Difference between del, remove and pop on lists

Ask Question

618

214

```
>>> a=[1,2,3]
>>> a.remove(2)
>>> a
[1, 3]
>>> a=[1,2,3]
>>> del a[1]
>>> a
[1, 3]
>>> a= [1,2,3]
>>> a.pop(1)
2
>>> a
[1, 3]
>>>
```

Is there any difference between the above three methods to remove an element from a list?

python list

edited Jul 17 '12 at 10:27
Martijn Pieters ♦
726k 143 2547 2349

asked Jul 17 '12 at 10:21
sachin irukula
3,476 4 13 18

1 Related post on similar lines for set data structure - Runtime difference between set.discard and set.remove methods in Python? – RBT Aug 1 '18 at 2:51

10 Answers

872

Yes, `remove` removes the *first* matching *value*, not a specific index:

```
>>> a = [0, 2, 3, 2]
>>> a.remove(2)
>>> a
[0, 3, 2]
```

`del` removes the item at a specific

```
>>> a
[3, 2, 1]
```

and `pop` removes the item at a specific index and returns it.

```
>>> a = [4, 3, 5]
>>> a.pop(1)
3
>>> a
[4, 5]
```

Their error modes are different too:

```
>>> a = [4, 5, 6]
>>> a.remove(7)
Traceback (most recent call last)
  File "<stdin>", line 1, in <modu
ValueError: list.remove(x): x not
>>> del a[7]
Traceback (most recent call last)
  File "<stdin>", line 1, in <modu
IndexError: list assignment index
>>> a.pop(7)
Traceback (most recent call last)
  File "<stdin>", line 1, in <modu
IndexError: pop index out of range
```

edited Aug 24 '18 at 13:38
glibdud
5,699 2 17 31

answered Jul 17 '12 at 10:24
Martijn Pieters ♦
726k 143 2547
2349

I thought `del` was a python 2 syntax holdover like `print` , but it still works in python 3. – jxramos Sep 1 '17 at 20:12

12 @jxramos: `del` is not a syntax holdover, no. The syntax is unchanged, just like `return` or `if` or `while` . – Martijn Pieters ♦ Sep 1 '17 at 20:57

It is worth mentioning that users should be careful when iterating over a list and using these functions on it

at the same time as they are iterating. – hamaney Mar 29 at 23:12

Use `del` to remove an element by index, `pop()` to remove it by index if the list, and raises `ValueError` if no such value occurs in the list.

When deleting index `i` from a list of `n` elements, the computational complexities of these methods are

```
del     O(n - i)
pop     O(n - i)
remove  O(n)
```

edited Jul 17 '12 at 10:34

answered Jul 17 '12 at 10:25
Sven Marnach
360k 80 758 702

1 Does pop require searching the list – sachin irukula Jul 17 '12 at 10:31

@kratos: No, see my edit. – Sven Marnach Jul 17 '12 at 10:34

9 +1 for complexity breakdown. Illustrates how delete and pop are constant when the element is at the end of the list. – Big Sharpie Nov 6 '14 at 5:56

Remember guys... anything index based is one shot O(n-1)... if you have to do a lookup (by value), it will traverse the collection until the element is found. – Pepito Fernandez Oct 14 '17 at 14:13

@PepitoFernandez Look-ups by index in a list are O(1) in Python. (A list in Python is similar to a vector in C++.) – Sven Marnach Oct 14 '17 at 19:28

60

Since no-one else has mentioned it, note that `del` (unlike `pop`) allows the removal of a range of indexes because of list slicing:

```
>>> lst = [3, 2, 2, 1]
>>> del lst[1:]
>>> lst
[3]
```

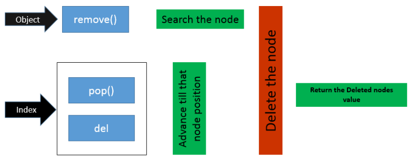
This also allows avoidance of an `IndexError` if the index is not in the list:

```
>>> lst = [3, 2, 2, 1]
>>> del lst[10:]
>>> lst
[3, 2, 2, 1]
```

answered Jan 16 '17 at 10:27
Chris_Rands
17.3k 5 41 77

38

Already answered quite well by others. This one from my end :)


Object
remove()
Search the node
Index
pop()
del
Advance till that node position
Delete the node
Return the Deleted nodes value

Evidently, `pop` is the only one which returns the value, and `remove` is the only one which searches the object, while `del` limits itself to a simple deletion.

edited Jun 28 '18 at 7:00

answered Aug 8 '17 at 15:33
Saurav Sahu
6,657 3 25 47

Thnx! One note: in python, due to the way the lists are implemented(there practically arrays...!), "Advance till that node position" is O(1) – ntg Nov 28 '18 at 11:14

Great visual representation! – Granny Aching Mar 14 at 17:12

13

pop - Takes Index and returns Value

remove - Takes value, removes first occurrence, and returns nothing

delete - Takes index, removes value at that index, and returns nothing

edited Apr 10 '18 at 12:30
Wool
142 9

answered Feb 7 '15 at 6:51
Bahubali Patil
460 1 7 17

6

Many best explanations are here but I will try my best to simplify more.

remove(): It used to remove first occurrence of element

`remove(i)` => first occurrence of i value

```
>>> a = [0, 2, 3, 2, 1, 4, 6, 5, 7
>>> a.remove(2)   # where i = 2
>>> a
[0, 3, 2, 1, 4, 6, 5, 7]
```

pop(): It used to remove element if:

unspecified

`pop()` => from end of list

```
>>>a.pop()
>>>a
[0, 3, 2, 1, 4, 6, 5]
```

specified

`pop(index)` => of index

```
>>>a.pop(2)
>>>a
[0, 3, 1, 4, 6, 5]
```

WARNING: Dangerous Method Ahead

delete(): Its a prefix method.

