

String Handling

old style: `import string; string.find(str, pattern)`

new style: `str.find(substr)`

```
''%s circus'' % (''flying'') # string formatting
r''bl\a'' # raw string, escape sequences ignored
index(substr), rindex(substr) # raises ValueError
find(substr), rfind(substr) # -1 on failure
replace(old, new) # replace all substrings old with new
count(substr) # occurrences of substr
upper(), lower() # convert to upper/lower case
substr in str # True if substr part of str
strip(chars), lstrip(chars), rstrip(chars) # default chars is all whitespace
split(separator), join(separator) # default separator is all whitespace
```

Regular Expression

... is a string used to describe or match a set of strings, according to certain syntax rules.

Special Character	Meaning
<code>bla</code>	Matches the string <code>bla</code>
<code>.</code>	Matches any character except newline
<code>^</code>	Matches the start of the string
<code>\$</code>	Matches the end of the string
<code>*</code>	Any number of occurrences of what just preceded me
<code>+</code>	One or more occurrences of what just preceded me
<code> </code>	Either the thing before or after me
<code>\w, \d, \s</code>	Matches alphanumeric, decimal, whitespace character
<code>\\</code>	Matches backslash character
<code>?, (), [], ...</code>	Option, grouping, characters, ...

RE Examples

<code>.at</code>	hat, cat, bat, . . .
<code>[hc]at</code>	hat, cat
<code>su+per</code>	super, suuper, suuuper, . . .
<code>-?\d+</code>	all integers
<code>^\s*</code>	see if ; is at beginning of string
<code>i\+\+ (=1)</code>	<code>i++</code> , <code>i+=1</code>
<code>((great)*grand)?((fa mo)ther)</code>	father, mother, grand father, grand mother, great grand father, . . .

RE in Python

See <http://www.amk.ca/python/howto/regex/>.

```
import re
if re.match('su+per(duper)?', 'suuuper'): print 'matches'
ex = re.compile(r'<(P<sometag>\w+)>$')
m = ex.match('<xml><bla>')
if not m: print 'no'
m = ex.search('<html><bla>')
if m: print m.groupdict()['sometag'] # 'bla'
```

File Parsing

```
import string, random
f = open('csv.txt', 'rt') # contains lines words separated with comma
for line in f:
    words = line.split(','); random.shuffle(words)
    print string.join(words, ';'), # output permuted words with ;
```

```
import fileinput, re, sys
comment = re.compile(r'^\s*|#')
for line in fileinput.input(sys.argv[1:]):
    if comment.match(line): continue # skip lines beginning with ; or #
    print line,
```

Common Log Format

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

remote host

ident (RFC 1413)

login as which the user authenticated

time and date

request line

status code (200, 404)

bytes (content-length) transferred

Bibliography

Mark Lutz, David Ascher, Learning Python, 2nd ed., O'Reilly, 2003.
Alex Martinelli, Python in a nutshell, 2nd ed., O'Reilly, 2006.