Patterns in Lua*

The first example in the manual# is to extract the digits (%d) making up a date in a text string, s.

```
s = "Deadline is 30/05/1999, firm"
```

The search pattern is going to be this:

```
date = "%d%d/%d%d/%d%d%d%d"
```

In the lua interpreter:

```
string.find(s, date) -> 13 22
```

The string.find() function tells us that the first and last digits which match the pattern are at positions 13 and 22.

We can extract this substring:

```
print(string.sub(s,13,22)) -> 30/05/1999
```

Nesting the string.find() and string.sub() functions gives the same end result.

```
print(string.sub(s, string.find(s, date))) ->
30/05/1999
```

The character classes:

`.`	for all characters,
`%a`	for letters,
`%c`	for control characters,
`%d`	for single character integers,
`%l`	for lower case letters,
`%p`	for punctuation characters,
`%s`	for space characters,
`%u`	for upper case letters,
`%w`	for alphanumeric characters,
`%x`	for hexadecimal digits and
`%z`	for the zero character.

To invert a selection use upper case letters e.g. %A would select all characters which are not letters.

The next example in the manual uses the global substitution command.

```
string.gsub("hello, up-down!", "%A", ".") ->
hello..up.down. 4
```

The 2nd parameter is the pattern for non-letter characters that are to be replaced with dots. Gsub tells us that there are 4 substitutions.

The search pattern for find(), sub() and gsub() can, of course, also just be a plain string, as below.

```
string.gsub("hello, up-down!", "e", "a") ->
hallo, up-down! 1
```

The magic characters used in patterns are:

```
( ) . % + - * ? [ ] ^ $
```

The % symbol is used to turn off the normal meaning of a character e.g. "%a" means that we are looking for all letters rather than just the letter "a" itself. Hence the % '*escapes*' the character from its normal meaning.

This example replaces every letter with an "X".

```
string.gsub("hello, up-down!", "%a", "X") ->
XXXXX, XX-XXXX! 11
```

The "+" symbol is to get the longest sequence that matches the pattern e.g. with "%a" which looks for single letters, we can make it look for groups of letters (i.e. words) by saying "%a+" instead. This replaces every word with "X":

```
string.gsub("hello, up-down!", "%a+", "X") ->
X, X-X! 3
```

Square brackets [] allow you to define a character class e.g. to replace every letter "e", "l" or "o" with "X":

```
string.gsub("hello, up-down!", "[elo]", "X") ->
hXXXX, up-dXwn! 5
```

Or if any of those three characters occur together, we might want to put in only one "X":

```
string.gsub("hello, up-down!", "[elo]+", "X")
-> hX, up-dXwn!    2
```

Let's say we had the word "hello" typed incorrectly as "he[]o" and we wanted to replace the "[]" with "ll"

```
string.gsub("he[]o, up-down!", "[]", "ll") ->
error
```

That fails because [and] are magic characters so we need to escape their special meaning with the % sign as follows.

```
string.gsub("he[]o, up-down!", "%[%]", "ll") ->
hello, up-down! 1
```

Note %b allows balanced string pattern matches e.g. to replace or extract strings within brackets, etc.

```
string.gsub("he[xyz]o, up-down!", "%b[]", "ll")
-> hello, up-down! 1
string.match("he[joe]o, up-down!", "%b[]") ->
[joe]
```

The ? magic character is for an optional match (i.e. 0 or 1 occurrence) and * is for 0 or more matching characters (- is similar). If we are looking for +, - or a decimal point in a string, e.g. to pull out a real number, we need to de-magic these characters with a %, as shown here:

```
string.match("The temperature is -23.1 degrees
Centigrade.","%-?%d*%.?%d+") -> -23.1
```

To grab all real numbers in a string, s, where:

```
s = "The data are 17.8,-16.3, +.2,22., +5 and
14.5."
```

we can do this:

```
for i in string.gmatch(s,"%-?%d*%.?%d+") do
print(i) end -> 17.8 -16.3 .2 22 5 14.5
```

To grab separate parts of a string we can use *captures* which are defined with ( )'s.

```
s = "Pi is 22/7 approx."
top, bottom = string.match(s,"(%d+)/(%d+)")
print(top, bottom) -> 22 7
```

To grab the fraction as text would be:

```
pi = string.match(s,"%d+/%d+")
print(pi) -> 22/7
```

The / is not a magic character so the % is not needed.

To split a string into words we look for everything that is not a space. Note that %S is the complement of %s.

```
for i in string.gmatch(s,"%S+") do print(i)
end ->   Pi   is   22/7     approx.
```

The ^ symbol usually means the complement e.g. the above code works with "[^%s]+" instead of "%S+"

To blank out everything in string, s, except the fraction:

```
fraction_text=string.gsub(s,"[^%d/]","")
print(fraction_text) -> 22/7
```

If ^ occurs at the start of the pattern, it matches something at the start of a string and $ matches something at the end.

To check if a string consists only of an integer we can use:

```
string.match(s, "^[+-]?%d+$")
```

* Lua's semantics for pattern matching are a bit different from POSIX regex and they are a bit easier to grasp.

https://www.lua.org/pil/20.2.html