

# لیست ها

در بحث رشته‌ها، مفهوم دنباله را در پایتون معرفی کردیم. لیست‌ها می‌توانند به عنوان نسخه‌ی عمومی‌ترین دنباله در پایتون در نظر گرفته شوند. برخلاف رشته‌ها، آنها قابل تغییر هستند، به این معنا که عناصر درون یک لیست می‌توانند تغییر کنند!

در این بخش، موارد زیر را خواهیم آموخت:

1. ایجاد لیست‌ها
2. ایندکس‌گذاری و برش لیست‌ها
3. متدهای پایه‌ای لیست
4. تو در تو کردن لیست‌ها
5. مقدمه‌ای بر فهم لیست‌ها

لیست‌ها با استفاده از براکت‌ها [] و کاماها بین هر عنصر در لیست ساخته می‌شوند.

بیایید شروع به ساختن لیست‌ها کنیم!

In [3]:

```
1 # Assign a list to an variable named my_list
2 my_list = [1,2,3]
```

ما تازه یک لیست از اعداد صحیح ایجاد کردیم، اما در واقع لیست‌ها می‌توانند شامل انواع مختلفی از اشیاء باشند. به عنوان مثال:

In [4]:

```
1 my_list = ['A string', 23, 100, 3.14, [55, 66]]
```

مانند رشته‌ها، تابع len() تعداد عناصر موجود در دنباله لیست را به شما می‌گوید.

In [5]:

```
1 len(my_list)
```

Out[5]:

5

In [ ]:

```
1
```

## ایندکس گذاری و برش

عمل ایندکس‌گذاری و برش مشابه با رشته‌ها عمل می‌کند. بیایید یک لیست جدید بسازیم تا به یادآوری نحوه کار این عملیات بپردازیم.

=====

-6

-5

-4

-3

-2

-1

'foo'

'bar'

'baz'

'qux'

'quux'

'corge'

0

1

2

3

4

5

In [6]:

```
1 my_list = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']
```

In [7]:

```
1 # Grab element at index 0
2 my_list[0]
```

Out[7]:

'foo'

In [8]:

```
1 # Grab index 1 and everything past it
2 my_list[1:]
```

Out[8]:

['bar', 'baz', 'qux', 'quux', 'corge']

In [9]:

```
1 # Grab everything UP TO index 3
2 my_list[:3]
```

Out[9]:

['foo', 'bar', 'baz']

In [10]:

```
1 my_list[::-1]
```

Out[10]:

['corge', 'quux', 'qux', 'baz', 'bar', 'foo']

بررسی عضویت یا عدم عضویت در لیست با عملگر in

In [11]:

```
1 'qux' in my_list
```

Out[11]:

True

In [12]:

```
1 'bob' in my_list
```

Out[12]:

False

In [13]:

```
1 'bob' not in my_list
```

Out[13]:

True

لیست قابل جهش هستند

In [14]:

```
1 my_list
```

Out[14]:

```
['foo', 'bar', 'baz', 'qux', 'quux', 'corge']
```

In [15]:

```
1 my_list[0] = 35
```

In [16]:

```
1 my_list
```

Out[16]:

```
[35, 'bar', 'baz', 'qux', 'quux', 'corge']
```

In [17]:

```
1 my_list[-1] = 66
```

In [18]:

```
1 my_list
```

Out[18]:

```
[35, 'bar', 'baz', 'qux', 'quux', 66]
```

In [19]:

```
1 l = my_list
```

In [20]:

```
1 id(my_list)
```

Out[20]:

2656956466304

In [21]:

```
1 id(1)
```

Out[21]:

2656956466304

In [22]:

```
1 l[0] = -555
```

In [23]:

```
1 l
```

Out[23]:

[-555, 'bar', 'baz', 'qux', 'quux', 66]

In [24]:

```
1 my_list
```

Out[24]:

[-555, 'bar', 'baz', 'qux', 'quux', 66]

## لیست ها در پایتون ترتیب دارند

In [25]:

```
1 l = [1, 2, 3]
2 l1 = l
3
4 l2 = [3, 2, 1]
```

In [26]:

```
1 l is l1
```

Out[26]:

True

In [27]:

```
1 1 is 12
```

Out[27]:

False

In [28]:

```
1 1 == 11
```

Out[28]:

True

همچنین می‌توانیم از + برای اتصال لیست‌ها استفاده کنیم، درست مانند کاری که برای رشته‌ها انجام دادیم.

In [29]:

```
1 my_list + ['new item']
```

Out[29]:

```
[-555, 'bar', 'baz', 'qux', 'quux', 66, 'new item']
```

توجه: این در واقع لیست اصلی را تغییر نمی‌دهد!

In [30]:

```
1 my_list
```

Out[30]:

```
[-555, 'bar', 'baz', 'qux', 'quux', 66]
```

برای دائمی شدن این تغییر، باید لیست جدید را مجدد در متغیر ذخیره نمایید!

