

Python Programming

List Methods

Mostafa S. Ibrahim

Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher

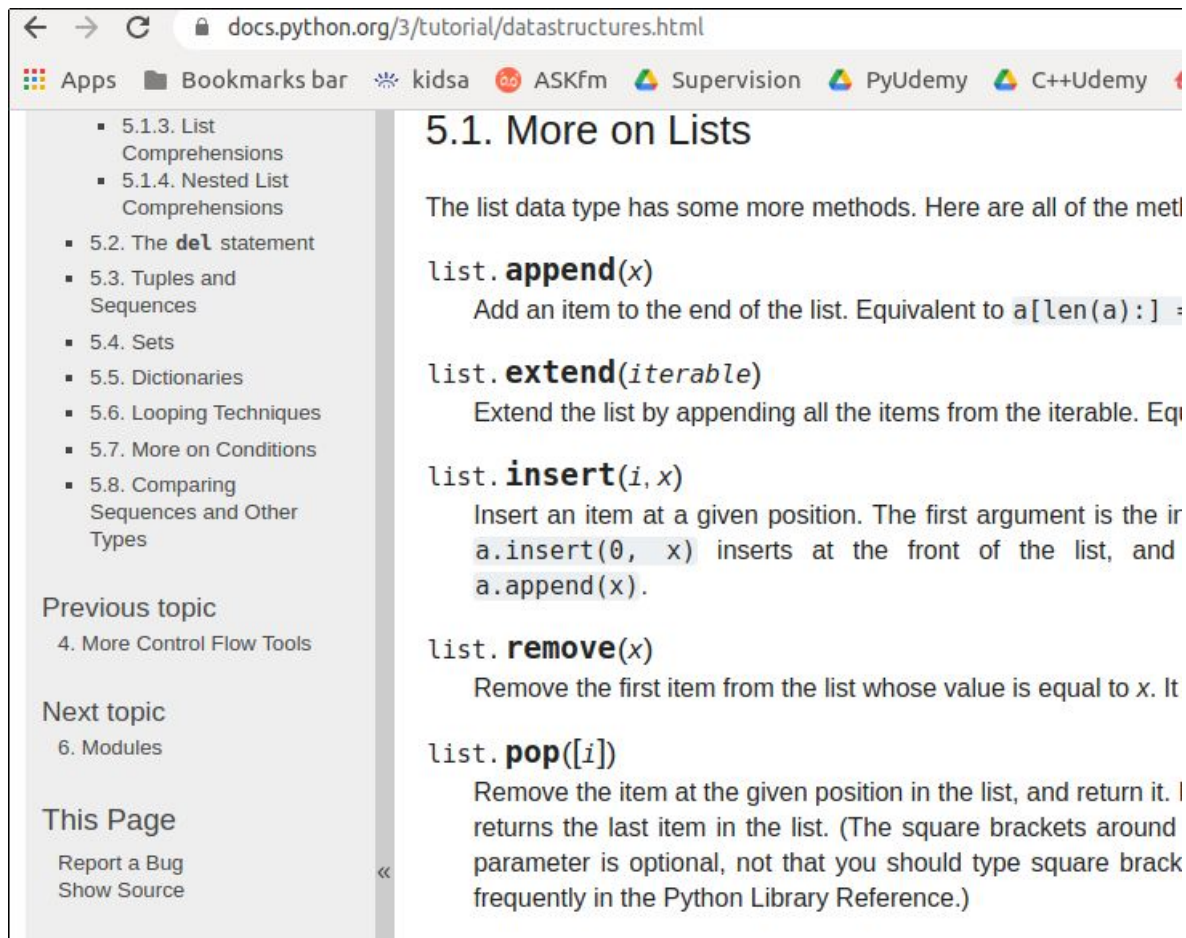
PhD from Simon Fraser University - Canada

Bachelor / Msc from Cairo University - Egypt

Ex-(Software Engineer / ICPC World Finalist)



Documentation



The screenshot shows the Python documentation website at docs.python.org/3/tutorial/datastructures.html. The left sidebar contains a table of contents with links to various list-related topics. The main content area is titled '5.1. More on Lists' and describes the list data type. It lists several methods: `list.append(x)`, `list.extend(iterable)`, `list.insert(i, x)`, `list.remove(x)`, and `list.pop([i])`, each with a brief description of its function.

docs.python.org/3/tutorial/datastructures.html

Apps Bookmarks bar kidsa ASKfm Supervision PyUdemy C++Udemy

- 5.1.3. List Comprehensions
- 5.1.4. Nested List Comprehensions
- 5.2. The **del** statement
- 5.3. Tuples and Sequences
- 5.4. Sets
- 5.5. Dictionaries
- 5.6. Looping Techniques
- 5.7. More on Conditions
- 5.8. Comparing Sequences and Other Types

Previous topic
4. More Control Flow Tools

Next topic
6. Modules

This Page
Report a Bug
Show Source

5.1. More on Lists

The list data type has some more methods. Here are all of the methods:

`list.append(x)`
Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`
Extend the list by appending all the items from the iterable. Equivalent to `a += iterable`.

