Regular expressions – Cheat-sheet and examples

(V. Nicosia – 2021)

Definitions

Character and string. Characters are the smallest components of a string, i.e., single letters, digit, punctuation symbols, etc. A string is a sequence of characters. The string "`Queen!`" consists of the six characters '`Q`', '`u`', '`e`', '`e`', '`n`', '`!`'.

Regular expression. A regular expression (in short `regexp`) is a pattern that describes a set of strings. We say that a string "matches" a given regular expression if it adheres to the pattern defined by the regexp.

Meta-character. Some characters have a special meaning when used in regular expressions, i.e., they do not directly represent the corresponding character but are used to signify something else. Meta-characters include: '`.`' (dot), '`*`', '`+`', '\', '`(`', '`)`', '`[`', '`]`', '{', '}', '`?`', '`^`', '`$`'. If you want to indicate a meta-character as literal (for instance because you are looking for a string that contains one of them), you need to preceed it by a '\'. For instance, the pattern "`a\*b`" indicates a string formed by the three characters '`a`', '`*`', '`b`', while the pattern "`a*b`" is a regular expression that matches a sequence of zero or more '`a`' followed by a '`b`' (no '`*`' in there...).

Patterns

Groups and ranges				
	Meaning	`grep -E`	`grep -P`	Python `re`
`.` (dot)	any character	Y	Y	Y
`[abcd]`	one character among those specified within `[]`	Y	Y	Y
`[^ab]`	none of the characters specified within `[]`	Y	Y	Y
`[b-g]`	any of the characters in a certain range (here, the characters between '`b`' and '`g`', both included)	Y	Y	Y
`(bar)`	a group of characters (here the sequence "`bar`")	Y	Y	Y

Quantifiers				
	Meaning	`grep -E`	`grep -P`	Python `re`
*	zero or more of whatever preceeds it	Y	Y	Y
+	one or more or whatever preceeds it	Y	Y	Y
?	zero or one of whatever preceeds it	Y	Y	Y
{2}	exactly 2 of whatever preceeds it	Y	Y	Y
{2,6}	between 2 and 6 times of whatever preceeds it	Y	Y	Y
{4,}	four or more of whatever preceeds it	Y	Y	Y

Anchors				
	Meaning	`grep -E`	`grep -P`	Python `re`
^	the beginning of string or line	Y	Y	Y
$	the end of string or line	Y	Y	Y
\A	start of string	N	Y	Y
\Z	end of string	N	Y	Y
\b	word boundary	N	Y	Y
\B	not a word boundary	N	Y	Y
\<	start of word	N	Y	Y
\>	end of word	N	Y	Y

Character classes				
	Meaning	`grep -E`	`grep -P`	Python `re`
\c	a control character	N	Y	Y
\s	a white space (including tabs)	N	Y	Y
\S	not a white space	N	Y	Y
\d	a digit	N	Y	Y
\D	not a digit	N	Y	Y
\w	a word	N	Y	Y
\W	not a word	N	Y	Y
\x	hexadecimal digit	N	Y	Y
\O	octal digit	N	Y	Y
`[:digit:]`	a digit	Y	Y	N
`[:alpha:]`	a letter	Y	Y	N
`[:alnum:]`	an alphanumeric character (letter or digit)	Y	Y	N
`[:space:]`	a blank space	Y	Y	N
`[:cntrl:]`	a control character	Y	Y	N
`[:lower:]`	a lowercase letter	Y	Y	N
`[:upper:]`	an uppercase letter	Y	Y	N
`[:punct:]`	a punctuation symbol	Y	Y	N

Reading out regular expressions

You find below some relevant examples of regular expressions, together with the way each of them would be expressed in plain English. The same list of regular expressions in reported in the table on the next page, with examples of which strings they do or do not match.

