منطبق با جدیدترین نگارش - قسمت 25 و پایانی

آموزش رایگان پایتون (python)- ساخت، ویرایش و مدیریت فهرست‌ها در پایتون

در شماره گذشته آموزش پایتون مقدمه‌ای در رابطه با فهرست‌ها آوردیم و گفتیم که فهرست‌ها برنامه‌های ما را منعطف کرده و قدرت بیشتری در اختیار ما قرار می‌دهند. در این شماره قصد داریم با نحوه ساخت و مدیریت فهرست‌ها آشنا شویم. با ارایه این قسمت بخش اول این آموزش پایان می‌گیرد. بدیهی است که هنوز مباحثی از پایتون مانده است که امیدواریم از در بخش بعدی این آموزش به آن ها هم بپردازیم.

برای مطالعه بخش بیستم و چهارم آموزش رایگان پایتون اینجا کلیک کنید

ساخت فهرست‌ها

همانند دنیای واقعی قبل از آن‌که بتوانید با فهرست‌ها کار کنید، ابتدا باید آن‌ها را ایجاد کنید. در شماره گذشته گفتیم که فهرست‌ها در **پایتون** می‌توانند ترکیبی از نوع‌های مختلف باشند. با این حال، اما بهتر است که فهرست‌های خود را محدود به یک نوع مشخص کنید. اما چگونه می‌توانیم فهرست‌ها را ایجاد کنیم؟ برای ساخت یک فهرست در **پایتون** مراحل زیر را انجام دهید:

1. IDLE را باز کرده و دستور زیر را تایپ کنید و کلید اینتر را فشار دهید.

```
List1 = ["One", 1, "Two", True]
```

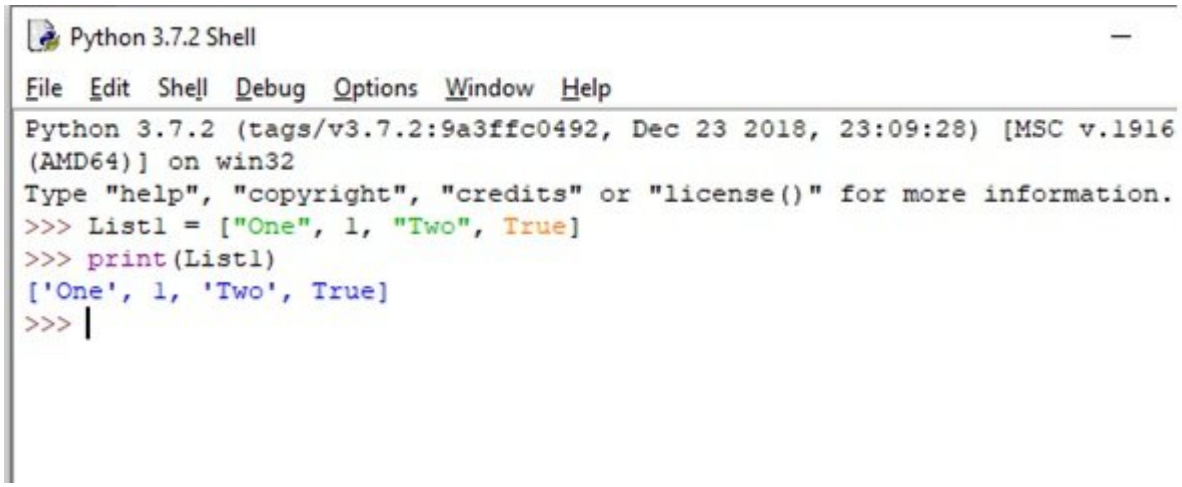
با اجرای این فرمان **پایتون** برای شما فهرستی به نام List1 ایجاد می‌کند. این فهرست شامل دو مقدار رشته‌ای One و Two و یک مقدار صحیحی (1) و یک مقدار منطقی (True) است. دقت کنید هر نوع داده‌ای که درون فهرست وارد می‌کنید یک رنگ مخصوص به خود دارد. زمانی که از اسکیمای پیش‌فرض استفاده می‌کنید، **پایتون** رشته‌ها را به رنگ سبز، اعداد را مشکی و مقادیر منطقی را به رنگ نارنجی نشان می‌دهد. رنگ هر نوع داده‌ای به شما اعلام می‌دارد در حال تایپ چه چیزی هستید.

