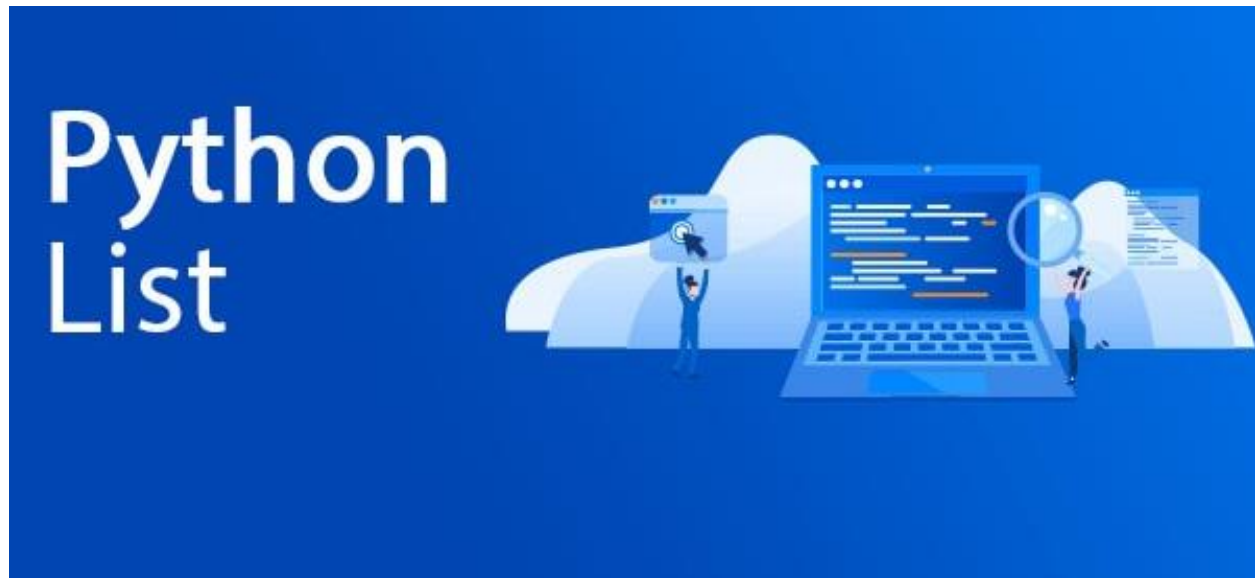




برنامه نویسی پایتون List ها

لیست در پایتون یکی از پر کاربردترین نوع داده‌ای است که تقریباً در تمام برنامه‌های خود از آن استفاده می‌کنیم. اگر بتوانیم در کار با لیست پایتون به مهارت کافی برسیم، روند برنامه نویسی ما سرعت بیشتری خواهد گرفت. به کمک لیست‌ها خواهیم توانست تعداد مقادیر زیادی را در یک متغیر نگه داشته و حتی روی آن عملیات‌های مختلفی را اجرا کنیم. اگر با زبان‌های برنامه نویسی مختلف کار کرده باشید، احتمالاً با آرایه‌ها آشنایی دارید. در یک تعریف بسیار کلی می‌توان لیست را یک نوع آرایه در پایتون دانست؛ آرایه‌ای که طول آن متغیر بوده و عناصر آن می‌توانند از انواع مختلف داده‌ای باشند.

لیست در پایتون

لیست یک ساختمان داده مهم در پایتون است. به کمک لیست‌ها می‌توانیم دنباله‌ای از داده‌ها را در یک متغیر ذخیره کرده و روی آن‌ها عملیات‌های مختلفی را اجرا کنیم.

نوع داده‌ای لیست یکی از انواع داده مجموعه‌ای یا **Collections** در این زبان محبوب است. لیست شامل مجموعه‌ای از عناصر به صورت ترتیبی و قابل تغییر است. لیست در پایتون با کروشه (براکت یا **bracket** به شکل `[]` مشخص می‌شود؛ به طوری که با علامت کروشه باز `[` لیست آغاز شده و تا علامت کروشه بسته `]` ادامه می‌یابد.

