

# Python Nested List

A [list](#) can contain any sort object, even another list (sublist), which in turn can contain sublists themselves, and so on. This is known as nested list.

You can use them to arrange data into hierarchical structures.

## Create a Nested List

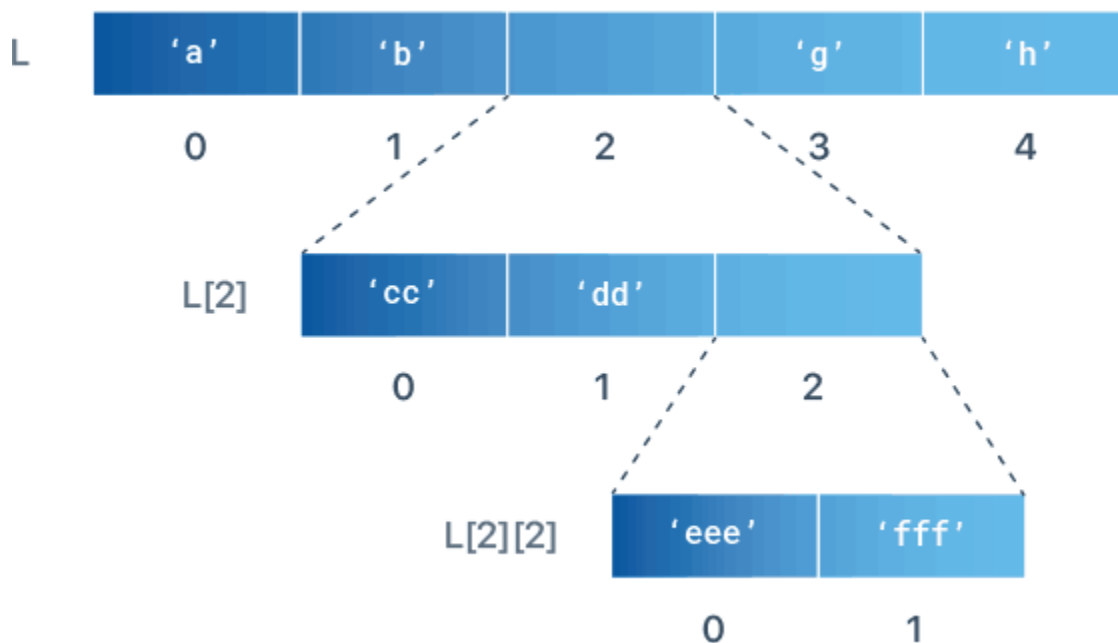
A nested list is created by placing a comma-separated sequence of sublists.

```
L = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', 'h']
```

## Access Nested List Items by Index

You can access individual items in a nested list using multiple indexes.

The indexes for the items in a nested list are illustrated as below:



```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']
```

```
print(L[2])  
# Prints ['cc', 'dd', ['eee', 'fff']]
```

```
print(L[2][2])  
# Prints ['eee', 'fff']
```

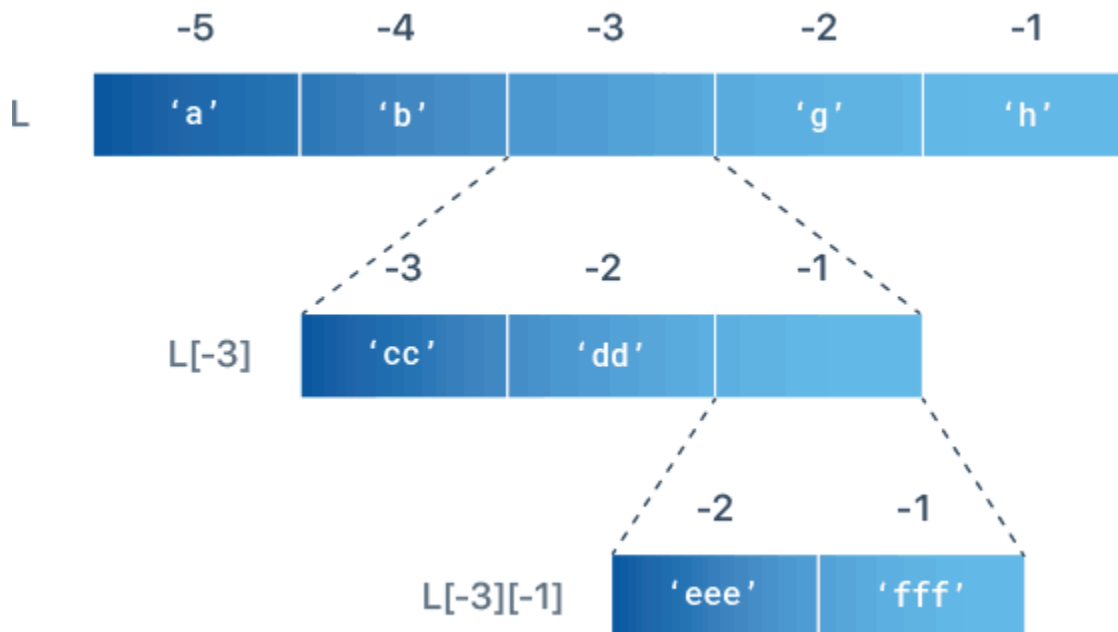
```
print(L[2][2][0])  
# Prints eee
```