2. اکنون فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

```
Print(List1)
```

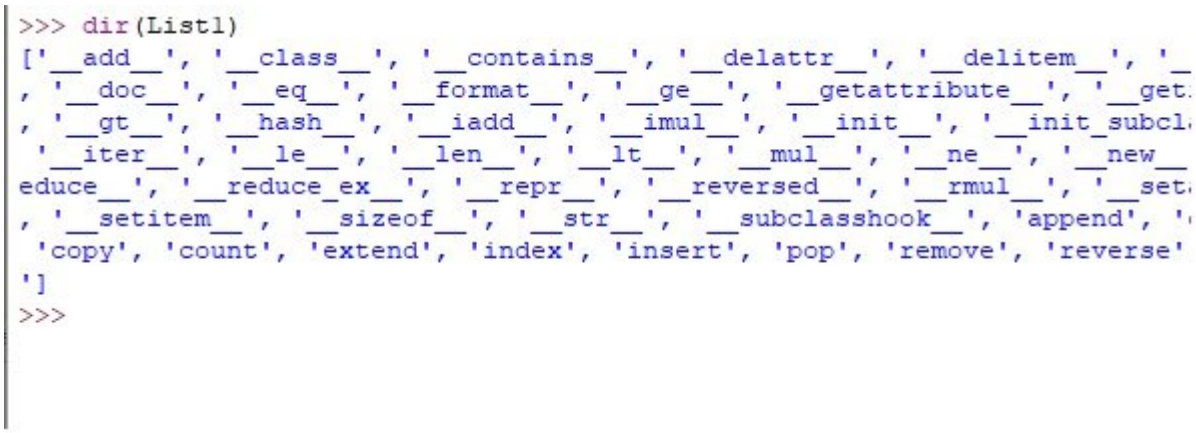
با اجرای این فرمان محتوای فهرستی که ایجاد کرده‌اید چاپ می‌شود. دقت کنید که مقادیر رشته‌ای درون گیومه‌هایی

به شکل متمایز نشان داده می‌شوند.



4. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

dir(List1)



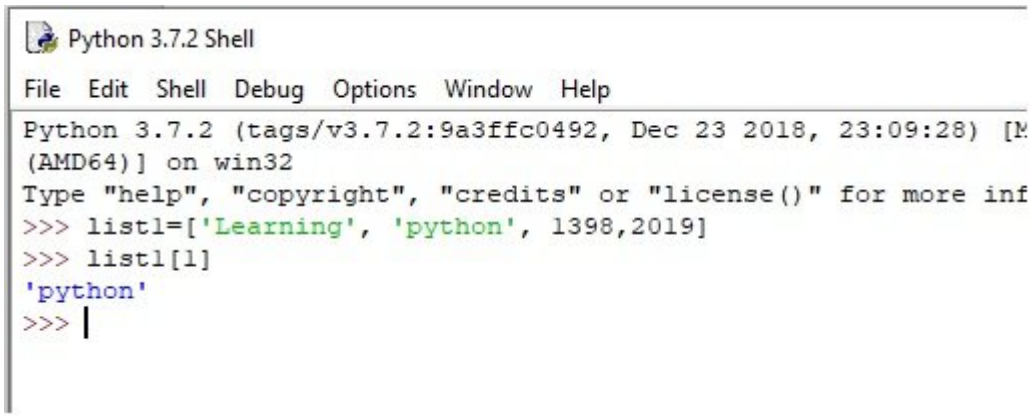
با اجرای این فرمان **پایتون** فهرستی از کارهایی که با استفاده از فهرست‌ها قادر به انجام آن‌ها هستید را نشان می‌دهد. دقت کنید خروجی فرمان dir نیز خود یک فهرست است! دقت کنید که همانند برش‌ها در رشته‌ها، اندیس فهرست‌ها نیز از مقدار 0 آغاز می‌شود. **پایتون** به شما اجازه می‌دهد فهرست‌ها را به یکدیگر چسبانده یا آن‌ها را برش دهید.

دسترسی به مقادیر

برای آن‌که به مقادیر درون رشته‌ها دسترسی داشته باشید باید از کروشه همراه با نام فهرست استفاده کنید. به مثال زیر دقت کنید:

list1=['Learning', 'python', 1398,2019]

