Regex: special character classes

There are certain character classes that are so frequently used that a special sequence was created for them. A special sequence, that will make the code shorter and more readable.

\d - digit - [0-9]

\d stands for "digit". It is the same as [0-9] except that it is much shorter to write and easier to read. So the regex /\d/ would match a single digit. There are two ways to use this special character class. One is stand-alone as we just saw it or in this example: /#[XYZ]-\d\d\d/ that will match strings like these:

```
#X-123

#Y-666
```

In this case we can put quantifiers on the character class. For example /#[XYZ]-\d{3}/ would mean the same as the previous regex and /#[XYZ]-\d+/ would allow any number of digits (at least one) after the dash.

The other usage is to put the \d in square brackets as part of a larger character class. For example[\dABCDEF] would match a single digit or one of the capital letters A, B, C, D, E, or F.

This is a character class that can match a digit of a hexa-decimal number.

Actually, we would probably not want to spell out the 6 letters but would write [\dA-F] and looking at this example I am not even sure I would not write [0-9A-Z] instead. It is a bit longer, but in this case it feels clearer.

In a nutshell, they do the same, write the one that looks more readable to you.

\w - word character - [0-9a-zA-Z_]

Similar to \d for digits, there is also \w that stands for "word characters". In this case "word characters" mean digits, ASCII letters, and the underscore. Actually this will also match "other connector punctuation chars plus Unicode marks" so they are not exactly the same. For specific definition look at the documentation.

\s - whitespace - [\t\n\f\r]

The third common character class sequence is \s representing a whitespace. The 5 characters it matches are horizontal tab, the newline, the form feed, the carriage return, and the space. In recent version of Perl there are some experimental changes that will probably not affect you. Check out the details if in doubt.

\D, \W, \S - negated character classes

As you might remember putting a caret as the first character in a character-class means that the character-class is negated. It will match any single character **except** the ones listed. So [^\d] would mean any non-digit character.

To make it easier to write this, perl provides the \D character-class which means exactly the same.

Similar to \D there are also negated versions of the other two character classes:

```
Character Class         Negated       Meaning
\d                      \D            [^\d]
\w                      \W            [^\w]
\s                      \S            [^\s]
```

POSIX character classes

POSIX is a family of standards for maintaining compatibility between operating systems. Among other things it also specifies a number of "groups of characters" with a name. It has groups called alpha, alnum, etc. Perl provides special sequences to match a single character in one of these groups.

The syntax is [:alpha:] to match a single character from the group called alpha.

The POSIX character classes can only be used inside a bigger character class, so you will need to write /[[:alpha:]]/ if you really wanted to match a character in the alpha group.

Actually it is very unlikely that you will need these character classes, but if you do, you can check out more details about them in the documentation.

Unicode character classes

Unicode is a definition of "all" the existing characters in the world. More than 100,000. Each character is described as a character point, but certain characters are also grouped together. You can use the \p{...}notation to match a single character from one of these groups.

For example if you need to match a Thai character you can use the \p{Thai} expression. If you need to match anything except a Thai character you can use the corresponding \P{Thai} expression.

You can use these as stand-alone character classes /\p{Thai}/ or as part of a larger character class:/[\p{Thai}\p{Arabic}]/.

There are, of course, a lot more details, so if you need to handle interesting characters, you should read the documentation

```
Expression    Meaning
Usage

\w            Word characters: [a-zA-Z0-9_]                        \w+
or [\w#-]+

\d            Digits: [0-9]

\s            [\f\t\n\r ]

              form-feed, tab, newline, carriage return and SPACE

\W            [^\w]

\D            [^\d]

\S            [^\s]

[:class:]    POSIX character classes (alpha, alnum...)
[[:alpha:]]+ or [[:alpha:]#-]+

\p{...}      Unicode definitions (Alpha, Lower, That, ...)
\p{Thai}+ or [\p{Thai}#!-]

\P{...}      Negation of Unicode character class
```