In [31]:

```
1 # Reassign
2 my_list = my_list + ['add new item permanently']
```

In [32]:

```
1 my_list
```

Out[32]:

```
[-555, 'bar', 'baz', 'qux', 'quux', 66, 'add new item permanently']
```

In [33]:

```
1 # list + number
2
3 my_list + 5
```

**TypeError**

Traceback (most recent call last)

t)

Cell In[33], line 3

```
1 # list + number
```

```
----> 3 my_list + 5
```

**TypeError:** can only concatenate list (not "int") to list

In [34]:

```
1 # list + string
2 my_list + 'string'
```

**TypeError**

Traceback (most recent call last)

t)

Cell In[34], line 2

```
1 # list + string
```

```
----> 2 my_list + 'string'
```

**TypeError:** can only concatenate list (not "str") to list

همچنین می‌توانیم از \* برای تکرار یک لیست، مشابه رشته‌ها استفاده کنیم:

In [35]:

```
1 # Make the list double
2 my_list * 2
```

Out[35]:

```
[-555,
'bar',
'baz',
'qux',
'quux',
66,
'add new item permanently',
-555,
'bar',
'baz',
'qux',
'quux',
66,
'add new item permanently']
```

In [36]:

```
1 # Again doubling not permanent
2 my_list
```

Out[36]:

```
[-555, 'bar', 'baz', 'qux', 'quux', 66, 'add new item permanently']
```

## متدهای پایه لیست ها

اگر با یک زبان برنامه‌نویسی دیگر آشنایی دارید، ممکن است شروع کنید به تشبیه بین آرایه‌ها در آن زبان و لیست‌ها در پایتون. اما لیست‌ها در پایتون به دو دلیل از آرایه‌ها در زبان‌های دیگر انعطاف‌پذیرتر هستند: آنها اندازه ثابتی ندارند (بدین معنی که نیازی نداریم اندازه یک لیست را مشخص کنیم) و محدودیت نوع ثابتی ندارند (همانطور که در بالا دیدیم).

بیا بید به بررسی تعدادی از متدهای ویژه برای لیست‌ها بپردازیم:

In [37]:

```
1 # Create a new list
2 list1 = [1,2,3]
```

متد `append` یک و تنها یک عضو به انتهای لیست اضافه می‌کند:

In [38]:

```
1 # Append
2 list1.append(-100)
```

In [39]:

```
1 # Show
2 list1
```

Out[39]:

```
[1, 2, 3, -100]
```

اگر بخواهیم چند عضو را بصورت همزمان به لیست اضافه کنیم دیگر متد `append` پاسخگوی نیاز ما نمی باشد

In [41]:

```
1 list1.append([55, 66])
```

In [42]:

```
1 list1
```

Out[42]:

```
[1, 2, 3, -100, [55, 66]]
```

در این حالت باید از متد `extend` استفاده نماییم

In [43]:

```
1 # add multiple items to end of list
2 list1.extend([555, 666, 777])
```

In [44]:

```
1 list1
```

Out[44]:

```
[1, 2, 3, -100, [55, 66], 555, 666, 777]
```

درج عضو جدید در ایندکس مورد نظر با استفاده از متد `insert`

In [45]:

```
1 # insert an item to specific index
2 list1.insert(1, 'quux')
3
```

In [46]:

```
1 list1
```

Out[46]:

```
[1, 'quux', 2, 3, -100, [55, 66], 555, 666, 777]
```

افزودن چندین عضو به ابتدا یا انتها با استفاده از عملگر `+=`

In [47]:

```
1 a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']
2
3 a += ['bbb', 'ccc']
```

In [48]:

```
1 a
```

Out[48]:

```
['foo', 'bar', 'baz', 'qux', 'quux', 'corge', 'bbb', 'ccc']
```

In [49]:

```
1 a = [200, 100] + a
2
3 a
```

Out[49]:

```
[200, 100, 'foo', 'bar', 'baz', 'qux', 'quux', 'corge', 'bbb', 'ccc']
```

یک نکته جالب



In [51]:

```
1 a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']
2
3 a += 'string'
```

In [52]:

```
1 a
```

Out[52]:

```
['foo', 'bar', 'baz', 'qux', 'quux', 'corge', 's', 't', 'r', 'i', 'n',
'g']
```

متد pop برای حذف و بازگرداندن یک عضو از لیست مورد استفاده قرار می گیرد. بصورت پیش فرض این متد آخرین ایندکس یا 1- را حذف کرده و مقدار آن را برای استفاده در سایر محاسبات برمیگرداند

In [53]:

```
1 list1 = [1,2,3,4,5]
```

In [54]:

```
1 # Pop off the 0 indexed item
2 list1.pop(0)
```

Out[54]:

```
1
```

In [55]:

```
1 # Show
2 list1
```

Out[55]:

```
[2, 3, 4, 5]
```

In [56]:

```
1 list1.pop(0)
```

Out[56]:

```
2
```

In [57]:

```
1 list1
```

Out[57]:

```
[3, 4, 5]
```

In [58]:

```
1 # Assign the popped element, remember default popped index is -1
2 popped_item = list1.pop()
```

In [59]:

```
1 popped_item
```

Out[59]:

5

In [60]:

```
1 # Show remaining list
2 list1
```

Out[60]:

[3, 4]

حذف اعضا از لیست به چندین روش مقدور می باشد.