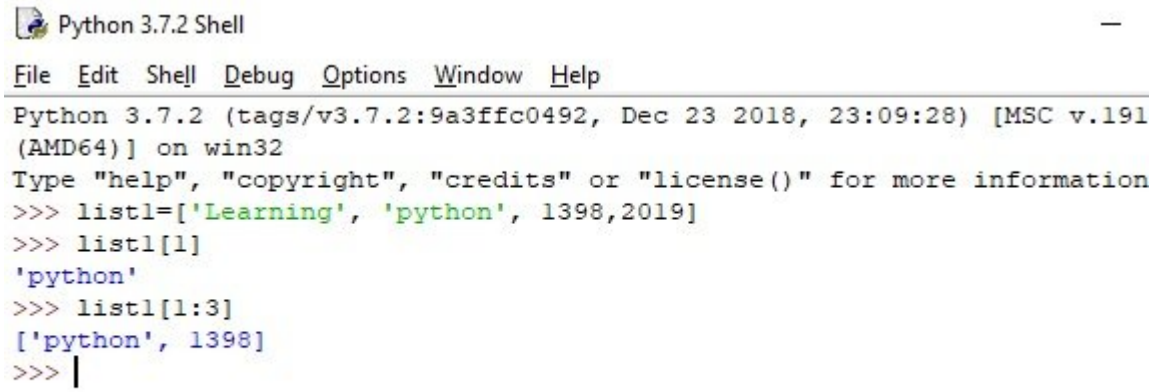
اکنون عبارت list1[1] را تایپ کرده و کلید اینتر را فشار دهید. همان‌گونه که مشاهده می‌کنید خروجی این دستور واژه python است. عددی که درون براکت‌ها استفاده می‌شود اندیس نام دارد.



اکنون دستور زیر را نوشته و کلید اینتر را فشار دهید:

List1[1:3]

محدوده‌ای از مقادیر را مشاهده می‌کنید که شامل دو عنصر است. زمانی که محدوده‌ای را تعیین می‌کنید، پایان محدوده همیشه بزرگ‌تر از تعداد عناصر بازگشتی است. این حرف به معنای آن است که شما به جای عناصر 1 تا 3 عناصر 1 و 2 را دریافت می‌کنید.



اکنون دستور زیر را تایپ کرده و کلید اینتر را فشار دهید.

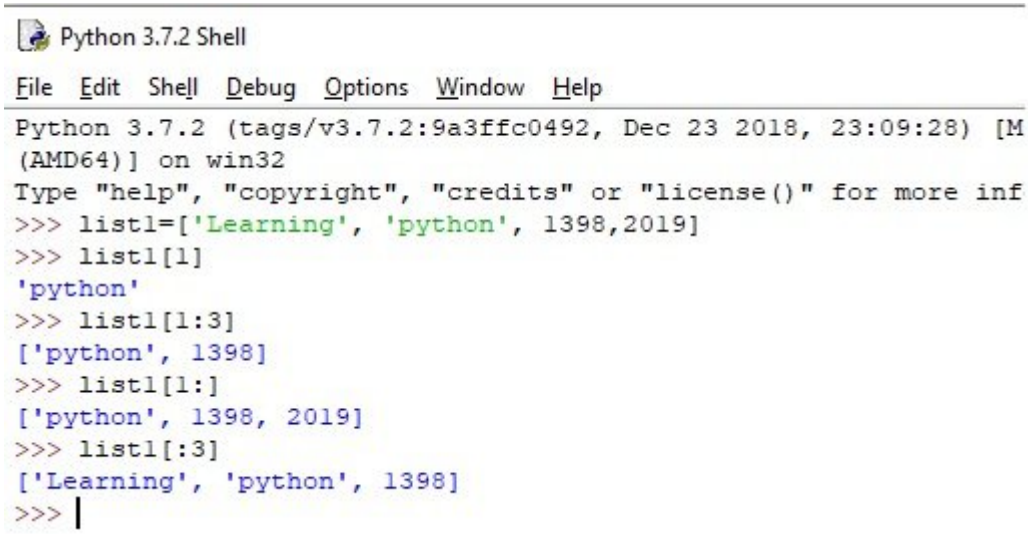
List1[1:]

این مرتبه همه عناصر را مشاهده خواهید کرد. عناصری که از مکان 1 فهرست تا پایان فهرست قرار دارند.

اکنون عبارت زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1[:3]

پایتون عناصری که در محدوده 0 تا 2 قرار دارند را نشان می‌دهد.



نکته: اگر در زمان تعیین اندیس فهرست‌ها از علامت منفی استفاده کنید، **پایتون** به جای آن‌که فرآیند نمایش را از سمت چپ آغاز کند از سمت راست آغاز خواهد کرد. به‌طور مثال list1[-2] عنصر 1398 را نشان خواهد داد.

