String Methods

This is a list of common string methods. In it, S is the string to which the method is applied, and s and t are other strings.

Operation	Description
S.`capitalize()`	If the first character of S is a letter, capitalize it
S.`count(`s`)`	Count the number of times s occurs in S
S.`endswith(`s`)`	`True` if S ends with s; `False` otherwise
S.`find(`s`)`	Return the index of the first occurrence of s in S; -1 if s not in S
S.`index(`s`)`	Return the index of the first occurrence of s in S; `ValueError` exception if s not in S
S.`isalnum()`	`True` if S contains only alphanumerics (letters and digits); `False` otherwise
S.`isalpha()`	`True` if S contains only alphabetics (letters); `False` otherwise
S.`isdigit()`	`True` if S contains only digits; `False` otherwise
S.`islower()`	`True` if all letters in S are lower case; `False` otherwise
S.`isspace()`	`True` if S contains only white space; `False` otherwise
S.`isupper()`	`True` if all letters in S are upper case; `False` otherwise
S.`lower()`	Change all upper case letters in S to lower case
S.`lstrip()`	Delete all leading white space from S and return the result
S.`replace(`s`,`t`)`	Replace all occurrences of s with t in S
S.`rfind(`s`)`	Return the index of the last occurrence of s in S; -1 if s not in S
S.`rindex(`s`)`	Return the index of the last occurrence of s in S; `ValueError` exception if s not in S
S.`rstrip()`	Delete all trailing white space from S
S.`strip()`	Delete all leading and trailing white space from S
S.`swapcase()`	Change all upper case letters in S to lower case and all lower case letters to upper case
S.`title()`	Capitalize each word in S
S.`upper()`	Change all lower case letters in S to upper case

List Methods

This is a list of list methods. In it, L is the list to which the method is applied, M is a list, x is an element to be added to, looked for, or removed from, a list, and i is an index of a list element.

Operation	Description
L.`append(`x`)`	Append element x to L
L.`count(`x`)`	Count the number of times x occurs in L
L.`extend(`M`)`	Extend L by adding the elements of M at the end
L.`index(`x`)`	Return the index of the first occurrence of x in L; `ValueError` exception if x not in L
L.`insert(`i`,`x`)`	Insert x at position i in L
L.`pop()`	Remove and return the last element of L
L.`pop(`i`)`	Remove and return the element of L at position i; `IndexError` exception if i out of range
L.`remove(`x`)`	Remove the first occurrence of x from L; `ValueError` exception if x not in L
L.`reverse()`	Reverse L in place (does *not* make a copy)
L.`sort()`	Sort x in place (does *not* make a copy)