Lists in Python

Andres Ramirez Nov 20 · 5 min read

Lists in Python are very handy in maintaining elements that may or may not be similar in a particular order that you assign them. You can create lists of grocery store locations, items being sold online, or even favorite fruits! Luckily there are simple methods that we can employ to create and modify lists to our liking. We will cover a number of them below. We will be using a list of fruits to use as an example.

```
fruits = [‘apples’, ‘bananas’, ‘blueberries’, ‘cherries’, ‘dates’, ‘figs’, ‘grapes’,’kiwis’]

print(fruits)

Output: ['apples', 'bananas', 'blueberries', 'cherries', 'dates', 'figs', 'grapes', 'kiwis']
```

When we print our list of fruits, we are sent a list that looks similar to the input we currently have. In order to retrieve items from our list individually, we let Python know exactly what element we are looking for in our list by stating the position in which our element we are seeking is at. If we look at our list we have a total of eight items.

We can do a quick check by using len()

```
print(len(fruits))

Output: 8
```

Checks out!

It is important to remember that Python, just like other programming languages, the first element in a list is in position 0 and NOT position 1. So for our example, the elements would be apples in position 0, bananas in 1, blueberries in 2 and so on. With this in mind lets ask for elements in positions 5 and 7.

```
print(fruits[5])
print(fruits[7])

Output:
figs
kiwis
```

If we count out the values we can see that it all checks out. Another quick tip, if you want to see the final element in a list, which is helpful when you have a large list you can use negative numbers. For example:

```
print(fruits[-1])

Output: kiwis
```

This also works for the second to last and so forth.

```
print(fruits[-2])
print(fruits[-3])

Output:
kiwis
```

```
grapes
```

From here you can use individual elements as you'd like. For example, we can call them into a message.

```
print("My favorite fruits are " + fruits[2] + " and " + fruits[5] +
"."
)

Output:
My favorite fruits are blueberries and figs.
```

Manipulating elements in a list

There are times where we may want to add, remove or alter the elements in our list to fit our needs. We will continue using our list of fruits to see how we can change our elements based on our needs.

```
print(fruits)

Output:

['apples', 'bananas', 'blueberries', 'cherries', 'dates', 'figs',
'grapes', 'kiwis']
```

We can change any element in a list to another value by calling the specific element in our list and telling Python to assign it with the value that we would like instead. In the example below, we will be changing the element in fruits[3], cherries, to cantaloupe.

```
fruits[3] = 'cantaloupe'

print(fruits)
```

```
Output:
['apples', 'bananas', 'blueberries', 'cantaloupe', 'dates', 'figs',
'grapes', 'kiwis']
```

What if we forgot to add a fruit to our list? Python makes it very easy to add a new element to the end of a list using .append(). Remember, .append() will add a new element at the ***end*** of the list.

```
fruits.append(‘mango’)

print(fruits)

Output:
['apples', 'bananas', 'blueberries', 'cantaloupe', 'dates', 'figs',
'grapes', 'kiwis', 'mango']
```

We can also use .append() to create lists as we go, which in some situations can be super helpful! What if we were asked to catalog the most liked berries. Below, we will create the list berries and tell Python it is an empty list. Then we go ahead and add different types of berries to this list as shown below. At the end of our code we will end with a list that includes blueberries, blackberries, raspberries, and strawberries.

```
berries = []

berries.append(‘blueberries’)
berries.append(‘blackberries’)
berries.append(‘raspberries’)
berries.append(‘strawberries’)

print(berries)

Output:
['blueberries', 'blackberries', 'raspberries', 'strawberries']
```

We can also replicate our previous fruits list from our example using .append(), just like our berries list above.

```
fruits = []

fruits.append('apples')
fruits.append('blueberries')
fruits.append('cantaloupe')
fruits.append('dates')
fruits.append('figs')
fruits.append('grapes')
fruits.append('kiwis')
fruits.append('mango')

print(fruits)

Output:
['apples', 'blueberries', 'cantaloupe', 'dates', 'figs', 'grapes',
'kiwis', 'mango']
```

Now, maybe we want to put cherries back on our list, which we had previously changed to cantaloupe. At the same time we want to keep our list in alphabetical order; how do we go about this? We can easily insert cherries back in the list by using .insert(). Since we want to place cherries as the fourth element in our list we will use the following code:

```
fruits.insert(3,’cherries’)

print(fruits)

Output:
['apples', 'blueberries', 'cantaloupe', 'cherries', 'dates', 'figs',
'grapes', 'kiwis', 'mango']
```

Adding and changing elements is great, but what if we wanted to remove elements from the list? Maybe figs fell out of favor of being one of the most liked fruits. There are three different methods we can use to remove elements from a list, one that accomplishes completely and one that allows the removed element to be used in the future.

Let's say for example that upon creating our list of fruits that cantaloupe should not have been added. We can use the ***del*** method to remove the element, while calling the position that cantaloupe is in.

```
del fruits[2]
print(fruits)

Output:
['apples', 'blueberries', 'cherries', 'dates', 'figs', 'grapes',
'kiwis', 'mango']
```

At this point the element cantaloupe is no longer available for our use.

What if we wanted to use an element from our list of fruits after we have removed it? We can use the second method for element removal, .pop() to achieve this. It is important to keep in mind that when you use the .pop() method, you are removing the last element in a list. If you wanted to remove a specific element, you can specify its location in the brackets after .pop.

Example

```
print(fruits)

Output:
['apples', 'blueberries', 'cherries', 'dates', 'figs', 'grapes',
'kiwis', 'mango']

popped_fruit = fruits.pop()
print(fruits)
print(popped_fruit)

Output:
['apples', 'blueberries', 'cherries', 'dates', 'figs', 'grapes',
'kiwis']

mango
```

As we can see, mango was the final element on our list, which was removed.

In the following bit, we will be removing the fourth element of the list, which is cherries, to demonstrate how we can specify which position we wish to remove.

```
fruits = [‘apples’, ‘bananas’, ‘blueberries’, ‘cherries’, ‘dates’, ‘figs’, ‘grapes’,’kiwis’, ‘mango’]
print(fruits)

Output:
['apples', 'bananas', 'blueberries', 'cherries', 'dates', 'figs', 'grapes', 'kiwis', 'mango']

popped_fruit = fruits.pop(3)
print(fruits)
print(popped_fruit)

Output:
['apples', 'bananas', 'blueberries', 'dates', 'figs', 'grapes', 'kiwis', 'mango']

cherries
```

The third method we have is helpful when we are working with a large list and have no idea what the position number is. Say that we wanted to remove grapes from our list of fruits but had no idea its position. We can use the .remove() method by defining the element that we want to remove in the brackets. Let’s remove grapes from our list.

```
fruits = [‘apples’, ‘bananas’, ‘blueberries’, ‘cherries’, ‘dates’, ‘figs’, ‘grapes’,’kiwis’, ‘mango’]
print(fruits)

Output:
['apples', 'bananas', 'blueberries', 'cherries', 'dates', 'figs', 'grapes', 'kiwis', 'mango']

fruits.remove(‘grapes’)
print(fruits)

Output:
['apples', 'bananas', 'blueberries', 'cherries', 'dates', 'figs', 'kiwis', 'mango']
```

*As a special note you can still work with the removed element by assigning it to another list.

That concludes some of the fun and entertaining things you can do when creating lists with elements. There are unlimited ways that one can use these methods with their data so have fun and give them a try!

Lists Python Tutorial