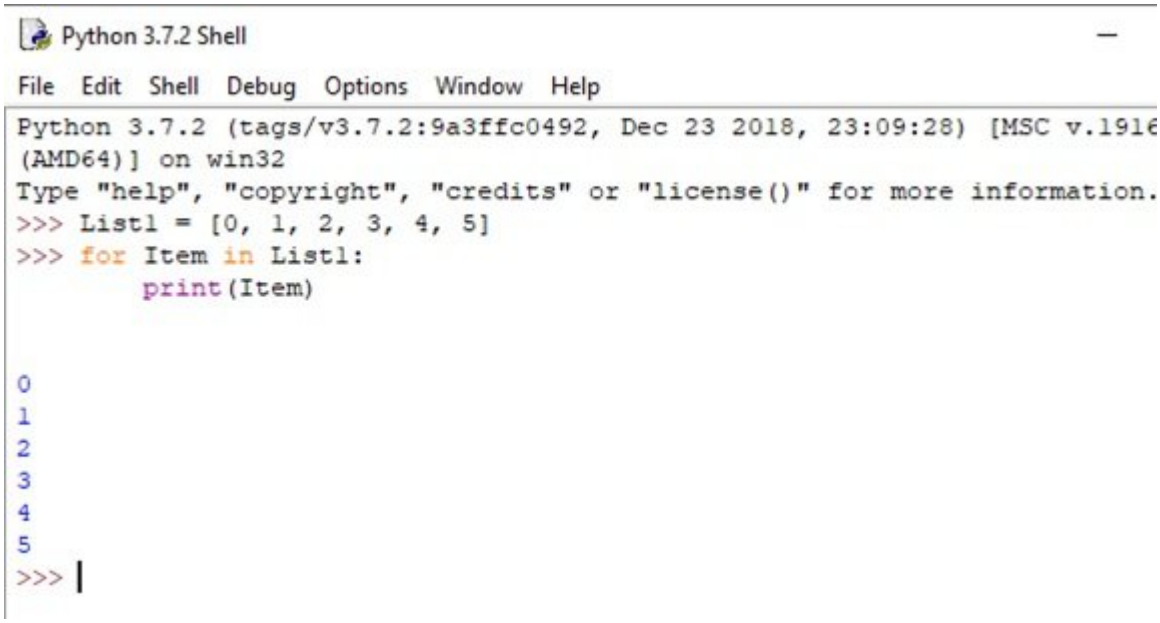
شمارش مقادیر فهرست از طریق حلقه for

برای آن‌که فرآیند خواندن همه عناصر درون یک فهرست به شکل خودکار انجام شود، باید از یک حلقه برای خواندن عناصر درون یک فهرست استفاده کنید. ساده‌ترین راه به‌کارگیری حلقه for است. برای روشن شدن بحث به مثال زیر دقت کنید:

IDLE.1 را باز کرده و قطعه کد زیر را درون پنجره اصلی تایپ کنید.

```
List1 = [0, 1, 2, 3, 4, 5]

for Item in List1:

print(Item)
```

قطعه کد بالا با ساخت فهرستی از مقادیر عددی آغاز می‌شود. در ادامه از حلقه for برای به دست آوردن هر عنصر داده‌ای و چاپ مقادیر روی صفحه استفاده شده است. تصویر زیر خروجی قطعه کد بالا را نشان می‌دهد.



ویرایش فهرست‌ها

پایتون به شما اجازه می‌دهد هر زمان نیاز داشتید مقادیر درون فهرست‌ها را ویرایش کنید. ویرایش یک فهرست به معنای اضافه، حذف یا تغییر عنصر خاصی درون یک فهرست است. برای انجام هر یک از این عملیات شما ابتدا باید به یک موجودیت دسترسی داشته باشید. توابعی که برای این کار در اختیارتان قرار دارد به شرح زیر است:

Append: یک موجودیت جدید به انتهای فهرست اضافه می‌کند.

Clear: همه عناصر درون فهرست را پاک می‌کند.

Copy: یک کپی از فهرست جاری ایجاد می‌کند.

Extend: عناصری به فهرست جاری اضافه می‌کند.

Insert یک موجودیت جدید در مکان تعیین شده در فهرست اضافه می‌کند.

Pop: یک عنصر از انتهای فهرست را حذف می‌کند.

Remove: عنصری که مکان آن مشخص شده است را پاک می‌کند.

برای آشنایی با نحوه عملکرد دستورات زیر به قطعه کد زیر دقت کنید.

IDLE.1 را باز کرده، دستور زیر را تایپ کرده و کلید اینتر را فشار دهید.

