COMP 141: List Functions

Assume L, L1, and L2 are lists; and p and q are integers.

len(L)	Returns the length of L.
L1 + L2	Returns a **new** list consisting of all the items of L1 followed immediately by all the items of L2. Note: just doing L1 + L2 doesn't change L1 or L2. You must store this new list somewhere if you want to use it later, by saying something like L = L1 + L2 (L will be a new list).
L[p]	Returns the item at index p in list L. (Indices start at zero!)
L[p:q]	Returns the sub-list consisting of all items in L starting at index p and ending one before index q. Note: if either p or q (or both) is left out, Python will assume p=0 (beginning of the list) and q=len(s) (the end of the list). Ex: L[1:] will return L with the first item left out. Using negative numbers for p and/or q counts from the end of the list: Ex: L[-1] returns the last item in L; L[-2:] returns the last two items in L.
item in L	Returns True if item occurs somewhere in list L, False otherwise. item can be a variable or a constant. Ex: 3 in [1, 2, 3] returns True Ex: num = 6 num in [1, 2, 3] returns False
L.index(item)	Similar to "item in L," but returns the lowest index at which the item is found in list L, and an error if L doesn't contain item. Think of this as looking through the list from left to right until the item is found (this left to right ordering is important if the item occurs more than once in L).
L.insert(p, item)	Inserts item into list L at position p, shifting elements to the right as necessary. **Note: this changes L, so you don't have to do L = L.insert(p, item)**
L.append(item)	Attaches Item to the end of list L. Equivalent to L = L + [item]. **Note: This changes L, so you don't have to do L = L.append(item)**
L.remove(item)	Removes the first instance of item in list L, but gives an error if there is no such item in L. **Note: This changes L, so you don't have to do L = L.remove(item)**
L.sort()	Sorts all the items in L from least to greatest. **Note: this changes L, so you don't have to do L = L.sort().** **You should not use this function solely to get the smallest or largest item in L, because sorting is a time-consuming operation.**
L.reverse()	Reverses the order of the items in L. **Note: this changes L, so you don't have to do L = L.reverse().**
max(L)	Returns the maximum value in the list L.
min(L)	Returns the minimum value in the list L.
sum(L)	Returns the total of all the values in the list L (only works with lists of numbers).
del L[p]	Removes the item at index p from the list L.

Notice that len(), min(), max(), and sum() are the only functions that are not "attached" to a list with a period.

Python String Functions

Assume s, t, and t2 are string variables, and p and q are integer variables.

Basic operations	
len(s)	Returns the length of s.
s[p]	Returns the character at index p in string s. (Indices start at zero!)
s[p:q]	Returns the substring consisting of all characters in s starting at index p and ending at index q-1, inclusive. Note: if either p or q (or both) is left out, Python will assume p=0 (beginning of the string) and q=len(s) (the end of the string). Ex: s[1:] will return s with the first character left out. Using negative numbers for p and/or q counts from the end of the string: Ex: s[-1] returns the last character in s; s[-2:] returns the last two characters in s.
s + t	Returns the string concatenation of s and t (a new string consisting of all the characters in s, followed immediately by all the characters in t. Note: just doing s + t doesn't change s or t. You must store this new string somewhere if you want to use it later, by saying something like z = s + t (z will be a new string variable).
s += t	Same as s = s + t. In other words, concatenates s with t and stores the new string back in s.
Tests	
s in t	Returns True if s occurs as a substring somewhere in t, False otherwise.
s not in t	Returns False if s occurs as a substring somewhere in t, True otherwise.
s.isalpha()	Returns True if s contains only alphabetic characters and len(s) > 0.
s.isdigit()	Returns True if s contains only numeric characters and len(s) > 0.
s.islower()	Returns True if all of the alphabetic characters in s are lowercase and len(s) > 0.
s.isupper()	Returns True if all of the alphabetic characters in s are uppercase and len(s) > 0.
s.isspace()	Returns True if all of the characters in s are spaces, tabs, or newlines, and len(s) > 0.
s.startswith(t)	Returns True if t occurs as a substring at the beginning of s.
s.endswith(t)	Returns True if t occurs as a substring at the end of s.
Searching, replacing, and stripping	
s.find(t)	Returns the lowest index in s where t is found (searches left to right). Returns -1 if t is not in s.
s.find(t, p)	Same as s.find(t), but starts searching left to right starting at index p.
s.replace(t, t2)	Returns a copy of s with all occurrences of t replaced by t2.
s.rstrip()	Returns a copy of s with all whitespace removed from the right side of the string.
Transformations	
s.lower()	Returns a copy of s with all of the alphabetic characters converted to lowercase. Any non-alphabetic character (or those that are already lowercase) are copied unchanged.
s.upper()	Same as s.lower(), but converts to uppercase.
Splitting	
s.split(t)	Divides string s into pieces based on where the separator t occurs. Usually used in a program as s1, s2, … = s.split(t) to capture the string pieces that are returned.

Notice that len() is the only function that is not "attached" to a string with a period.