Expression	**reads out as...**
`arch`	the string "`arch`", i.e., an '`a`', followed by an '`r`', followed by a '`c`', followed by an '`h`'
`...`	a sequence of any three characters
`ab.c`	the string "`ab`", followed by any single character, followed by the character '`c`'
`^arch`	a string that starts with the substring "`arch`"
`ber$`	a string that ends with the substring "`ber`"
`^co.*ler$`	a string that starts with "`co`", followed by zero or more characters, and ends with "`ler`"
`a.*t`	the character '`a`', followed by zero or more occurrences of any single character, followed by a '`t`'
`(at){2,}`	the string "`at`" repeated two or more times
`at.+(at){1,}`	a string that contains the string "`at`", followed by at least one of any character, followed by the string "`at`" repeated one or more times
`^[aeiou]`	a string that starts with a lowercase vowel
`[aeiou]{2,}`	a string that contains a sequence of two or more vowels
`^[^aeiouAEIOU]`	a string that does not start with a vowel (either lowercase or uppercase). Equivalently, a string that starts with a consonant! (not a vowel)
`[0-9]+.*[g-j]{2}$`	A string that contains a sequence of one or more digits (a number), then zero or more characters, and ends with two contiguous letters from the set ['`g`', '`h`', '`i`', '`j`']
`(jo|ra)b`	a string that contains either the string "`jo`" or the string "`ra`", each followed by the letter '`b`'
`1[0-9]{2}`	a string containing a '`1`' followed by any two digits
`^[0-9]+$`	a string that consists exclusively of digits (at least one!)
`^([-a-zA-Z0-9_]+\.){2}uk$`	a string consisting of exactly two sequences of the form "one or more letters or digits or '`-`' (dash) or '_' (underscore) followed by a "`.`" (dot)", then ending with "`uk`"

Examples

The same set of regular expressions reported in the table above, with relevant examples of which strings they would or would not match.

Expression	Matches	Does not match
`arch`	“March”, “archery”, “starch”	“ARCH”, “arc”, “Arch”
`...`	“matter”, “abc”, “a_b”, “b2c”	“ab”, “” (empty string), “13”
`ab.c`	“abacus”, “slab chain”, “abscess”, “Arabic”	“abcd”, “chamber”, “Abacus”
`^arch`	“arch”, “archery”, “archeology”	“March”, “under the arch”, “Archery”
`ber$`	“amber”, “September”, “number” “chamber”	“beer”, “bergamot”, “numbered”, “cumbersome”
`^co.*ler$`	“compiler”, “cooler”, “cobbler”	“toddler”, “cooled”
`a.*t`	“at”, “hat”, “yachts”, “compartment”, “aesthetic”, “breakfast”, “alphabet”	“bit”, “mash”, “At”, “haT”
`(at){2,}`	“datatype”, “catatonic”	“latter”, “athletics”, “wombat”
`at.+(at){1,}`	“bathmat”, “attenuated”, “saturation”	“datasets”, “datatype”
`^[aeiou]`	“aerospace”, “i”, “union”	“Ihaho”, “trampoline”, “22catch”
`[aeiou]{2,}`	“armour”, “Lilliputian”, “raining”, “queue	“TRAIN”, “last”
`^[^aeiouAEIOU]`	“train”, “random”, “Hotel”	“indigo”, “orange”, “Apples”
`[0-9]+.*[g-j]2$`	“write 22 emoji”, “120-laugh”, “give me 1 egg”	“have a laugh”, “I got 3 pigs”, “You need to go through it”
`(jo|ra)b`	“my new job is fun”, “crabs”, “I found a scarab”	“jab”, “rub”, “jorab”
`1[0-9]{2}`	“178”, “alpha-102-bravo”, “197865”, “231462”, “Capt. Sir Tom Moore died at age 100”	“22.87”, “1st position”, “232462”
`^[0-9]+$`	“7”, “18764562340”, “888”	“alpha33”, “3 and me”, “78 times 22”
`^([-a-zA-Z0-9_]+\.){2}uk$`	“yahoo.co.uk”, “your-email.co.uk”, “data.gov.uk”, “02_real.mean.uk”	“www.yahoo.co.uk”, “mail.uk.gov”, “lab.inria.fr”, “sunsite.unc.edu”