1. استفاده از متد `remove`

2. استفاده از تابع `del`

3. استفاده از روش انتساب چند ایندکسی

In [61]:

```
1 # remove a first occurrence of value
2 list1 = [1,2,3,4,5,6,1,1]
3
4 list1.remove(1)
```

In [62]:

```
1 list1
```

Out[62]:

[2, 3, 4, 5, 6, 1, 1]

In [63]:

```
1 del list1[5]
```

In [64]:

```
1 list1
```

Out[64]:

[2, 3, 4, 5, 6, 1]

In [65]:

```
1 del list1[0:2]
```

In [66]:

```
1 list1
```

Out[66]:

```
[4, 5, 6, 1]
```

همچنین باید توجه داشت که اگر در ایندکس مورد نظر عنصری وجود نداشته باشد، عمل ایندکس‌گذاری در لیست خطا خواهد داد. به عنوان مثال:

In [67]:

```
1 list1[100]
```

**IndexError**

Traceback (most recent call last)

t)

Cell In[67], line 1

```
----> 1 list1[100]
```

**IndexError:** list index out of range

می‌توانیم از متد `sort` و متد `reverse` نیز برای تغییر لیست‌های شما استفاده کنیم:

In [68]:

```
1 new_list = ['a','e','x','b','c']
```

In [69]:

```
1 #Show
2 new_list
```

Out[69]:

```
['a', 'e', 'x', 'b', 'c']
```

In [70]:

```
1 new_list[::-1]
```

Out[70]:

```
['c', 'b', 'x', 'e', 'a']
```

In [71]:

```
1 # Use reverse to reverse order (this is permanent!)
2 new_list.reverse()
```

In [72]:

```
1 new_list
```

Out[72]:

```
['c', 'b', 'x', 'e', 'a']
```

In [73]:

```
1 # Use sort to sort the list (in this case alphabetical order, but for numbers it wil
2 new_list.sort()
3 new_list
```

Out[73]:

```
['a', 'b', 'c', 'e', 'x']
```

In [74]:

```
1 new_list.sort(reverse=True)
2 new_list
```

Out[74]:

```
['x', 'e', 'c', 'b', 'a']
```

In [75]:

```
1 l = new_list.copy()
2
3 l is new_list
```

Out[75]:

```
False
```

In [76]:

```
1 l
```

Out[76]:

```
['x', 'e', 'c', 'b', 'a']
```

## تغییر چندین مقدار در لیست با عملگر انتساب

In [77]:

```
1 a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']
```

In [78]:

```
1 a[1:4]
```

Out[78]:

```
['bar', 'baz', 'qux']
```

1. تعداد موارد انتخاب شده با موارد جایگزین برابر باشند

In [79]:

```
1 a[1:4] = [11, 22, 33]
```

In [80]:

```
1 a
```

Out[80]:

```
['foo', 11, 22, 33, 'quux', 'corge']
```

2. تعداد موارد انتخاب شده کمتر از موارد جایگزین باشد

In [81]:

```
1 a[0:2] = [-100, -200, -300, -400]
2
3 a
```

Out[81]:

```
[-100, -200, -300, -400, 22, 33, 'quux', 'corge']
```

In [83]:

```
1 a[2:2] = ['foo', 'bar', 'baz']
2
3 a
```

Out[83]:

```
[-100, -200, 'foo', 'bar', 'baz', -300, -400, 22, 33, 'quux', 'corge']
```

