

Lists

A list is a data structure that contains a series of values. Python allows the construction of a list containing values of different types. It's iterable, indexable, ordered, mutable and not hashable.

combination of lists and dictionaries

<code>[..., [...], [...], ...]</code>	list of list
<code>list_of_list[...][...]</code>	index elements in a list of list
<code>dict{key : [...], ...}</code>	values of a dictionary can be a list
<code>[dict1, dict2]</code>	list of dictionary

But the keys of dictionary are hashables, not ordered and not duplicated.

`list[:]` is a new list duplicating the original, so `list[:][0]` will return the first element of the list

Operation in the lists

<code>list + list</code>	add lists
<code>list * int</code>	multiply the list

- We cannot subtract directly a list from another by using `'-'`. `[i for i in list1 if i not in list2]`; `set(list1) - set(list2)`
- To repeat each element of lists, `[i for i in list for _ in range(int)]`
- `list_1 += list_2` equal to `list_1 = list_1 + list_2`

Examples

```
[i for i in range(10)]
[i for i in range(31) if i % 2 == 0]
[[m.upper(), len(m)] for m in msg_lst]
[seq[i:i+width] for i in range(0, len(seq), width)]
```

Common Functions

<code>list[s tart :s top :s tep]</code>	list slicing (tranche)
<code>enumerate (list)</code>	return positions and items
<code>list.index(item)</code>	return the position of the item
<code>list.count(item)</code>	returns the number of times that element appears in the list
<code>list(string)</code>	convert a string to a list one by one character
<code>string.split (sep)</code>	convert a string to a list with a separator
<code>'sep'.join(list)</code>	convert a list to a string

Common Functions (cont)

<code>len(list)</code>	length of lists
<code>max(list)</code>	find the maximum
<code>min(list)</code>	find the minimum
<code>sum(list)</code>	calculate the sum
<code>list.sort (reverse=)</code>	sort the elements of a list in-place. reverse False from smallest to largest values; alphabetic order possible
<code>sorted (list, reverse=)</code>	create a new sorted list without modifying the original list; alphabetic order possible
<code>list.reverse()</code>	reverse the elements of a list in-place
<code>reversed(list)</code>	create a new reversed list without modifying the original list
<code>list.append(item)</code>	add an element to the end of lists
<code>list.insert(item, pos)</code>	insert an element at a position of lists
<code>list.remove(item)</code>	remove an item from lists; remove only one first element.
<code>list.pop()</code>	remove and return the last element
<code>del list[pos]</code>	remove the item by its position index
<code>range(start, stop, step)</code>	similar to lists, but immutable. stop at n-1
<code>set(list)</code>	remove the duplicated elements

- `list[s tart :s top :step]` step 1 by default; stop at n-1 even if negative index
- `list[:]` create a new list. `lst2 = lst1` creates a reference to the original list with the same ID
- `'sep'.join(list)` cannot combine a list containing only number (int & float). `[str(i) for i in list]`
- `list.remove( item)` If there're duplicated elements, it remove only the first element
- `range( start, stop, step)` stop could be higher than start with a negative step