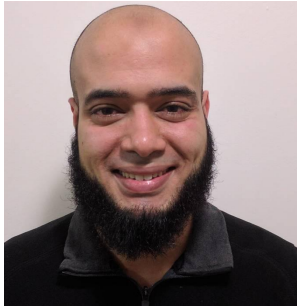
Python Programming

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

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

Apps | Bookmarks bar | kidsa | ASKfm | Supervision | PyUdemy | C++Udemy

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 met

`list.`**`append`**(*x*)

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

`list.`**`extend`**(*iterable*)

Extend the list by appending all the items from the iterable. Eq

`list.`**`insert`**(*i*, *x*)

Insert an item at a given position. The first argument is the ir `a.insert(0, x)` inserts at the front of the list, and `a.append(x)`.

`list.`**`remove`**(*x*)

Remove the first item from the list whose value is equal to *x*. It

`list.`**`pop`**([*i*])

Remove the item at the given position in the list, and return it. I returns the last item in the list. (The square brackets around parameter is optional, not that you should type square brack 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

```python
my_list = [1, 5, 10, 17, 2]

# append: add item to the end
my_list.append('Hii')    # 1 5 10 17 2 Hii

# Extend the list by appending all the items from the iterable
another_lst = [3, 1]
my_list.extend(another_lst)
# 1 5 10 17 2 Hii 3 1

#TypeError: 'int' object is not iterable
#my_list.extend(2)

# Insert an item at a given position
my_list.insert(2, 'Wow')
# 1 5 Wow 10 17 2 Hii 3 1

for item in my_list:
    print(item, end=' ')
print()
```

Pop, remove + del statement

```python
my_list = [1, 5, 10, 17, 2, 'Hii']

# pop removes the item at a specific index and returns it.
print(my_list.pop())    # Hii  - default last item
# Now list is : 1 5 10 17 2

print(my_list.pop(3))    # 17
# Now list is : 1 5 10 2

# del removes the item at a specific index:
del my_list[0]  # 5 10 2

# remove removes the first matching value, not a specific index:
my_list.remove(10)  # 5 2

# ValueError: list.remove(x): x not in list
#my_list.remove('Hei')
```

Index and clear methods

```python
my_list = [1, 15, 7, 'mostafa', 7, True, 0]

# search and return the FIRST index
print(my_list.index(7))          # 2
print(my_list.index('mostafa')) # 3
print(my_list.index(True))       # 0 **
print(my_list.index(False))      # 6 **

#ValueError: 'Wow' is not in list
#print(my_list.index('Wow'))

my_list.clear()
print(len(my_list))      # 0
```

Count method

```python
my_list = [4, 5, 7, 4, 5, 4, 8]

print(min(my_list), max(my_list))   # 4 8

print(my_list.count(4))          # 3
print(my_list.count([4, 5]))     # 0 **

my_list = ['ali', 'ALI', 'ali']

print(my_list.count('ali'))     # 'ali'
```

+ vs +=

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

```python
lst = [1, 2, 3]
print(id(lst))      # 0x111
lst += [4]          # in-place change    - internally: __iadd__
print(id(lst))      # 0x111
lst = lst + [5]     # NEW memory creation - internally: __add__
print(id(lst))      # 0x222

# += is behaving similar to .extend

lst += ['Hey']  # iterate on list: add 1 item
print(lst)      # [1, 2, 3, 4, 5, 'Hey']

lst += 'Hey'    # iterate and add 3 items
print(lst)      # [1, 2, 3, 4, 5, 'Hey', 'H', 'e', 'y']

#TypeError: can only concatenate list (not "str") to list
#lst = lst + 'Hey'
#lst = lst + 10
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."