هر عنصر درون لیست به وسیله کاما (ویرگول انگلیسی یا **comma** به شکل `,` از یکدیگر جدا شده و می‌تواند از هر نوع داده دلخواهی باشد.

```
lst = ["Omid", ۲۱۷, ۲۲,۵, "SabzDnaesh"]
```

در یک خط کد بالا، یک لیست با نام **lst** ایجاد کرده‌ایم. سپس درون آن چهار مقدار با نوع داده‌ای مختلف قرار داده‌ایم.

کاربرد لیست

از لیست در هر جایی که بخواهید می‌توانید استفاده کنید! هر کجا که دنباله‌ای معنی‌دار یا بدون معنی از داده‌ها دارید، لیست‌ها سریع‌ترین و شاید بهترین انتخاب شما باشند. مزیت لیست‌ها در پایتون این است که به سادگی تعریف شده، اعضای آن اضافه یا کم می‌شوند. همچنین اعمال بین لیست‌ها بسیار ساده و سریع است. در ادامه توابع و تکنیک‌هایی به شما معرفی می‌کنیم که به راحتی هر بلایی که خواستید سر لیست‌ها در بیاورید!

دسترسی به اعضای لیست در پایتون

همانطور که پیش‌تر نیز گفته شد، ساده‌ترین نوع تعریف یک لیست در پایتون، استفاده از دو علامت کروشه در کنار هم است. با این تعریف، متوجه خواهید شد که امکان ایجاد یک لیست خالی نیز وجود دارد. در نتیجه حتماً نیازی نیست که در ابتدای کار تمام مقادیر درون لیست را داشته باشیم؛ بلکه در حین برنامه خواهیم توانست مقادیر دلخواه خود را به آن اضافه یا از آن کم کنیم.

دسترسی به اعضای لیست بر اساس اندیس‌های آن صورت می‌گیرد. اندیس‌ها اعدادی هستند که موقعیت هر عنصر در لیست را مشخص می‌کنند.

اولین عنصر یک لیست دارای اندیس صفر (۰) و به طور کلی n آمین عنصر دارای اندیس $n-1$ خواهد بود.

اندیس لیست (ایندکس یا نمایه)

فرض کنید یک لیست از اسامی افراد مختلف را درون لیست خود داریم.

```
lst = ["sara", "omid", "amir", "roya", "ehsan"]
```

برای دسترسی به هر عنصر از لیست، با داشتن موقعیت یا اندیس آن عنصر و آوردن اندیس در جلوی نام متغیر لیست، به عنصر مورد نظرمان دسترسی خواهیم داشت.

یعنی قطعه کد زیر به ما عنصری که در اندیس شماره ۳ قرار دارد با خروجی می‌دهد؛ یعنی چهارمین عنصر در لیست که معادل **roya** است.

```
lst[۳]
```

به همین سادگی خواهیم توانست به تمام عناصر درون یک لیست دسترسی پیدا کنیم.

توجه کنید که اندیس یک مقدار از نوع عددی صحیح (**integer**) است. در نتیجه فراخوانی اندیس اعشاری یا رشته‌ای ما را با خطا مواجه خواهد کرد.

در لیست بالا ما پنج عنصر داریم. بازه اندیس‌های ما از ۰ تا ۴ خواهد بود. وارد کردن عددی بزرگ‌تر از ۴ ما را با خطای اندیس (**IndexError**) مواجه خواهد کرد.

اندیس منفی لیست در پایتون

زبان برنامه نویسی پایتون تعریف اندیس برای اعضای یک لیست را امکان پذیر کرده است. به این صورت که آخرین عنصر هر لیست دارای اندیس -1 ، عنصر یکی مانده به آخر -2 و همینطور تا عنصر اول...

به کمک اندیس منفی در پایتون می توانیم به عناصر یک لیست از انتهای آن دسترسی داشته باشیم.

به طور کلی اندیس های عناصر یک لیست در پایتون به صورتی که در تصویر زیر مشخص شده در نظر گرفته می شوند.