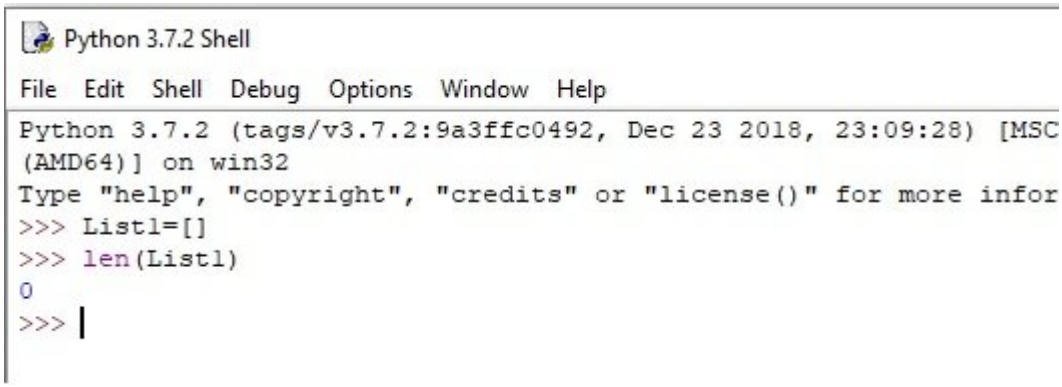
```
List1 =[]
```

پایتون فهرستی به نام List1 ایجاد می‌کند. در این تعریف شما یک فهرست خالی ساخته‌اید. List1 شامل هیچ عنصری نیست. شما می‌توانید یک فهرست خالی ایجاد کرده و در ادامه آن‌را با اطلاعات پر کنید.

3. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

```
Len(List1)
```

تابع len خروجی 0 را نشان می‌دهد، زیرا هنوز هیچ عنصری درون فهرست قرار ندارد.



4. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.append(1)

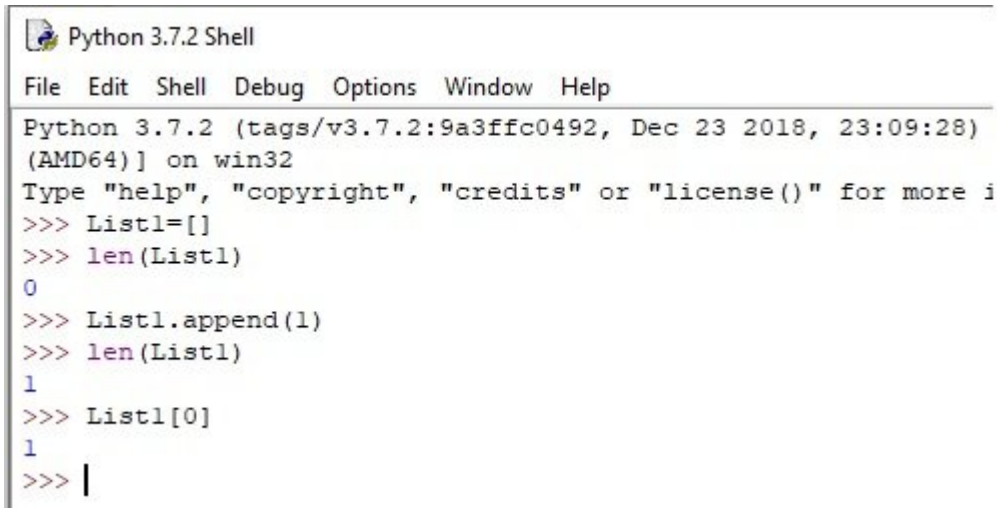
5. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید. تابع فوق اکنون اندازه 1 را باز می‌گرداند.

len(List1)

6. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1[0]

اکنون باید مقداری که در مکان صفرم فهرست قرار گرفته است را مشاهده کنید.

