

Regular Expressions in Practice



CSPC 229: Foundations of

56

Regular Expressions in Practice

- used for *pattern matching*
- the alphabet is typically the characters on the keyboard
- use an *escape character* to distinguish *meta-characters* from alphabet characters
 - *, |, () are part of the regular expression syntax, but we may also want to use them as alphabet characters
 - escape character is typically \

<code>\ </code>	the characters	<code>\n</code>	newline
<code>*</code>	, *, (,)	<code>\t</code>	tab
<code>\(\)</code>			

CSPC 229: Foundations of Computation • Spring 2026

57

Regular Expressions in Practice

$r_1 r_2$	or (union)	
r^*	Kleene star (0 or more)	there are variations in syntax from one application to the next, especially beyond the basic core
(r)	grouping	
r^+	1 or more times	
$r?$	0 or 1 times	
$r\{n\}$	n times	
$[a_1 a_2 a_3]$	match any one of a_1 , a_2 , or a_3	
$[a_1 - a_n]$	match any one character between a_1 and a_n	
$[\^a_1 a_2 a_3]$	match any one character <i>not</i> a_1 , a_2 , or a_3	
$[\^a_1 - a_n]$	match any one character <i>not</i> between a_1 and a_n	
$.$	match any single character	
$^$	match the beginning of the line	
$\$$	match the end of the line	
$\backslash b$	match word boundary	
$\backslash s$	match any whitespace	

58

1. The backslash is itself a meta-character. Suppose that you want to match a string that contains a backslash character. How do you suppose you would represent the backslash in the regular expression?
2. Using the notation introduced in this section, write a regular expression that could be used to match each of the following:
 - a) Any sequence of letters (upper- or lowercase) that includes the letter Z (in uppercase).
 - b) Any eleven-digit telephone number written in the form (xxx)xxx-xxxx.
 - c) Any eleven-digit telephone number *either* in the form (xxx)xxx-xxxx or xxx-xxx-xxxx.
 - d) A non-negative real number with an optional decimal part. The expression should match numbers such as 17, 183.9999, 182., 0, 0.001, and 21333.2.
 - e) A complete line of text that contains only letters.
 - f) A C++ style one-line comment consisting of // and all the following characters up to the end-of-line.

CSPC 229: Foundations of Computation • Spring 2026

59

Substitution

- \$0 entire substring matching the pattern
 - \$1 substring matched by the part of the pattern beginning with the first (and ending with the matching)
 - ...
 - \$9
- there are variations in syntax from one application to the next, especially beyond the basic core

3. Give a search pattern and a replace pattern that could be used to perform the following conversions:
- a) Convert a string that is enclosed in a pair of double quotes to the same string with the double quotes replaced by single quotes.
 - b) Convert seven-digit telephone numbers in the format xxx-xxx-xxxx to the format (xxx)xxx-xxxx.
 - c) Convert C++ one-line comments, consisting of characters between // and end-of-line, to C style comments enclosed between /* and */.
 - d) Convert any number of consecutive spaces and tabs to a single space.

Back References

- \0 entire substring matching the pattern
 - \1 substring matched by the part of the pattern beginning with the first (and ending with the matching)
 - ...
 - \9
- there are variations in syntax from one application to the next, especially beyond the basic core

4. In some implementations of “regular expressions,” the notations \1, \2, and so on can occur in a search pattern. For example, consider the search pattern $^([a-zA-Z]).*\1$$. Here, \1 represents a recurrence of the same substring that matched $[a-zA-Z]$, the part of the pattern between the first pair of parentheses. The entire pattern, therefore, will match a line of text that begins and ends with the same letter. Using this notation, write a pattern that matches all strings in the language $L = \{a^nba^n \mid n \geq 0\}$. (Later in this chapter, we will see that L is *not* a regular language, so allowing the use of \1 in a “regular expression” means that it’s not really a regular expression at all! This notation can add a real increase in expressive power to the patterns that contain it.)

Some Applications

```
import re
import time
start_time = time.time()
clean_urls = re.findall(r'href=[\"]?([^" >]+)', raw_url_data)
print('--- Executed in %s seconds ---' % (time.time() - start_time))
print('\n', join(clean_urls))
```

parsing –
extract links in an
HTML document

```
function isValidPhone(phone) {
  if (/^\(\d{3}[\d]{3})\d{3}-\d{4}$/.test(phone)) {
    console.log('phone number is valid');
  } else {
    console.log('phone number is invalid');
  }
}
```

input validation –
phone number

```
import re
for sentence in Text:
  new_line_removed = str(sentence).replace(r'\n', ' ')
  email_removed = re.sub(r'[A-Za-z0-9]*@[A-Za-z]*\.[A-Za-z0-9]*', ' ', new_line_removed)
  symbols_removed = re.sub(r'[A-Za-z0-9]+', ' ', email_removed)
  clean_data = re.sub(r"[^\w]\d+", ' ', symbols_removed)
  print(clean_data)
```

data cleansing –
preprocess text by replacing newlines with spaces,
removing email addresses, symbols, digits