ساختار لیست در پایتون									
اندازه لیست: ۹									
عناصرها (Elements)	6	11	5	12	94	"omid"	42	"sara"	21 9
اندیس ها (Indices)	۰	۱	۲	۳	۴	۵	۶	۷	۸ ۹
اندیس های منفی (Neg. Indices)	-۱۰	-۹	-۸	-۷	-۶	-۵	-۴	-۳	-۲ -۱

عناصر خانه ی هفتم

ساختار لیست در پایتون و ایندیس های مجاز

بنابراین برای چاپ اسم **ehsan** در لیست مفروض خود، کدی شبیه به کد زیر خواهیم داشت.

```
print( lst[-۱] )
```

Output: ehsan

سوال: با توجه به این توضیحات، به نظرتان بازه اندیس منفی مجاز در پایتون در یک لیست با n عضو چیست؟

دسترسی به بخشی از لیست (برش زدن List)

یکی از ویژگی ها و قابلیت های جذاب لیست ها در پایتون، توانایی برش زدن لیست یا **Slicing** است.

بگذارید این موضوع را همراه با یک مثال برایتان توضیح دهم. فرض کنید لیست زیر را در برنامه خود ایجاد کرده ایم.

```
lst = ['s', 'a', 'b', 'z', 'd', 'a', 'n', 'e', 's', 'h']
```

یک لیست با اندازه ۱۰. اگر بخواهیم فقط به بخشی از این لیست دسترسی داشته باشیم باید چگونه عمل کنیم؟
ویژگی **slicing** یا برش زدن **list** این امکان را به ما می‌دهد که فقط به بخشی از یک لیست دسترسی داشته باشیم.

دسترسی به لیست با تعریف بازه اندیس

برای تعریف یک بازه اندیس در هنگام فراخوانی عناصر لیست، کافی است به جای وارد کردن یک شماره اندیس، بازه آن را مشخص کنیم. مشخص کردن بازه با علامت دو نقطه (:) انجام می‌شود.

```
lst[۳:۷]
```

```
# Output: ['z', 'd', 'a', 'n']
```

در قطعه کد بالا، عناصری که اندیس آن‌ها در بازه ۳ تا ۷ بود به ما برگردانده شد. توجه کنید که بازه اندیس‌ها چگونه تعریف شده است:

اگر بازه به صورت **i:j** تعریف شود، این تعریف شامل عنصر **i** تا عنصر **j-۱** خواهد شد و عنصر **j** در فراخوانی آورده نمی‌شود.

اگر هر یک از کران بازه را مشخص نکنیم، به طور پیش‌فرض تا انتهای آن را در نظر خواهد گرفت. یعنی برای فراخوانی اعضای لیست از عنصر دوم به بعد به صورت زیر عمل می‌کنیم.

```
lst[۲:]
```

```
# Output:
```

```
# ['b', 'z', 'd', 'a', 'n', 'e', 's', 'h']
```

در قطعه کد بالا، عناصر از اندیس دوم تا انتهای لیست به ما نمایش داده می‌شود.

همچنین برای اینکه به بخشی از لیست از ابتدا تا اندیسی مشخص دسترسی داشته باشیم، می‌توان به صورت زیر عمل کرد.

```
lst[:۲]
```

```
# Output:
```

```
# ['s', 'a', 'b', 'z', 'd', 'a', 'n', 'e']
```

در اینجا تمام عناصر موجود در لیست را از ابتدای آن تا به دو تا مانده به آخری (اندیس ۲-را فراخوانی کرده ایم.

تکنیک: دسترسی به میان لیست در پایتون

همانطور که احتمالاً تا به اینجا کار برداشته کرده اید، می توانیم از اندیس های مثبت و منفی در هنگام برش دادن لیست استفاده کنیم.

اگر این دو نوع اندیس را به طور همزمان استفاده کنیم، خواهیم توانست خروجی های جالب و شگفت انگیزی از لیست خود بگیریم!

برای مثال، فرض کنید در لیست دلخواه خود، می خواهیم دو عنصر اول و دو عنصر آخر لیست را حذف کرده و بقیه را نشان دهیم. برای این کار به راحتی کد زیر عمل خواهیم کرد. به این روش می توانید لیست را به هر صورتی که مایلید برش دهید!

```
lst[۲:-۲]
```

```
# Output:
```

```
# ['b', 'z', 'd', 'a', 'n', 'e']
```

تکنیک: دسترسی دنباله ای به اعضای لیست (همراه با گام)

فرض کنید یک لیست از اعداد ۱ تا ۲۰ در اختیار دارید و می خواهید اعداد را دو در میان از آن خارج کنید. اولین راهی که احتمالاً به ذهنمان می رسد استفاده از [حلقه های پایتونی](#) (نظیر **for** برای پیمایش لیست است.

