لیست در پایتون:

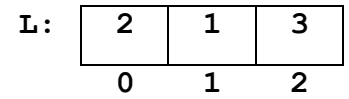
- لیست یک نوع داده چند قسمتی است. به عبارتی لیست یک توالی ترتیب دار از اطلاعات است، که با اندیس می‌توان به آن‌ها دسترسی داشت.
- یک لیست با براکت، [] مشخص می‌شود.
- یک لیست شامل عناصر است.
  - معمولا همگن، مثلا همه int
  - درپایتون این امکان وجود دارد که لیست ها ناهمگن باشد، اما رایج نیست.
- عناصر لیست را میتوان تغییر داد، به اصلاح mutable اند.

اندیس در لیست:

یک لیست شامل چندین عنصر است. با اندیس می‌توان به آن‌ها دسترسی داشت.

```
L1 = []  # لیست خالی
L2 = ['salam', 20, 12, True, 6]  # یک لیست ناهمگن با ۵ عنصر از اندیس ۰ تا ۴
L = [2, 5, 3]  # یک لیست همگن با ۳ عنصر از اندیس ۰ تا ۲
#-------------------------------------------------------
len(L)  # ارزیابی این عبارت عدد ۳ را نتیجه می‌دهد.
L[0]  # ارزیابی این عبارت عدد ۲ را نتیجه می‌دهد.
L[2]+1  # ارزیابی این عبارت عدد ۴ را نتیجه می‌دهد.
L[3]  # ارور یا خطا می‌دهد.
```

اندیس داده ها در لیست از چپ به راست و از صفر مشخص می شود.



اندیس لیست می‌توان عدد یا یک عبارت باشد، اما در نهایت حتما باید مقدار آن به یک عدد صحیح (int) ارزیابی شود.

```
i = 2
L[i-1]  # ارزیابی i-1 عدد ۱ را نتیجه می‌دهد. در نتیجه ارزیابی کل عبارت مقدار ذخیره شده در L[1] یعنی ۵ را نتیجه می‌دهد
```

ممکنه **لیستی درون یک لیست دیگر** داشته باشیم:

```
a= ['ali',20,['reza',100],1.5]
a[2]      # ['reza', 100]
a[2][0]   # 'reza'
```

همانطور که در بالا مشاهده می‌کنید از دو کروشه برای دسترسی به زیر لیست استفاده کردیم. کروشه‌ی نخست از لیست اصلی اندیس شماره ۲ که در واقع ['reza',100] و دومین کروشه از این زیر لیست عنصر اول (عنصری با اندیس ۰) را برمی‌گرداند.

تمرین کلاسی ۱) در لیست زیر، مقدار True را به False تغییر دهید.

```
L = [1, 2, [3, 'ali', 'reza', [5, True], 9], 8]
```

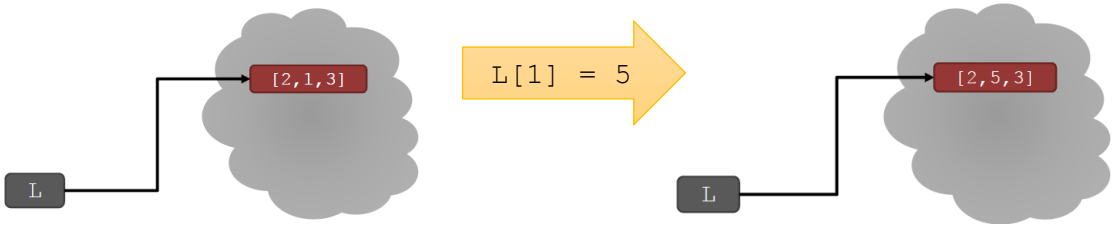
تغییر عناصر:

لیست‌ها در پایتون تغییر پذیر[1] هستند. به این معنی که با انتساب مقدار جدید به یک اندیس از لیست آن مقدار تغییر می‌کند.

```
L = [2, 1, 3]
L[1] = 5
```

L حالا برابر [2 ,5 ,3] است.

✍ **نکته:** پس از انجام انتساب بالا شئ L همان شئ قبلی است و تنها مقدار اندیس شماره ۱ آن تغییر کرده است.



لیست‌ها در این خاصیت با رشته‌ها متفاوت هستند. در واقع رشته‌ها تغییرناپذیر[2] هستند.

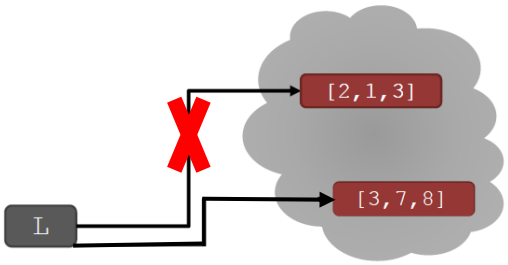
در نتیجه عبارت زیر با خطا مواجه می‌شود:

```
S = 'example'
S[1] = 'g' #---> Error
```

[1] Mutable
[2] Immutable

اما انتساب یه مقدار جدید به لیست، باعث ایجاد یک شئ جدید در حافظه و مقید سازی آن متغیر به آن شئ جدید می‌شود:

```
L = [2, 1, 3]
L = [3, 7, 8]
```



