Regular expressions

Houcine Senoussi

Regular expressions

Definition :

A regular expression (*regex*) is a pattern describing a set of strings. It is defined as a sequence of characters and metacharacters. Characters have literal meaning, metachracters have special meaning.

- Examples :
  1. In *Books*?, the characters 'B', 'o', 'k' and 's' are literal characters and ? is a metacharacter meaning **zero or one occurence** of the preceding character. It follows that this regex matches the two strings *Book* and *Books*.
  2. The regex $a+b$ matches any string beginning with **one or more** a (this is the meaning of the metacharacter $+$) followed by an ending b.

Forming regular expressions

- To define regular expressions we first need **characters** and **character classes**.
  1. Characters are, for example, those of ASCII or Unicode. Characters that are used as metacharacters (e.g. (, }, \, ...) need to be despecialized to be used as simple characters.
  2. Character classes are defined by (small) sequences of characters, containing at least one metacharacter and intended to match a set of characters. For example [0-9] represents the set of digits.

Forming regular expressions(2)

- Using characters and classes of characters, regular expressions are built as follows :
  1. The empty string ϵ is a regular expression.
  2. Each character is a regular expression, and each sequence of characters representing a character class is a regular expression.
  3. The **concatenation** of two regular expressions is a regular expression.
  4. The **alternation** of two regular expressions is a regular expression. It matches the **union** of the sets matched by the regular expressions.
  5. It follows from $1 - 4$ that **repetitions** 0, 1 or more times of a regular expression is a regular expression : the results of such repetitions are equal respectively to ϵ, the regular expression itself and a concatenation of the expression and a repetition of it.

Forming regular expressions(3)

- It follows from the above that, to define regular expressions over a given set of characters (by the defaut ASCII or Unicode characters), we need :
  1. a set of metacharacters,
  2. a set of character classes and/or methods to define character classes,
  3. definitions of the concatenation, alternation and repetition operators.
- In the following slides we present such a formalism. This formalism is the one used by several programming languages (e.g. Python and Perl);

Regular expressions-Metacharacters

Characters	Meaning
.	Matches any character except newline.
^ , \$	Match the start and and the end of a string, respectively.
|	Boolean OR : *expr*1|*expr*2 matches *expr*1 and *expr*2.
?, $*$, $+$ {n}, {min,max} {min,}, {,max}	Repetition qualifiers.

Regular expressions-Metacharacters

Characters	**Meaning**
[]	Indicates a set of characters.
[^]	Indicates the complement of a set of characters.
()	Grouping : Matches whatever the expression is inside the parentheses.
\	(1) defines special sequences (2) despecializes special characters.

Regular expression syntax

- Classes of characters are defined as **sets** and as **wild cards**. Sets are defined using [] and a few other metachracters. The more general wildcard is defined by the metacharacter "." and it matches all characters, except newline. Other wildcards are defined using the metacharacter \.

Sets of characters

Definition	Example
Set of individual characters	$[aeiouy]$
Range of characters	$[a-z]$
Union of sets	$[0-9a-fA-F]$,
Complement of a set	[ˆ a-b]

- The character '-' has a literal meaning if it is escaped using the metacharacter '\', or placed as the first or the last character of a set.
- Special characters lose their special meaning inside sets. For example, $[\backslash()\{\}?+*]$ will match any of the literal characters put between [and].

Wildcards

Definition	**Meaning**
.	Matches any character except newline.
$\backslash d$	Matches digits.
$\backslash D$	Matches characters other than digits.
$\backslash w$	Matches word characters.
$\backslash W$	Matches non word characters.
$\backslash s$	Matches whitespace(space, tab, newline).
$\backslash S$	Matches characters other than whitespace.
$\backslash b$	Matches beginning and end of words.
$\backslash A$	Matches beginning of words.
$\backslash Z$	Matches end of words.
$\backslash B$	Matches middle of words.

Repetition qualifiers

Characters	**Meaning**
?	Zero or one repetition.
$*$	Zero or more repetitions.
$+$	One or more repetitions.
$\{n\}$	*n* repetitions
$\{min, max\}$	*min* to *max* repetitions
$\{min, \}$	At least *min* repetitions
$\{, max\}$	At most *max* repetitions

Regular expressions-Examples

- Any string is a regular expression that matches itself.
- $[AB]\backslash\backslash\backslash\{[xy]\backslash\}$ matches $A\backslash\{x\}$, $A\backslash\{y\}$, $B\backslash\{x\}$ and $B\backslash\{y\}$.
- $1(000)*$ matches 1, 1000, 1000000, ...
- $M(r|s|rs)\backslash.?$ matches Mr, $Mr.$, Ms, $Mr.$, Mrs and $Mrs.$.
- $\backslash(.+\backslash)$ matches any (non empty) substring between parentheses.
- $(0|1)*$ matches any binary string.
- $[01]*$ matches any binary string.
- $[0-9a-fA-F]*$ matches any hexadecimal string.

Regular expressions in Python

- Python has a module called **re** for working with regular expressions.
- The syntax of regular expressions in Python is the one described in the previous slides.
- The main functions of this module are :
  1. *findall*: returns all non-overlapping matches of a regular expression in a given string.
  2. *finditer*: does the same as the previous function with more details.
  3. *search*: if the regex matches one or more substrings, this function returns the first match.
  4. *split*: splits the string into a list of substrings delimited by regex.
  5. *sub*: replaces every occurrence of the regex with a string.
- See the file *forReg.py* for more details.