این موضوع را جلوتر بررسی خواهیم کرد. اما این جا یک ویژگی جذاب در هنگام تعریف بازه لیست در پایتون را با هم یاد خواهیم گرفت.

همانطور که آموختید، با گذاشتن یک علامت دو نقطه (:) می‌توان بازه‌ای برای اندیس‌های لیست تعریف کرد. اگر پس از تعریف بازه اندیس، یک بار دیگر دو نقطه بگذاریم، خواهیم توانست گام حرکت خود را نیز مشخص کنیم.

برای درک بهتر موضوع به تصویر زیر توجه کنید:

نحوه برش لیست و حرکت با گام

همانطور که در تصویر نیز مشخص است، در ابتدا بازه ابتدا و انتهای حرکت را مشخص کرده‌ایم و سپس گفته‌ایم که با چه گامی بر روی آن‌ها حرکت کرده و خروجی را به ما بدهد.

یک لیست به نام **numbers** ایجاد کرده و اعداد ۱ تا ۲۰ را در آن قرار می‌دهیم. سپس به صورت یک در میان، از ابتدا تا انتها آن حرکت می‌کنیم.

```
numbers = [۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹, ۱۰, ۱۱, ۱۲, ۱۳, ۱۴, ۱۵, ۱۶, ۱۷, ۱۸, ۱۹, ۲۰]
```

```
numbers::۲]
```

Output:

```
# [۱, ۳, ۵, ۷, ۹, ۱۱, ۱۳, ۱۵, ۱۷, ۱۹]
```

به این صورت تمام اعداد فرد موجود در لیست را فراخوانی کرده‌ایم. به همین سادگی!

حال اگر بخواهیم از عنصر پنجم شروع کرده و سه تا سه تا حرکت کنیم چه؟ کد آن چیزی مشابه به کد زیر خواهد شد:

```
numbers[۴::۳]
```

Output:

```
# [۵, ۸, ۱۱, ۱۴, ۱۷, ۲۰]
```

امیدوارم به خوبی با نحوه برش لیست و تکنیک‌های آن آشنا شده باشید و از آن در زمان لازم استفاده کنید.

لیست تو در تو یا لیست چند بعدی در پایتون

عناصر لیست‌ها هر چیزی می‌توانند باشند؛ از عدد و رشته گرفته تا یک لیست دیگر! بنابراین اگر عناصر یک لیست، یک لیست دیگر باشد، لیست‌های تو در تو (**Nested List**) را خواهیم داشت.

فرض کنید یک لیست داریم که اطلاعات یک دانشجو را نگهداری می‌کند. می‌خواهیم شناسه (**id**) ، اسم و کلیه نمرات او را داشته باشیم.

```
student = [۶۲۱۴, "Mitra", [۱۸, ۱۷, ۲, ۱۹, ۵, ۱۶, ۲۰, ۱۹, ۹, ۲۰]]
```

عنصر سوم این لیست، خودش یک لیست است. به همین صورت می‌توان لیست‌های چند سطحی را نیز ایجاد کرد.

تعریف ماتریس در پایتون

همانطور که می‌دانید ماتریس یک لیست تو در تو است که اعضای آن اندازه یکسان و مشخصی دارند.

به طور پیش‌فرض در پایتون ساختمان داده ماتریس نداریم. البته به کمک برخی کتابخانه‌ها و ابزارها می‌توان چنین ساختمان داده‌ای را ایجاد کرده و از آن استفاده کرد.

ما در این جا ساده‌ترین حالت ایجاد یک ماتریس در پایتون را با هم بررسی می‌کنیم.

برای ایجاد ماتریس، کافی است یک لیست ایجاد کنیم که هر عنصر آن یک لیست دیگر باشد.

```
mtrx = [[], [], []]
```

بهتر است در طول برنامه خود همواره بررسی کنیم که اندازه لیست‌های داخلی از اندازه مشخص شده بیشتر نشود. راه‌حل جایگزین بررسی اندازه لیست‌ها، استفاده از تابع اندازه لیست (**len()**) است که در قسمت توابع کاربردی معرفی می‌شود.