جمع عناصر:

فرض کنید یک لیست از اعداد به نام L داریم، اگر بخواهیم جمع عناصر این لیست را حساب کنیم خواهیم داشت:

✓ در جدول زیر این مثال با ۲ روش مختلف حل شده است.

```
total = 0
for i in range(len(L)):
    total += L[i]
print(total)
```

```
total = 0
for e in L:
    total += e
print(total)
```

نکته:

- ✓ اندیس عناصر لیست از ۰ تا `len(L)-1` است.
- ✓ `range(n)` اعداد ۰ تا `n-1` را تولید می‌کند.
- ✓ همچون رشته‌ها؛ میتوان روی خود عناصر لیست `iterate` انجام داد. (حالت سمت چپ)

تمرین کلاسی ۲) فرض کنید لیستی از اعداد صحیح به نام a داریم، برنامه ای بنویسید که تنها اعداد زوج را چاپ کند.

تمرین کلاسی ۳) فرض کنید لیستی از اعداد به نام b داریم، برنامه ای بنویسید که معدل این اعداد را حساب کند.

عملیات روی لیست:

- اضافه کردن یک عنصر به لیست:

می‌توانیم یک عنصر را با `L.append(element)` به انتهای یک لیست اضافه کنیم.

```
L = [2, 1, 3]
L.append(5)#L is now [2, 1, 3, 5]
```

- **تسلسل[3] دو لیست:**
  - برای ترکیب دو لیست با یکدیگر از عملگر + استفاده می‌شود، حاصل یک لیست جدید است.
  - می‌توان لیست L را با `L.extend(some_list)` گسترش داد. در این صورت لیست L همان شئ قبلی است که عناصری جدید به آن اضافه شده است.

```
L1 = [2, 1, 3]
L2 = [4, 5, 6]

L3 = L1 + L2 #L3 is [2, 1, 3, 4, 5, 6]
L4 = L1 + [12, 9] #L4 is [2, 1, 3, 12, 9]


L1.extend ([0, 6]) #mutated L1 to [2, 1, 3, 0, 6]
L1.extend (L2)     #mutated L1 to [2, 1, 3, 0, 6, 4, 5, 6]
```

✍ **نکته:** عملگر + هیچ کدام از طرفین + را تغییر نمی‌دهد و یک لیست جدید می‌سازد، اما extend لیست سمت چپ نقطه را تغییر داده و لیست جدیدی ایجاد نمی‌کند.

✍ **نکته:** بدیهی است append و extend با هم فرق دارند. به دو لیست a و b دقت کنید:

```
a = [2, 3, 5]
a.append(7) #a is now [2, 3, 5, 7]
a.extend(11) #---> Error


b = [2, 3, 5]
b.append([7, 11]) #a is now [2, 3, 5, [7, 11]]
b.extend([13, 17])#a is now [2, 3, 5, [7, 11],13 ,17]
```

[3] Concatenation

- حذف یک عنصر از لیست:

```
L = [2, 1, 3, 6, 3, 7, 5] # به ترتیب تاثیر عملیات زیر بر روی این لیست رادقت کنید
L.remove(2) # L = [1,3,6,3,7,5]
```

حذف اولین عنصر که مقدار ۲ دارد.

```
L.remove(3) # L = [1,6,3,7,5]
```

حذف اولین عنصر که مقدار ۳ دارد. (دقت کنید که بازهم در لیست عدد ۳ وجود دارد.)

```
del(L[1])    # L = [1,3,7,5]
```

حذف عنصری با اندیس ۱ در لیست L.

```
L.pop()      # 5 and mutates L = [1,3,7]
```

برگرداندن آخرین عنصر لیست، و حذف آن از لیست.

مثال) کد زیر چه چیزی چاپ می‌کند؟

```
L = [2, 1, 3, 6, 3, 7, 5]
a = L.pop()
b = L.pop()
print(a, b)
print(L)
```

پاسخ:

5 7

[2, 1, 3, 6, 3]

تمرین کلاسی ۴) فرض کنید یک لیست از اعداد به نام L داریم؛ برنامه‌ای بنویسید که تمام اعداد برابر با ۷ را از لیست L پاک کند. (برای اجرای صحیح و بدون خطای برنامه, خودتان یک لیست به نام L با مقادیر دلخواه ایجاد کنید.)

- مرتب سازی عناصر یک لیست:

```
L=[9,6,0,3]
sorted(L) # لیست اصلی را تغییر نمی‌دهد, یک لیست جدید مرتب شده بر می‌گرداند
L.sort() # L=[0,3,6,9]
```

مثال) پس از اجرای کد زیر چه چیزی چاپ می‌شود؟

```
a1= [5, 3, 7, 9, 2]
a2= [12, 4, 33, 1]
b=sorted(a1)
a2.sort()
print(a1)
print(a2)
print(b)
```

پاسخ:

[5, 3, 7, 9, 2]

[1, 4, 12, 33]

[2, 3, 5, 7, 9]