Keep an eye on two different syntax for same method: [] and (). It possesses power to:

1.Delete index

`del a[index]` => used to delete index and its associated value just like pop.

```
>>>del a[1]
>>>a
[0, 1, 4, 6, 5]
```

2.Delete values in range [index 1:index N]

`del a[0:3]` => multiple values in range

```
>>>del a[0:3]
>>>a
[6, 5]
```

3.Last but not list, to delete whole list in one shot

`del (a)` => as said above.

Hope this clarifies the confusion if any.

edited Aug 26 '18 at 16:05

answered Aug 26 '18 at 15:34

Mayur Patil
587 1 6 17

1

Any operation/function on different data structures is defined for particular actions. Here in your case i.e. removing an element, delete, Pop and remove. (If you consider sets, Add another operation - discard) Other confusing case is while adding. Insert/Append. For Demonstration, Let us Implement deque. deque is a hybrid linear data structure, where you can add elements / remove elements from both ends.(Rear and front Ends)

```
 class Deque(object):

   def __init__(self):

     self.items=[]

   def addFront(self,item):

     return self.items.insert(0,ite
   def addRear(self,item):

     return self.items.append(item)
   def deleteFront(self):

     return self.items.pop(0)
   def deleteRear(self):
     return self.items.pop()
   def returnAll(self):

     return self.items[:]
```

In here, see the operations:

```
 def deleteFront(self):

     return self.items.pop(0)
 def deleteRear(self):
     return self.items.pop()
```

Operations have to return something. So, pop - With and without an index. If I don't want to return the value: del self.items[0]

Delete by value not Index:

- remove :

```
            list_ez.remove(i)
    print list_ez
```

Returns [1,3,5,7]

let us consider the case of sets.

```
set_ez=set_ez=set(range(10))

set_ez.remove(11)

# Gives Key Value Error.
##KeyError: 11

set_ez.discard(11)

# Does Not return any errors.
```

answered Jul 10 '17 at 17:29

phanindravarma
762 4 8

0

While pop and delete both take indices to remove an element as stated in above comments. A key difference is the time complexity for them. The time complexity for pop() with no index is O(1) but is not the same case for deletion of last element.

If your use case is always to delete the last element, it's always preferable to use pop() over delete(). For more explanation on time complexities, you can refer to https://www.ics.uci.edu/~pattis/ICS-33/lectures/complexitypython.txt

answered Sep 4 '16 at 20:22

skashyap
102 9

1 This is wrong in multiple ways. There is no such method as `delete`. The differences are that `pop` returns the value, and that `del` works on slices. In cases where `pop` works, `del` has exactly the same computational complexity (and is slightly faster by a constant term). – abarnert Mar 31 '18 at 9:57

0

The [remove](#) operation on a list is given a value to remove. It searches the list to find an item with that value and deletes the first matching item it

```
>>> x = [1, 0, 0, 0, 3, 4, 5]
>>> x.remove(4)
>>> x
[1, 0, 0, 0, 3, 5]
>>> del x[7]
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <
    del x[7]
IndexError: list assignment index
```

The del statement can be used to delete an entire list. If you have a specific list item as your argument to del (e.g. listname[7] to specifically reference the 8th item in the list), it'll just delete that item. It is even possible to delete a "slice" from a list. It is an error if there index out of range, raises a IndexError.

```
>>> x = [1, 2, 3, 4]
>>> del x[3]
>>> x
[1, 2, 3]
>>> del x[4]
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <
    del x[4]
IndexError: list assignment index
```

The usual use of pop is to delete the last item from a list as you use the list as a stack. Unlike del, pop returns the value that it popped off the list. You can optionally give an index value to pop and pop from other than the end of the list (e.g listname.pop(0) will delete the first item from the list and return that first item as its result). You can use this to make the list behave like a queue, but there are library routines available that can provide queue operations with better performance than pop(0) does. It is an error if there index out of range, raises a IndexError.

```
>>> x = [1, 2, 3]
>>> x.pop(2)
3
>>> x
[1, 2]
>>> x.pop(4)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <
    x.pop(4)
IndexError: pop index out of range
```

See collections.deque for more details.

answered Aug 9 '18 at 17:13

Kushan Gunasekera
71 5

-3

You can also use remove to remove a value by index as well.

```
n = [1, 3, 5]
```

```
n.remove(n[1])
```

n would then refer to [1, 5]

edited Oct 29 '15 at 13:48
Mathias Müller
17k 12 44 55

answered Oct 16 '12 at 0:41
max runia
35 1

38 Try `n = [5, 3, 5]` , then `n.remove(n[2])` . – abarnert Feb 18 '14 at 3:46

@abarnert your use case works in sync with the below case n = [5,3,5] , then n.remove(5). Both of these remove the first encountered element from the list. – Akhil Ghatiki Mar 31 '18 at 7:24

@AkhilGhatiki `n.remove(n[2])` removes `n[0]` , not `n[2]` . So it's not just linear time for no reason (maybe not a big deal when N=3), it's also wrong (a big deal no matter what N is) – abarnert Mar 31 '18 at 9:14

@abarnert you are right !!! I overlooked it. Thanks !! – Akhil Ghatiki Mar 31 '18 at 9:21

protected by Shai May 23 '16 at 6:45

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?