command shell

- The quotes above are not a part of the regular expression but are needed by the

```
grep ""(FROM|Subject|Date): "folder"
```

- Checking the author, subject, and date of each message in a given mail folder

and replace strings

Used by several Unix utilities such as ed, vi, emacs, grep, sed, and awk to search for

mark characters to select filenames

You have already used them in scripting files when you used asterisk (*) and question

or more strings

Defined as a string composed of letters, numbers, and special symbols, that defines one

Key to powerful, efficient, and flexible text processing

Regular expressions
are the words and the metacharacters are the grammar.

- We can treat the regular expressions as a language in which the literal characters
  \( \{ \text{a special character} \} \) are with \texttt{grep} (\texttt{grep}) are \texttt{grep}.

* \texttt{quote} and needs to be \texttt{quote}.

- A special character is one that does not represent itself (such as a metacharacter).

- All the characters that represent themselves are called \texttt{heres}.

- Most characters represent themselves within a regular expression.

- A character is any character on the keyboard except the newline character \( `\backslash n` \).

\begin{itemize}
  \item Characters, and other metacharacters defined below.
  \item Regular expression is composed of characters, delimiters, simple strings, special
\end{itemize}
Delimiters are not used with the grep family of utilities.

- Delimiters are omitted.
- If the second delimiter is to be immediately followed by a carriage return, it may be
  - More often than not, people use forward slash, `/` as the delimiter (guess why).
  - At both ends of the regular expression appearance.
  - Any character can be used as a delimiter as long as it is the same character appears
  - Regular expression.
  - The delimiter does not represent itself but marks the beginning and end of the
  - Delimiter is always a special character for the regular expression being delimited.
  - A delimiter is a character to mark the beginning and end of a regular expression.delimiters


Display the entire line in which the regular expression matches.

The regular expression search is not done on a word basis but utilizes like `RegExp`

between the beginning and the end of range characters.

* Even in this case, the dash cannot be the first character and must be enclosed.

- The dash (\-) is considered to be a metacharacter only within the square brackets.

- Extended regular expressions (such as the one used by `RegExp`) have been added to the above for

- In addition, the following metacharacters have been added to the above for

  \( \) \[ \] \{ \} \* . $ ~
- Simple strings
  - The most basic regular expression
    - Matches only itself

<table>
<thead>
<tr>
<th>Reg. Exp.</th>
<th>Matches</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>/or not/</td>
<td>Thursday's</td>
<td>ringing</td>
</tr>
<tr>
<td>or not</td>
<td>Thursday</td>
<td>stringing</td>
</tr>
<tr>
<td>or not</td>
<td>ring</td>
<td>spring</td>
</tr>
<tr>
<td>poor nothing</td>
<td>ring/</td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td>Matches</td>
<td>Period</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>09-17-98 /09.17.98 /09/17/98</td>
<td>Date with any separator preceded by slash</td>
<td>-</td>
</tr>
</tbody>
</table>
| All strings that contain a space will talk | * | Cause a regular expression to match more than one string
| preceded by slash | * | Special characters |
| followed by slash | * | |
Within a character class definition, the special characters (backslash, asterisk, and dollar signs) lose their special meaning when within square brackets.

Within a character class definition, a range of characters can only appear as the first character immediately following the square brackets.

A right square bracket appearing as a member of the character class can only appear as the first character immediately following the square brackets.

A dot within square brackets will not be a metacharacter.

A metacharacter will not match 07/17/98 but will match 07-17-98.

A metacharacter will not be a metacharacter.

A metacharacter will not be a metacharacter.
<table>
<thead>
<tr>
<th>Examples</th>
<th>Matches</th>
<th>RegExp Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>/k/</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>/b/</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>/K/</td>
<td></td>
</tr>
</tbody>
</table>

Stop with a character that is not a letter, any character, and a k

Get number 9 class 6 through 9 and a member of the character number 8: number 60 followed by a space

 tanker
 teach
 think
 talkative

billed
 Billed
 Member of the character class

Examples *
possible string, starting as far toward the beginning (left) of the line as possible.

A regular expression that includes a special character always matches the longest
characters that are members of the character class
      A character class definition followed by an asterisk matches any string of
      An asterisk following a period matches any string of characters
      Represents zero or more occurrences of a match of the regular expression
      Can follow a regular expression that represents a single character

—Asterisk
| Get (this and that) that string starts with (and ends with) this | /(**)(~)/ |
| Get (this) and (that) between (and) | /(**.)(~)/ |
| Get (this) and (that) between (and) | /(*[a-zA-Z]*)/ |
| numbers or punctuation | a string composed only of letters | /[*[a-zA-Z]*]/ |
| I thought of going | time | characters followed by one or more |
| | | characters followed by zero or more |
| x=26.345 x cat | ab| characters followed by a |
| ab| characters followed by a c |
| abc| characters followed by a c |
| abc| characters followed by a c |
| abc| characters followed by a c |
| abc| characters followed by a c |
| abc| characters followed by a c |

**Examples**

<table>
<thead>
<tr>
<th>Matches</th>
<th>Exp.</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>
**Examples**

- Quoting a special character makes it represent itself.
- Proceeding it with a backslash
  - Any special character, except a digit or a parenthesis, can be quoted by
    "Quoting special characters"
You can also do the following:

If you want to make the same substitution again, the following is sufficient:

Let us give the following command to `vit`:

An empty regular expression always represents the last regular expression used.

Empty regular expressions towards the beginning of the line are possible.

A regular expression always matches the longest possible string starting as far

Longest match possible.

Rules
The expression \texttt{/\texttt{dxer}/} and \texttt{/(\texttt{dxer})\textbackslash/} will match the
same patterns.

A regular expression within the quoted parentheses matches exactly what the
quoted digits.

The regular expression does not attempt to match quoted parentheses.

The string matching the bracketed regular expression can be subsequently used as
\texttt{\textbackslash(} and \texttt{\textbackslash)}.

Bracketed expressions

\texttt{\textbackslash}
Example

There is no ambiguity in identifying the nested quoted parentheses as they are identified by the opening \\

by the following `\command`

It can be changed to the format

Assume a list of people in the format

within the regular expression, a quoted digit (\u) takes on the value of the

- Quoted digits

3 t DMNORx7 I u

matches

(\x(\*[Z-Za-z]\*[Z-Za-z])\)

Example *

There is no ambiguity in identifying the nested quoted parentheses as they are identified by the opening \\

by the following `\command`

It can be changed to the format

Assume a list of people in the format

within the regular expression, a quoted digit (\u) takes on the value of the

- Quoted digits
Regular expressions cannot be used for the newline character.

- End of word: `<\`
- Beginning of word: `>\`

Expressed by:
- Character, period, end-of-line, or beginning of line.

The word boundaries in the regular expressions are denoted by any whitespace.

**Example:**

/\$\*\ Number/[6-0][6-0]/s:

- An ampersand takes on the value of the string that the search string matched.
- Strings within the replacement string can be used to match the replacement.
- Ampersands (\$) and quoted digits (\{\}) can be used to use regular expressions as search strings with the substitute command.

**Replacement string:**

```
\$\*\ Number/[6-0][6-0]/s:
```