

Lab 07: Regular Expressions

You should turn in:

1. A file named Lab07.docx.

Instructions:

1. Below are five sets of strings, you will need to create a regular expression for each set which **does** match the first two strings, but **does not** match the third.
 - a. Do not use the period operator (wildcard that matches any one element) anywhere in any of your regular expressions.
 - b. Begin all of your expressions with a ^, meaning that it must be located at the beginning of a line
 - c. End all of your expressions with a \$, meaning that no other characters can come after the matched characters on the same line
2. Test the regular expressions in MATLAB by setting each string to the variable str, and your regular expression to the variable expr (use single quotes for both to mark them as strings), and then running the following line of code:
result = regexp(str, expr, 'match')
If the result variable contains the string, then the regular expression matches the input string. If result is an empty cell array, then the expression does not match the input string.
3. This assignment **will not have partial credit**, you must test your regular expressions thoroughly

	Expression 1	Expression 2	Expression 3	Expression 4	Expression 5
Accepted String 1	35.01	100,000,000	2020-10-31	qlanik2@huskers.unl.edu	1400 Maple Rd. Lincoln, NE
Accepted String 2	57920	12,345,678	1970-1-01	support@gmail.com	221B Baker St. London, UK
Unacceptable String	038.201	12,23,45,	252525-15- 30	asdf-1234#notmail.com	123 Fake Street Nowhere, USA

Useful Tips:

Below is a summary of regular expression rules, you may also want to review the video of Lecture 22 which goes into the topic in more detail.

Wildcards:

- . matches any single character (do not use in this lab)
- * matches 0 or more of the expression which came before it
- + matches 1 or more of the expression which came before it
- ? matches 0 or 1 of the expression which came before it
- () around one or more expressions causes them to be considered a single expression for the purposes of a *, +, or ? operator

Character sets:

- [] around a set of characters denotes that anything present in the set can be matched
 - between two characters denotes that any elements in the ASCII table between the two characters should be included in the set. When used as the first or last character in square brackets, a literal hyphen is included in the set
- A wildcard after a set of square brackets acts as on all elements in the brackets (e.g. [0-9]+ matches 1 or more numeric characters)
- A ^ used as the first character inside a pair of square brackets says to match any element not included in the set (e.g. [^A-Z] matches anything except a capital letter)

Anchors:

- ^ when not used in square brackets says that a string must be at the beginning of a line
- \$ denotes the end of a line, meaning that no characters can come after it on a line

String literals:

Characters which have a specific meaning in regular expressions, such as parentheses or the plus sign need to be 'escaped' to have them be matched literally. If you want to include any of the following characters, they need a backslash in front of them: `.*+[]{}()|^\$`

In contrast, some characters require a backslash in front of them to convert them into a special character (none are needed in this lab):

- `\n` denotes a newline character
- `\r` denotes a carriage return, an outdated character that goes with `\n`
- `\t` denotes a tab
- `\s` denotes any whitespace character