Because Perl makes heavy use of strings, regular expressions are a very important component of the language. They can be used:

- in conditional expressions to test whether a string matches a pattern
e.g. checking the contents of a string
  ```perl
  if ($name =~ /[0-9]/) { print "name contains digit\n"; }
  ```
- in assignments to modify the value of a string
e.g. convert McDonald to MacDonald
  ```perl
  $name =~ s/Mc/Mac/;
  ```
e.g. convert to upper case
  ```perl
  $string =~ tr/a-z/A-Z/;
  ```

Perl extends POSIX regular expressions with some shorthand:

- \d matches any digit, i.e. [0-9]
- \D matches any non-digit, i.e. [~0-9]
- \w matches any "word" char, i.e. [a-zA-Z_0-9]
- \W matches any non "word" char, i.e. [~a-zA-Z_0-9]
- \s matches any whitespace, i.e. [ \t\n\r\f]
- \S matches any non-whitespace, i.e. [~ \t\n\r\f]

Perl also adds some new anchors to regexps:

- \b matches at a word boundary
- \B matches except at a word boundary

And generalises the repetition operators:

- patt* matches 0 or more occurrences of patt
- patt+ matches 1 or more occurrences of patt
- patt? matches 0 or 1 occurrence of patt
- patt{n,m} matches between n and m occurrences of patt
Perl Regular Expressions

The default semantics for pattern matching is "first, then largest". E.g. /ab+/ matches abbbabbb not abbbabbbb or abbbbabbb

A pattern can also be qualified so that it looks for the shortest match. If the repetition operator is followed by ? the "first, then shortest" string that matches the pattern is chosen. E.g. /ab+/? would match abbbabbb

Perl Regular Expressions

Regular expressions can be formed by interpolating strings in between /.../.
Example:

```
$pattern = "ab+";
$replace = "Yod";
$text = "abba";

$text =~ s/$pattern/$replace/;
```

# converts "abba" to "Yoda"

Note: Perl doesn’t confuse the use of $ in $var and abc$, because the anchor occurs at the end.

Using Matching Results

In a scalar context matching & substitute operators return how many times the match/substitute succeeded. This allows them to be used as the controlling expression in if/while statements. For example:

```
print "Destroy the file system? 

if ($answer = <STDIN>) {
    system "rm -r ";
}

s/[aeiou]//g or die "no vowels to replace";
```

Using Matching Results

In a list context the matching operators returns a list of the matched strings. For example:

```
$string = "-5==10zzz200_";
@numbers = $string =~ /\d+/g;
print join(",", @numbers), "\n";
```

# prints 5,10,200

If the regex contains ()s only the captured text is returned

```
$string = "Bradley, Marion Zimmer";
($family_name, $given_name) = $string =~ /([^-]+), (\S+)/;
print "$given_name $family_name\n";
```

# prints Marion Bradley
Pattern Matcher

A Perl script to accept a pattern and a string and show the match (if any):

#!/usr/bin/perl

$pattern = $ARGV[0]; print "pattern=/$pattern/\n";

$string = $ARGV[1]; print "string ="$string"\n";

$string =~ /$pattern/; print "match ="$&"\n";

You might find this a useful tool to test out your understanding of regular expressions.