مثلاً اطلاعات مجموع ساعت کاری کارمندان در سه روز از هفته را در ماتریس زیر نگهداری کرده‌ایم.

```
logs = [['Mon', '۱۰۶/۲۲', 'sina', ۱۲],  
        ['Tue', '۱۰۶/۲۳', 'ehsan', ۷],  
        ['Wed', '۱۰۶/۲۴', 'roya', ۱۰],  
        ['Wed', '۱۰۶/۲۴', 'sina', ۵ ]]
```

کار با عناصر لیست

علاوه بر دسترسی به عناصر یک لیست، نیاز به وارد کردن، خارج کردن یا تغییر داده‌ها در لیست خواهیم داشت. برای هر کدام از این کارها روش‌ها یا توابع مختلفی در پایتون وجود دارد. در ادامه به مهم‌ترین و کاربردی‌ترین آن‌ها را بررسی می‌کنیم.

افزودن عنصر جدید به لیست

برای افزودن یک عنصر جدید به لیست فعلی، سه روش وجود دارد. در دو روش اول، عنصر به انتهای لیست اضافه می‌شود؛ اما به کمک روش سوم می‌توانیم عنصر را در اندیس دلخواه خود قرار دهیم.

لیست **nums** را با سه مقدار اولیه ایجاد کرده و سپس به آن عناصری را اضافه می‌کنیم.

```
nums = [۳۲, ۱۵, ۲۰]
```

۱. افزودن عنصر جدید به لیست با تابع **append**

با صدا زدن تابع **append()** روی لیست، مقدار ورودی خود را به عنوان عنصر جدید در انتهای لیست درج می‌کند. این متد صرفاً یک ورودی گرفته و نوع داده‌ای آن اهمیتی ندارد.

```
nums.append(۱۱)
```

```
print(nums)
```

Output:

```
# [۳۲, ۱۵, ۲۰, ۱۱]
```

```
nums.append(۷۳)
```

```
print(nums)
```

```
# Output:
```

```
# [۳۲, ۱۵, ۲۰, ۱۱, ۷۳]
```

۲۰. افزودن سریع عنصر به لیست پایتون با علامت جمع (+)

این روش که به نوعی اتصال دو لیست به یکدیگر محسوب می‌شود، می‌تواند یکی از سریع‌ترین روش‌های افزودن عنصر یا عناصر جدید به لیست مورد نظرمان باشد.

در قسمت توابع کاربردی در مورد اتصال دو لیست به یکدیگر بیشتر صحبت می‌کنیم.

برای درک بهتر، قبل از توضیح چگونگی انجام کار، یک مثال می‌زنم. به مثال زیر توجه کنید:

```
nums = [۳۲, ۱۵, ۲۰]
```

```
nums = nums + [۲۵]
```

به همین سادگی، عدد ۲۵ را به انتهای لیست `nums` اضافه کردیم!

اگر لیست را چاپ کنیم، می‌بینیم که لیست ما با مقدار جدید به‌روزرسانی شده است.

```
print(nums)
```

```
# Output:
```

```
# [۳۲, ۱۵, ۲۰, ۲۵]
```

حال اگر درون لیست دومی که در حال جمع بستن آن هستیم، چند مقدار مختلف وجود داشت، تمام آن‌ها به انتهای لیست اضافه خواهند شد.

```
nums = nums + [۶۸, ۲۳, ۸]
```

```
print(nums)
```

Output:

```
# [۳۲, ۱۵, ۲۰, ۲۵, ۶۸, ۲۳, ۸]
```

تکنیک: خلاصه کردن عملیات جمع

همانطور که احتمالاً می‌دانید، به جای عبارت `x = x + ۳` می‌توانیم بنویسیم `x += ۳`. این قضیه برای جمع بین دو لیست نیز برقرار است.

یعنی دو خط کد زیر معادل یکدیگرند.

```
nums = nums + [۲۵]
```

```
nums += [۲۵]
```

۳. تابع `insert`

تابع `insert()` روی لیست صدا زده شده و دو ورودی می‌گیرد:

- **آرگومان اول**: شماره ایندکس مورد نظر (ما به صورت عدد صحیح `integer`)

- **آرگومان دوم**: عنصر مورد نظر برای افزودن به لیست

برای شماره ایندکس وارد شده دو حالت وجود دارد:

۱. ایندکس خارج از محدوده و اندازه لیست است؛ در این صورت عنصر در انتهای لیست اضافه خواهد شد.