## Negative List Indexing In a Nested List

You can access a nested list by negative indexing as well.

Negative indexes count backward from the end of the list. So, `L[-1]` refers to the last item, `L[-2]` is the second-last, and so on.

The negative indexes for the items in a nested list are illustrated as below:



```
L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']
```

```
print(L[-3])  
# Prints ['cc', 'dd', ['eee', 'fff']]
```

```
print(L[-3][-1])  
# Prints ['eee', 'fff']
```

```
print(L[-3][-1][-2])  
# Prints eee
```

## Change Nested List Item Value

You can change the value of a specific item in a nested list by referring to its index number.

```
L = ['a', ['bb', 'cc'], 'd']  
L[1][1] = 0  
print(L)  
# Prints ['a', ['bb', 0], 'd']
```

## Add items to a Nested list

To add new values to the end of the nested list, use [append\(\)](#) method.

```
L = ['a', ['bb', 'cc'], 'd']  
L[1].append('xx')  
print(L)  
# Prints ['a', ['bb', 'cc', 'xx'], 'd']
```

When you want to insert an item at a specific position in a nested list, use [insert\(\)](#) method.

```
L = ['a', ['bb', 'cc'], 'd']  
L[1].insert(0, 'xx')  
print(L)  
# Prints ['a', ['xx', 'bb', 'cc'], 'd']
```

You can merge one list into another by using [extend\(\)](#) method.

```
L = ['a', ['bb', 'cc'], 'd']
L[1].extend([1,2,3])
print(L)
# Prints ['a', ['bb', 'cc', 1, 2, 3], 'd']
```

## Remove items from a Nested List

If you know the index of the item you want, you can use [pop\(\)](#) method. It modifies the list and returns the removed item.

```
L = ['a', ['bb', 'cc', 'dd'], 'e']
x = L[1].pop(1)
print(L)
# Prints ['a', ['bb', 'dd'], 'e']

# removed item
print(x)
# Prints cc
```

If you don't need the removed value, use the del statement.

```
L = ['a', ['bb', 'cc', 'dd'], 'e']
del L[1][1]
print(L)
# Prints ['a', ['bb', 'dd'], 'e']
```

If you're not sure where the item is in the list, use [remove\(\)](#) method to delete it by value.

```
L = ['a', ['bb', 'cc', 'dd'], 'e']
L[1].remove('cc')
print(L)
# Prints ['a', ['bb', 'dd'], 'e']
```

# Find Nested List Length

You can use the built-in [len\(\)](#) function to find how many items a nested sublist has.

```
L = ['a', ['bb', 'cc'], 'd']
```

```
print(len(L))
```

```
# Prints 3
```

```
print(len(L[1]))
```

```
# Prints 2
```

# Iterate through a Nested List

To iterate over the items of a nested list, use simple [for loop](#).

```
L = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
```

```
for list in L:
```

```
    for number in list:
```

```
        print(number, end=' ')
```

```
# Prints 1 2 3 4 5 6 7 8 9
```