Character Sets specify one or more characters that may or may not appear in the string. They may be as simple as the characters you want to match, but you can also use things like "." which matches any character or `[a-z]` which would match any lower case letter. The following table shows the main ways to specify a character set:

Character Set	**Matches**
`The`	Any occurrence of "the". Matches the, but also matches **the**m, **the**re, ba**the**, etc.
`.`	Any single occurrence of any character (except newline)
`[a]`	A single occurrence of the character "a"
`[abc]`	A single occurrence either the character "a" the character "b" or the character "c"
`[a-z]`	A single occurrence of any lower case letter. (in csh/tcsh `setenv LC_ALL 'C'` or in sh/bash `export LC_ALL=C` to avoid matching uppercase alpha too.
`[A-Z]`	A single occurrence of any upper case letter
`[a-zA-Z]`	A single occurrence of any letter, either upper or lower case
`[0]`	The character "0"
`[012]`	A single occurrence either the character "0" the character "1" or the character "2"
`[0-9]`	Any numeral
`[0-9A-Za-z]`	Any numeral or any upper or lower case letter
`[^a]`	Any character that is **not** the character "a" (The ^ means not)
`[^a-z]`	Any character that is **not** a lower case letter
`[^1]`	Any character that is **not** the numeral character "1"
`[^0-9]`	Any character that is **not** a number
`[-a-z]`	Any lower case letter or a "-"
`[0-9-]`	Any number or a "-"
`[^-a-z]`	Any character except a lower case letter or a "-"
`[]0-9]`	Any number or a "]"
`]`	The character "]"
`[0-9]]`	Any number followed by the character "]"
`[0-9\]]`	Any number or the character "]"
`[0\-9]`	The character "0" or the character "-" or the character "9"
`[\^1]`	The character "^" or the character "1"

Examples

`'[0-9][0-9]'`	Any number 00-99
`'Room [1-2][0-9][0-9]'`	"`Room`" followed by 100 through 299
`'Title: ....[0-9]'`	"`Title:`" followed by any 4 characters, followed by 1-9

Occurrence Modifiers are placed after a character to say how many times that character may appear. For example "`*`" says there may be zero or more occurrences of the ***previous*** character.

Modifier	**Meaning**
`*`	zero or more occurrences of the preceding
`\?`	zero or one occurrences of the preceding
`\+`	one or more occurrences of the preceding
`\{x\}`	x number of occurrences of the preceding
`\{x,y\}`	x to y number of occurrences of the preceding
`regexp1 | regexp2`	regular expression 1 **or** Regular expression 2

Examples

`'[0-9]*'`	Nothing or any number of digits. For example `123` or `99` or `4509`
`'.*'`	Nothing, or any number of any characters
`'[0-9]\{3\}'`	Any 3 character number `000-999`, same as `[0-9][0-9][0-9]`
`'[0-9]\{3,4\}'`	Any 3 or 4 character number `000-9999`
`'RBI | Runs Batted In'`	The string "`RBI`" or the string "`Runs Batted In`"

Positional Modifers or Anchors specify the string's position in the line. There are two anchors, `^` and `$`. `^` represents the start of a line, while `$` represents the end of the line. For the anchors to function as anchors they must be in the For example `^Tony` would match the string "`Tony`" only if it were the first thing on the line. `Tony$` would match the string "`Tony`" only if it were the last thing on the line.

If you use the ^ or $ anywhere besides the start or end respectively, then they will be interpreted as a normal character. For example in the regular expression "`This is a caret ^`" the ^ is just a caret and loses any special meaning since it's not at the beginning of the line.

Positional modifiers can also be used to specify that a pattern must be after the beginning of a word, or before the end of a word. A word is defined as a set of characters with a space character or newline before and after the characters. To say that the characters "red" need to be in word by themselves the regular expression would be '`\<red\>`' Ideally we'd like to just write '`<red>`', but the < and > need to be protected so they each have to be proceeded by a slash. Note that '`\<red\>`' will match red at the beginning and end of a line as well, which is what makes it different than just using spaces before and after the word as in '` red `'

Regular Expression	Meaning
^	Beginning of line
$	End of line
\<	Beginning of word
\>	End of word

Examples

`'^Tony'`	The string `Tony` at the beginning of a line
`'Tony$'`	The string `Tony` at the end of a line
`'^[0-9][0-9]'`	Any number `00-99` that are the first characters in a line
`'Room [1-2][0-9][0-9]$'`	"`Room`" followed by `100` through `299` as the last characters in a line
`'^Title: ....[0-9]$'`	"`Title:`" followed by any 4 characters, followed by 1-9 as the only characters on a line
`'\<the\>'`	The word "`the`". Does **not** match "`then`" or "`breathe`"
`'^\<Tony\>'`	The word "`Tony`" as the first word on the line

PERL Extensions
NOTE - these may not be supported or you may have to use `grep -P`

\d	Any single number	`[0-9]`
\D	Any character that is not a number	`[^0-9]`
\w	Any word made from alphanumeric characters and "_"	`[a-zA-Z0-9_]`
\W	Any non-word character	`[^a-zA-Z0-9_]`
\s	Any whitespace character such as (space, tab, newline)	
\S	Any non-whitespace character	
\n	The newline character	
\r	The carriage return character	
\t	The tab character	
\nnn	The ASCII character with the octal value nnn	
\xnn	The ASCII character with the hex value nn	

POSIX Character Classes

POSIX Character Classes			
POSIX	PERL	Description	Classic regular expression
[:alnum:]		Alphanumeric characters	[a-zA-Z0-9]
[:alpha:]		Alphabetic characters	[a-zA-Z]
[:ascii:]		ASCII characters	[\x00-\x7F]
[:blank:]		Space and tab	[\t]
[:cntrl:]		Control characters	[\x00-\x1F\x7F]
[:digit:]	\d	Digits	[0-9]
[:graph:]		Visible characters (i.e. anything except spaces, control characters, etc.)	[\x21-\x7E]
[:lower:]		Lowercase letters	[a-z]
[:print:]		Visible characters and spaces (i.e. anything except control characters, etc.)	[\x20-\x7E]
[:punct:]		Punctuation and symbols.	[!"#$%&'()*+,\-./:;<=>?@[\\\]^_`{|}~]
[:space:]	\s	All whitespace characters, including line breaks	[\t\r\n\v\f]
[:upper:]		Uppercase letters	[A-Z]
[:word:]	\w	Word characters (letters, numbers and underscores)	[A-Za-z0-9_]
[:xdigit:]		Hexadecimal digits	[A-Fa-f0-9]