3. تعداد موارد انتخاب شده بیشتر از موارد جایگزین باشد

In [84]:

```
1 a[0:6] = ['hello']
2
3 a
```

Out[84]:

```
['hello', -400, 22, 33, 'quux', 'corge']
```

In [85]:

```
1 a[1:3] = []
2
3 a
```

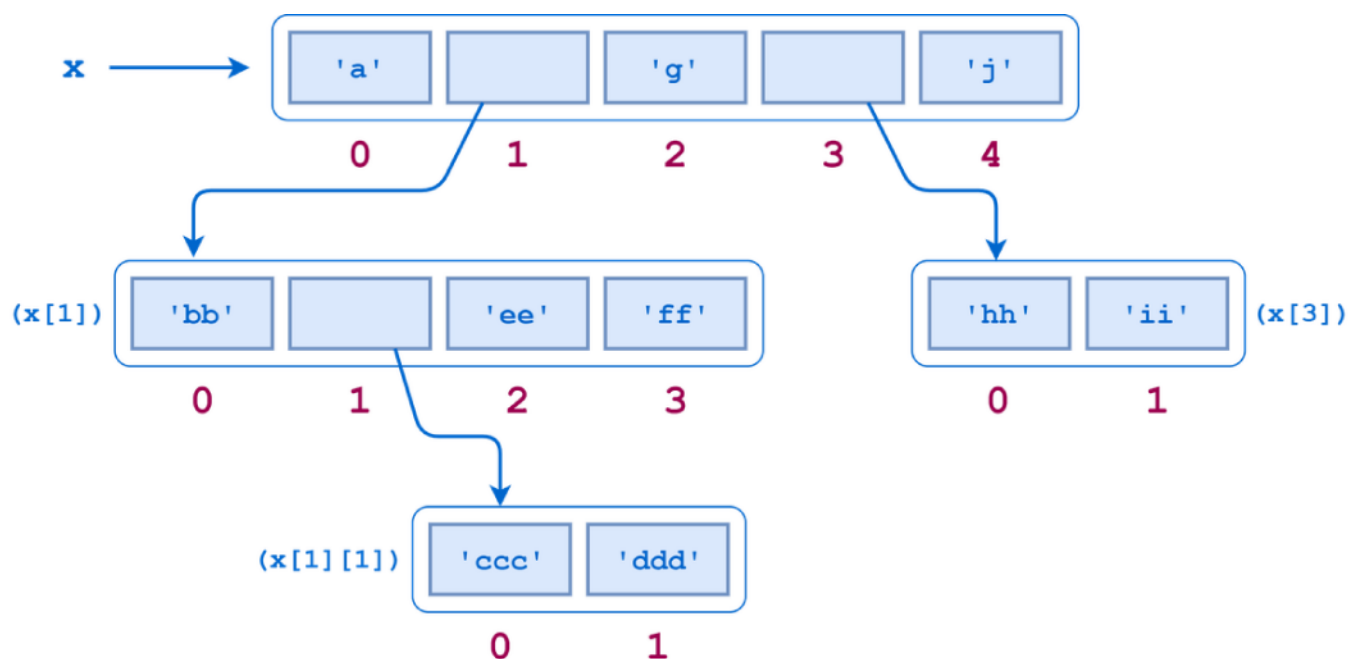
Out[85]:

```
['hello', 33, 'quux', 'corge']
```

## لیست های تودرتو

یک ویژگی بزرگ از ساختارهای داده در پایتون این است که آنها از تو در تو کردن پشتیبانی می کنند. این بدان معنی است که می توانیم ساختارهای داده را درون ساختارهای داده دیگر داشته باشیم. به عنوان مثال: یک لیست درون یک لیست.

بیایید ببینیم چگونه کار می کند!



In [86]:

```
1 lst1 = [1,2,3]
2 lst2 = [4,5,6]
3 lst3 = [7,8,9]
4
5
6 matrix = [lst1, lst2, lst3]
```

In [87]:

```
1 # Show
2 matrix
```

Out[87]:

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

ما می‌توانیم باز هم از عمل ایندکس‌گذاری استفاده کنیم تا عناصر را بگیریم، اما اکنون دو سطح برای ایندکس وجود دارد.  
عناصر درون شیء ماتریس و سپس عناصر درون آن لیست!

In [88]:

```
1 # Grab first item in matrix object
2 matrix[0]
```

Out[88]:

```
[1, 2, 3]
```

In [89]:

```
1 # Grab first item of the first item in the matrix object
2 [1, 2, 3][0]
```

Out[89]:

```
1
```

In [90]:

```
1 matrix[0][0]
```

Out[90]:

```
1
```

In [91]:

```
1 matrix[1]
```

Out[91]:

```
[4, 5, 6]
```

In [92]:

```
1 matrix[1][::-1]
```

Out[92]:

```
[6, 5, 4]
```

In [112]:

```
1 x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']
```

In [113]:

```
1 x[1]
```

Out[113]:

```
['bb', ['ccc', 'ddd'], 'ee', 'ff']
```

In [114]:

```
1 x[1][1]
```

Out[114]:

```
['ccc', 'ddd']
```

In [115]:

```
1 x[1][1][1]
```

Out[115]:

```
'ddd'
```

In [116]:

```
1 x[1][1][1][2]
```

Out[116]:

```
'd'
```

In [117]:

```
1 x[1][1][1].upper()
```

Out[117]:

```
'DDD'
```

In [118]:

```
1 'ddd'.upper()
```

Out[118]:

```
'DDD'
```

!بریم سراغ توپل ها