b's ("abb", "abbbb", etc.)
"ab{3,5}"	a followed by from three to five b's ("abbb", "abbbb", or "abbbbb")
"a(bc)*"	matches a string that has an a followed by zero or more copies of the sequence "bc"
"a(bc){1,5}"	a followed by one through five copies of "bc"
"hi|hello"	matches a string that has either "hi" or "hello" in it
"(b|cd)ef"	a string that has either "bef" or "cdef"
"(a|b)*c"	a string that has a sequence of alternating a's and b's ending in a c
"a.[0-9]"	matches a string that has an a followed by one character and a digit
"^.{3}$"	a string with exactly 3 characters

QUESTION ASKED IN CLASS

- To disregard a complete word from a string I have created a regex and you can find it here:

  https://regex101.com/r/DbRE7y/1

  Do read the explanation on the right hand side to understand how it works. If you have questions, we'll address them in the next lecture.

REGEX

Further Reading:

- https://regex101.com/
- https://regexone.com
- http://www.zytrax.com/tech/web/regex.htm
- https://docs.python.org/2/howto/regex.html

GREP WITH REGEX

- Literal matches – eg. grep -vn "the" textfile.txt
- Anchor matches – eg. grep "^GNU" textfile.txt
- Matching Any Character – eg. grep "..cept" textfile.txt
- Bracket Expressions – eg. grep "^[A-Z]" textfile.txt
- Repeat Pattern Zero or More Times – eg. grep "([A-Za-z]*)" textfile.txt

AWK

- A programming language designed for text processing
- Used for processing regular expressions in a script
- Used when the text is in file / delimited field format
- Typically used as a data extraction and reporting tool
- A powerful standard feature of most Unix-like operating systems.

AWK

awk operations:

- Scans a file line by line
- Splits each input line into fields
- Compares input line/fields to pattern
- Performs action(s) on matched lines

Useful for:

- Transforming data files
- Producing formatted reports

Programming constructs:

- Format output lines for reports
- Arithmetic and string operations
- Conditionals and loops

BASIC AWK SCRIPT

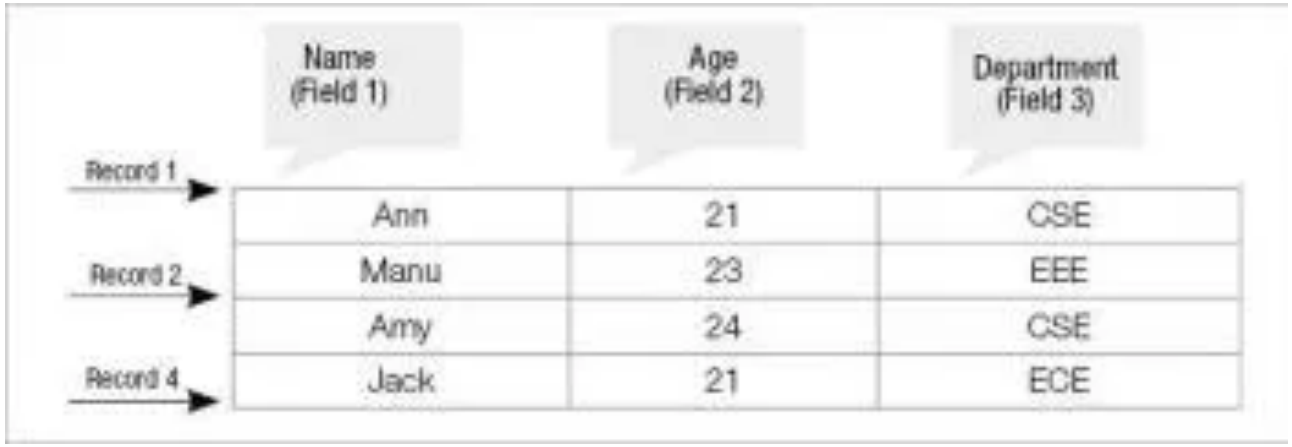
- Consists of patterns & actions:
  - Pattern {action}
    - If pattern is missing, action is applied to all lines
    - If action is missing, the matched line is printed
    - Must have either pattern or action
- Example:
  - Awk '/for/' testfile
    - Prints all lines containing string "for" in testfile

BASIC TERMINOLOGY

- A field is a unit of data
- Each field is separated from the other fields by the field separator
  - Default field separator is whitespace
  - A record is the collection of fields in a line
  - A data file is made up of records

BUFFERS

- Awk supports two types of buffers: record and field
- Fields buffer:
  - One for each fields in the current record
    - Names: $1, $2, ..
  - Record buffer:
    - $0 holds the entire record



shorturl.at/bjzEF

SYSTEM VARIABLES

- FS – Field Separator (default = whitespace)
- RS – Record Separator (default = \n)
- NF – Number of Fields in the current record
- NR – Number of the current record
- OFS – Output file separator (default = space)
- ORS – Output record separator (default = \n)
- FILENAME - Current FileName

PATTERN TYPES

- Match
  - Entire input record

  Regular expression enclosed by "/"s
  - Explicit pattern-matching expressions
    - ~(match), !~ (not match)

- Expression operators
  - Arithmetic
  - Relational
  - Logical

EXAMPLES

- awk '{print}' employee.txt
- awk '/manager/ {print}' employee.txt
- awk '{print $1,$4}' employee.txt
- awk '{print NR,$0}' employee.txt
- awk '{print $1,$NF}' employee.txt
- awk 'NR==3, NR==6 {print NR,$0}' employee.txt
- awk '$2 !~ /manager/ {print}' employee.txt
- awk '$2 ~ /manager/ {print}' employee.txt
- $ awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }'

AWK

Further Reading:

- www.hcs.harvard.edu/~dholland/computers/awk.html
- http://web.mit.edu/gnu/doc/html/gawk_8.html
- https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux

SED

- A programming language designed for text processing
- Used for processing regular expressions in a script
- Used when the text is in a stream (no delimited field structure)
- Typically used as a stream editor (thus the name)
- A powerful standard feature of most Unix-like operating systems

SED

sed operations:

- Loops through a file line by line
- Looks for text patterns
- Executes commands on a match
  - Substitute, delete, insert a new line, etc.

Useful for:

- Transforming data files
- Finding text strings and changing them

SED

Further Reading:

- http://www.wikiwand.com/en/Sed
- http://www.grymoire.com/Unix/Sed.html
- https://www.tutorialspoint.com/sed/sed_overview.htm