۲. ایندکس داخل بازه ایندکس‌های فعلی است؛ عنصر در خانه مورد نظر درج شده و عنصر فعلی این خانه و خانه‌های بعد از آن یکی به سمت راست شیفت داده می‌شوند.

```
nums = [۳۲, ۱۵, ۲۰]
```

```
nums.insert(۱, ۴۵)
```

```
print(nums)
```

```
# [۳۲, ۴۵, ۱۵, ۲۰]
```

```
nums.insert(۱۰۰, ۶۵)
```

```
print(nums)
```

```
# [۳۲, ۴۵, ۱۵, ۲۰, ۶۵]
```

حذف عنصر از لیست در پایتون

وقتی می‌خواهیم مقداری را از لیست حذف کنیم، دو حالت وجود دارد.

- ایندکس عنصر مورد نظر را می‌دانیم.
- مقدار عنصر مورد نظر را می‌دانیم.

برای هر یک از این حالات دو دستور متفاوت را معرفی می‌کنم. از هر کدام می‌توانید در جای مناسب خود استفاده کرده و عنصر مورد نظر را از لیست حذف کنید.

حذف عنصر با ایندکس به کمک دستور `del`

دستور `del` در پایتون برای حذف متغیرها استفاده می‌شود. به این صورت که هر گاه این کلمه کلیدی را قبل از نام متغیر بیاوریم، آن متغیر به طور کامل از درون برنامه ما حذف خواهد شد.

به کمک این دستور، می‌توان کل لیست یا یک عنصر از آن را حذف کرد.

اگر نام لیست یا لیست همراه به ایندکس را در جلوی کلمه کلیدی `del` قرار دهیم، لیست یا عنصر مورد نظر به طور کامل حذف خواهد شد.

```
lst = ['s', 'a', 'b', 'z', 'd', 'a', 'n', 'e', 's', 'h']
```

```
del lst[۳]
```

```
print(lst)
```

```
# ['s', 'a', 'b', 'd', 'a', 'n', 'e', 's', 'h']
```

```
del lst  
print(lst)
```

```
# Traceback (most recent call last):  
# File "<pyshell#۷۹>", line ۱, in <module>  
#     print(lst)  
# NameError: name 'lst' is not defined
```

در خط دوم قطعه کد بالا، اقدام به حذف عنصر موجود در ایندکس ۳ (لیست) معادل با عنصر **z** کردیم. همانطور که در نتیجه مشاهده می‌کنید، این عنصر از درون لیست حذف شده است.

در خط ششم، کل متغیر لیست را حذف کرده‌ایم. پس از اقدام برای چاپ لیست، با خطای عدم تعریف متغیر در پایتون رو به رو می‌شویم.

استفاده از متد **remove** برای حذف عنصر از لیست

این متد بر روی لیست صدا زده می‌شود و یک ورودی می‌گیرد. آنچه که به عنوان ورودی گرفته را در لیست جستجو کرده و اولین موردی که مطابقت داشت را حذف می‌کند.

در لیست زیر، ما دو بار حرف **a** را داریم. با صدا زدن متد **remove('a')** اولین موردی که در لیست پیدا می‌شود (یعنی ایندکس ۱) حذف خواهد شد.

```
lst = ['s', 'a', 'b', 'z', 'd', 'a', 'n', 'e', 's', 'h']  
lst.remove('a')  
print(lst)  
# ['s', 'b', 'z', 'd', 'a', 'n', 'e', 's', 'h']
```

تغییر عنصر در لیست

برای تغییر مقدار موجود در ایندکس مورد نظر، می توان پس از فراخوانی آن خانه از لیست، اقدام به تغییر آن کنیم. این کار با علامت انتصاب یا = انجام می شود.

```
lst = ["sara", "omid", "amir", "roya", "ehsan"]  
lst[۲] = "hamed"  
print(lst)  
# ['sara', 'omid', 'hamed', 'roya', 'ehsan']
```

اگر عنصر لیست ما یک مقدار عددی باشد و بخواهیم آن را کاهش یا افزایش بدهیم، به صورت مشابه انجام خواهد شد.

```
nums = [۱۵, ۲۴, ۶۰]  
nums[۱] += ۲  
print(nums)  
# [۱۵, ۲۶, ۶۰]
```