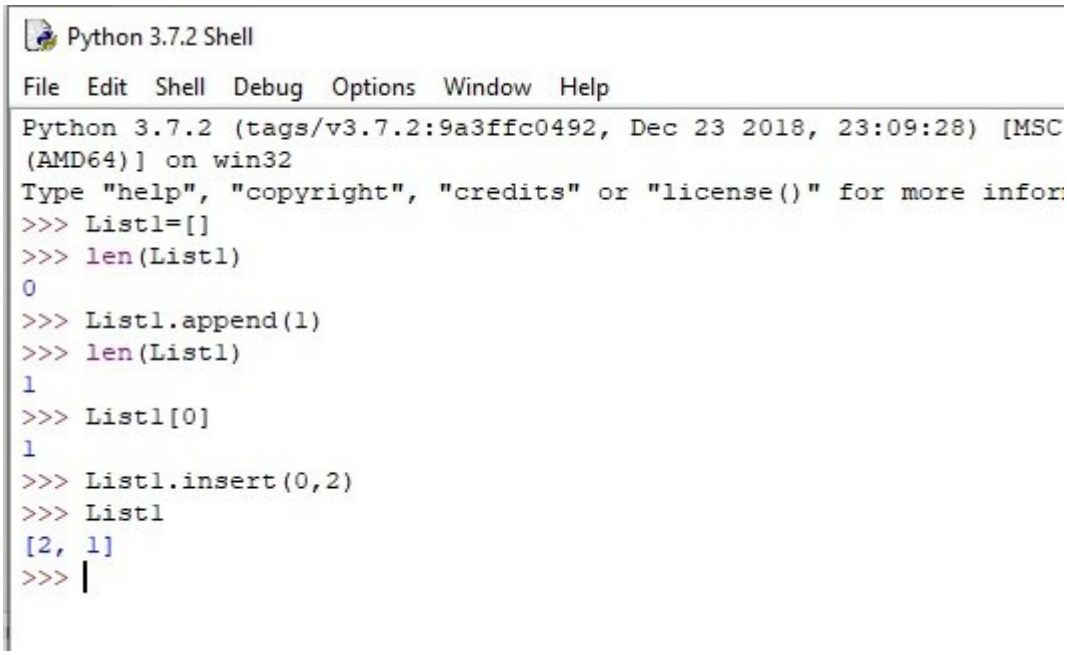


7. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.insert(0,2)

تابع insert دو آرگومان دارد. اولین آرگومان اندیسی است که مکان ذخیره کردن مقدار را مشخص کند که در مثال ما مکان 0 است. آرگومان دوم به عنصری که قرار است به فهرست اضافه شود، اشاره دارد که در مثال ما مقدار 2 قرار است اضافه شود.

8. List1 را تایپ کرده و کلید اینتر را فشار دهید.



مشاهده می‌کنید که **پایتون** عنصری را به List1 اضافه کرده است.

9. فرمان زیر ار تایپ کرده و کلید اینتر را فشار دهید.

List2=List1.copy()

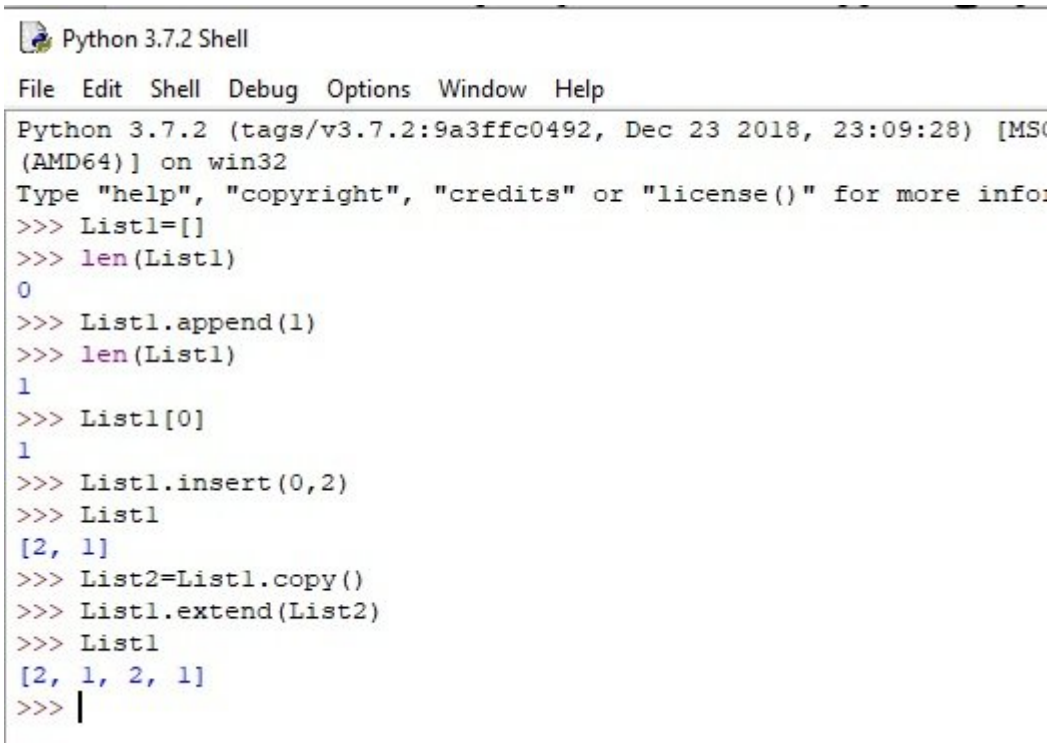
با اجرای این فرمان درون متغیر List2 یک کپی از List1 قرار می‌گیرد. فرآیند کپی کردن اغلب زمانی که قرار است یک نسخه موقت از یک فهرست ایجاد شود استفاده شده و پس از اتمام کار پاک می‌شود.

10. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.extend(List2)

پایتون همه عناصری List2 را به انتهای List1 اضافه می‌کند.

List1.11 را تایپ کرده و کلید اینتر را فشار دهید.