`list.insert(i, x)`
Insert an item at a given position. The first argument is the index where the new item is to be inserted. `a.insert(0, x)` inserts at the front of the list, and `a.append(x)` is equivalent to `a.insert(len(a), x)`.

`list.remove(x)`
Remove the first item from the list whose value is equal to `x`. It is an error if there is no such item.

`list.pop([i])`
Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` returns the last item in the list. (The square brackets around `i` in the parameter list are optional, not that you should type square brackets frequently in the Python Library Reference.)

Browsing the list class

```
* class list(object):  
    """  
    list() -> new empty list  
    list(iterable) -> new list initialized from iterable's items  
    """  
* def append(self, p_object): # real signature unknown; restored from __doc__  
    """ L.append(object) -> None -- append object to end """  
    pass  
* def clear(self):...  
* def copy(self):...
```

```
def __init__(self, seq=()):...  
def __iter__(self, *args, **kwargs):...  
def __len__(self, *args, **kwargs):...  
def __le__(self, *args, **kwargs): # real signature unknown  
    """ Return self<=value. """  
    pass
```

Append, extend and insert

```
2 my_list = [1, 5, 10, 17, 2]
3
4 # append: add item to the end
5 my_list.append('Hii') # 1 5 10 17 2 Hii
6
7 # Extend the list by appending all the items from the iterable
8 another_lst = [3, 1]
9 my_list.extend(another_lst)
10 # 1 5 10 17 2 Hii 3 1
11
12 #TypeError: 'int' object is not iterable
13 #my_list.extend(2)
14
15 # Insert an item at a given position
16 my_list.insert(2, 'Wow')
17 # 1 5 Wow 10 17 2 Hii 3 1
18
19 for item in my_list:
20     print(item, end=' ')
21 print()
22
```

Pop, remove + del statement

```
1 my_list = [1, 5, 10, 17, 2, 'Hii']
2
3
4 # pop removes the item at a specific index and returns it.
5 print(my_list.pop()) ... # Hii ... - default last item
6 # Now list is : 1 5 10 17 2
7
8 print(my_list.pop(3)) ... # 17
9 # Now list is : 1 5 10 2
10
11 # del removes the item at a specific index:
12 del my_list[0] # 5 10 2
13
14 # remove removes the first matching value, not a specific index:
15 my_list.remove(10) # 5 2
16
17 # ValueError: list.remove(x): x not in list
18 # my_list.remove('Hei')
```

Index and clear methods

```
2
3 my_list = [1, 15, 7, 'mostafa', 7, True, 0]
4
5 # search and return the FIRST index
6 print(my_list.index(7))          # 2
7 print(my_list.index('mostafa')) # 3
8 print(my_list.index(True))      # 0 **
9 print(my_list.index(False))     # 6 **
10
11 # ValueError: 'Wow' is not in list
12 # print(my_list.index('Wow'))
13
14 my_list.clear()
15 print(len(my_list))             # 0
16
```

Count method

```
2
3 my_list = [4, 5, 7, 4, 5, 4, 8]
4
5 print(min(my_list), max(my_list)) ... # 4 8
6
7 print(my_list.count(4)) ... # 3
8 print(my_list.count([4, 5])) ... # 0 **
9
10 my_list = ['ali', 'ALI', 'ali']
11
12 print(my_list.count('ali')) ... # 'ali'
13
14
15
```

+ VS +=

- += is changing in-place
- This will imply differences behind the scene
- They only similar in binding
 - UnboundLocalError

```
1
2 lst = [1, 2, 3]
3 print(id(lst))      # 0x111
4 lst += [4]           # in-place change - internally: __iadd__
5 print(id(lst))      # 0x111
6 lst = lst + [5]      # NEW memory creation - internally: __add__
7 print(id(lst))      # 0x222
8
9 # += is behaving similar to .extend
10
11 lst += ['Hey']       # iterate on list: add 1 item
12 print(lst)          # [1, 2, 3, 4, 5, 'Hey']
13
14 lst += 'Hey'         # iterate and add 3 items
15 print(lst)          # [1, 2, 3, 4, 5, 'Hey', 'H', 'e', 'y']
16
17 #TypeError: can only concatenate list (not "str") to list
18 #lst = lst + 'Hey'
19 #lst = lst + 10
20
```


“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”