```
Basic operations (NUMBERS)
2 3 0 0.333333333333
<type 'int'>
<type 'complex'>
1.0 1.0 1.41421356237
Basic operations (STRINGS)
<type 'str'>
STRING
STRING
STRING
STRING
LONG STRING

8 ['LONG', 'STRING']
LONG LONG STRING
LONG#LONG#STRING
LONGSTRING
Function DOC. Example type
***************
type(object) -> the object's type
type(name, bases, dict) -> a new type
***************
The time is 12:30pm!
<_sre.SRE_Match object at 0x7f6cf7363990>
12:30
('12:30',)
<_sre.SRE_Match object at 0x7f6cf7363918>
2:30
The time is 2:10pm!
***************
Support for regular expressions (RE).

This module provides regular expression matching operations similar to
those found in Perl.  It supports both 8-bit and Unicode strings; both
the pattern and the strings being processed can contain null bytes and
characters outside the US ASCII range.

Regular expressions can contain both special and ordinary characters.
Most ordinary characters, like "A", "a", or "0", are the simplest
regular expressions; they simply match themselves.  You can
concatenate ordinary characters, so last matches the string 'last'.

The special characters are:
    "."      Matches any character except a newline.
    "^"      Matches the start of the string.
    "$"      Matches the end of the string or just before the newline at
             the end of the string.
    "*"      Matches 0 or more (greedy) repetitions of the preceding RE.
             Greedy means that it will match as many repetitions as possible.
    "+"      Matches 1 or more (greedy) repetitions of the preceding RE.
    "?"      Matches 0 or 1 (greedy) of the preceding RE.
    *?,+?,?? Non-greedy versions of the previous three special characters.
    {m,n}    Matches from m to n repetitions of the preceding RE.
    {m,n}?   Non-greedy version of the above.
    "\\"     Either escapes special characters or signals a special sequence.
    []       Indicates a set of characters.
             A "^" as the first character indicates a complementing set.
    "|"      A|B, creates an RE that will match either A or B.
    (...)    Matches the RE inside the parentheses.
             The contents can be retrieved or matched later in the string.
    (?iLmsux) Set the I, L, M, S, U, or X flag for the RE (see below).
    (?:...)  Non-grouping version of regular parentheses.
    (?P<name>...) The substring matched by the group is accessible by name.
    (?P=name)     Matches the text matched earlier by the group named name.
    (?#...)  A comment; ignored.
    (?=...)  Matches if ... matches next, but doesn't consume the string.
    (?!...)  Matches if ... doesn't match next.
    (?<=...) Matches if preceded by ... (must be fixed length).
    (?<!...) Matches if not preceded by ... (must be fixed length).
    (?(id/name)yes|no) Matches yes pattern if the group with id/name matched,
                       the (optional) no pattern otherwise.
```

```
The special sequences consist of "\\" and a character from the list
below.  If the ordinary character is not on the list, then the
resulting RE will match the second character.
    \number  Matches the contents of the group of the same number.
    \A       Matches only at the start of the string.
    \Z       Matches only at the end of the string.
    \b       Matches the empty string, but only at the start or end of a word.
    \B       Matches the empty string, but not at the start or end of a word.
    \d       Matches any decimal digit; equivalent to the set [0-9].
    \D       Matches any non-digit character; equivalent to the set [^0-9].
    \s       Matches any whitespace character; equivalent to [ \t\n\r\f\v].
    \S       Matches any non-whitespace character; equiv. to [^ \t\n\r\f\v].
    \w       Matches any alphanumeric character; equivalent to [a-zA-Z0-9_].
             With LOCALE, it will match the set [0-9_] plus characters defined
             as letters for the current locale.
    \W       Matches the complement of \w.
    \\       Matches a literal backslash.

This module exports the following functions:
    match    Match a regular expression pattern to the beginning of a string.
    search   Search a string for the presence of a pattern.
    sub      Substitute occurrences of a pattern found in a string.
    subn     Same as sub, but also return the number of substitutions made.
    split    Split a string by the occurrences of a pattern.
    findall  Find all occurrences of a pattern in a string.
    finditer Return an iterator yielding a match object for each match.
    compile  Compile a pattern into a RegexObject.
    purge    Clear the regular expression cache.
    escape   Backslash all non-alphanumerics in a string.

Some of the functions in this module takes flags as optional parameters:
    I  IGNORECASE  Perform case-insensitive matching.
    L  LOCALE      Make \w, \W, \b, \B, dependent on the current locale.
    M  MULTILINE   "^" matches the beginning of lines (after a newline)
                   as well as the string.
                   "$" matches the end of lines (before a newline) as well
                   as the end of the string.
    S  DOTALL      "." matches any character at all, including the newline.
    X  VERBOSE     Ignore whitespace and comments for nicer looking RE's.
    U  UNICODE     Make \w, \W, \b, \B, dependent on the Unicode locale.

This module also defines an exception 'error'.


***************
Return the string obtained by replacing the leftmost
    non-overlapping occurrences of the pattern in string by the
    replacement repl.  repl can be either a string or a callable;
    if a string, backslash escapes in it are processed.  If it is
    a callable, it's passed the match object and must return
    a replacement string to be used.
***************
Lists
[10, 11, 12, 13, 4, 4, 4, 2, 4, 6] 10
[10, 11, 12, 'ciao', 4, 4, 4, 2, 4, 6] 10
True
[10, 11, 12, 4, 4, 4, 2, 4, 6]
False
[4, 4, 4, 2]
[2, 4, 4, 4, 4, 6, 10, 11, 12] 1 7
CONTROL FLOW
1
a b c d e
dogs cats bears
FUNCTIONS
Describe the structure
5
5
this function
    adds two numbers
READING FILES
[100.0, -20.300000000000001, -31.199999999999999]
```

```
[200.0, -22.699999999999999, -33.600000000000001]
WRITING FILES
```