مشاهده می‌کنید که فرآیند کپی به درستی انجام شده است. List1 اکنون شامل مقادیر 2,1,2,1 است.

12. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.pop()

پایتون مقدار 1 را نشان می‌دهد. مقدار 1 اضافه شده به انتهای فهرست از آن درون آن حذف می‌شود.

13. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.remove()

در این زمان **پایتون** آیتم واقع در عنصر 1 را پاک می‌کند. برعکس تابع pop، تابع remove هیچ مقداری مبنی بر حذف را نشان نمی‌دهد.

14. فرمان زیر را تایپ کرده و کلید اینتر را فشار دهید.

List1.clear()

با اجرای این فرمان همه مقادیر درون فهرست پاک می‌شوند. اکنون اگر از فرمان len(List1) استفاده کنید، مقدار برگشتی 0 را مشاهده خواهید کرد که به معنای خالی بودن فهرست است.

جست‌وجوی فهرست‌ها

ویرایش یک فهرست کار ساده‌ای نیست، به ویژه زمانی که ندانید یک فهرست از چه نوع اطلاعاتی ساخته شده است. به همین دلیل مهم است که با نحوه جست‌وجوی یک فهرست آشنا باشید. قطعه کد زیر نحوه جست‌وجو در فهرست‌ها را نشان می‌دهد:

1. IDLE را باز کرده و از منوی File گزینه New File را انتخاب کنید.

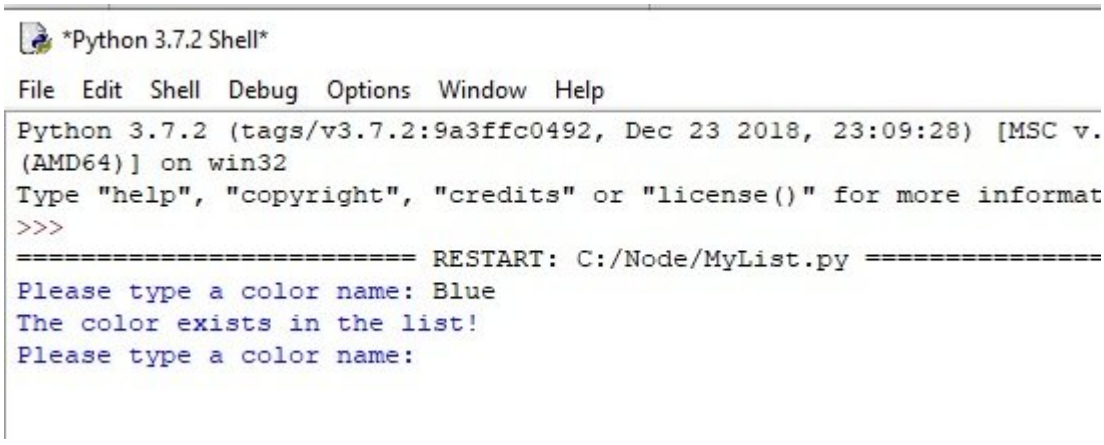
2. در پنجره جدید قطعه کد زیر را وارد کنید.

```
Colors = ["Red", "Orange", "Yellow", "Green", "Blue"]

ColorSelect = ""

while str.upper(ColorSelect) != "QUIT":

    ColorSelect = input("Please type a color name: ")

    if (Colors.count(ColorSelect) >= 1):

        print("The color exists in the list!")

    elif (str.upper(ColorSelect) != "QUIT"):

        print("The list doesn't contain the color.")
```

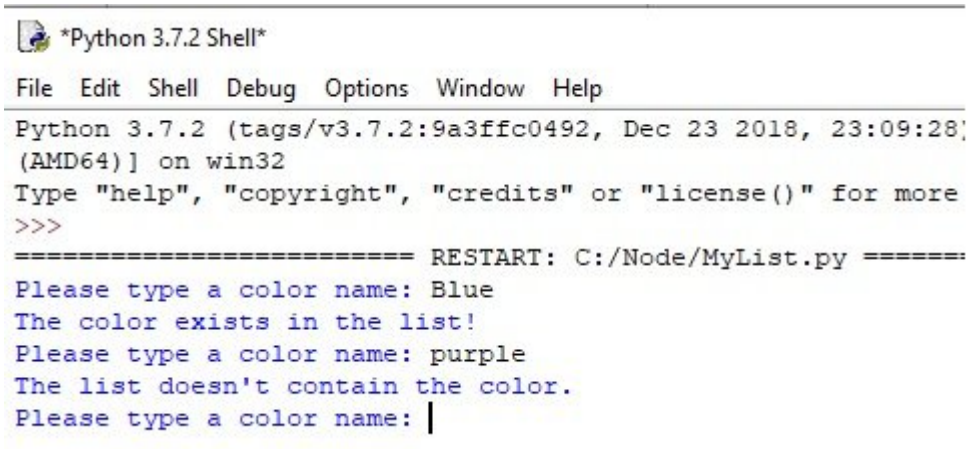
در قطعه کد بالا کار با ساخت فهرستی به نام colors شروع می‌شود که نام رنگ‌ها را نگه‌داری می‌کند. در ادامه متغیری به نام ColorSelect ایجاد شده که برای نگه‌داری نام رنگی که از کاربر به عنوان ورودی درخواست خواهد شد استفاده می‌شود. برنامه در ادامه وارد حلقه where شده، جایی که از کاربر درخواست می‌شود برای تغییر نام یک رنگ واژه‌ای را تایپ کند. واژه تایپ شده از سوی کاربر درون متغیر ColorSelect قرار می‌گیرد. دقت کنید اگر کاربر واژه QUIT را وارد کند برنامه از حلقه خارج شده و پایان پیدا می‌کند. هر زمان کاربر نام یک رنگ را وارد می‌کند، برنامه از فهرست درخواست می‌کند تا تعداد رنگ‌ها را شمارش کند. زمانی که مقدار برابر یا بیشتر از یک بود، به معنای آن است که فهرست حاوی یک رنگ است. در این حالت پیغامی روی صفحه ظاهر می‌شود. به عبارت دیگر، زمانی که فهرست شامل هیچ نام رنگی نباشد، یک پیغام پیش‌فرض روی صفحه به کاربر نشان داده می‌شود. توجه کنید که چگونه این برنامه از دستور elif برای بررسی مقدار متغیر ColorSelect برای خروج از حلقه استفاده می‌کند. به‌کارگیری elif تضمین می‌دهد که برنامه زمانی که واژه quit تایپ می‌شود پیغامی را روی صفحه نشان نمی‌دهد. شما نیز زمانی که برنامه‌ای می‌سازید که قرار است از کاربر ورودی دریافت کند و احتمال دارد کاربر ورودی اشتباهی را وارد کند باید از همین تکنیک استفاده کنید.

3. برنامه را ذخیره کرده و از منوی Run گزینه Run Module را انتخاب کنید.

4. واژه Blue را تایپ کرده و کلید اینتر را فشار دهید.



5. واژه Purple را تایپ کرده و کلید اینتر را فشار دهید.



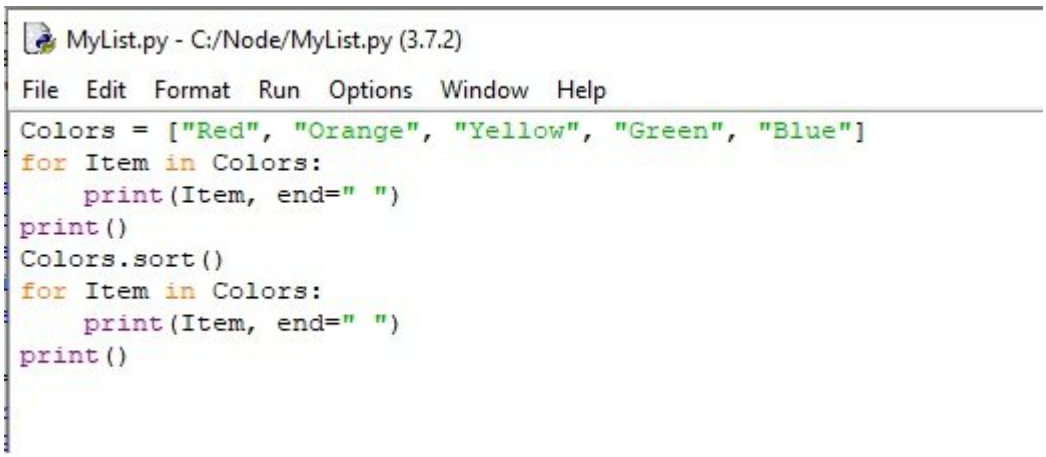
6. واژه Quit را تایپ کرده و کلید اینتر را فشار دهید. دقت کنید برنامه بدون نمایش پیغامی خاتمه پیدا می‌کند.

مرتب‌سازی فهرست‌ها

کامپیوترها فارغ از هرگونه مرتب‌سازی اطلاعات را درون فهرست‌ها قرار می‌دهند. اما زمانی که یک فهرست طولانی شود و قصد جست‌وجو در یک فهرست را داشته باشید، این تکنیک مرتب‌سازی است که کمک می‌کند به سرعت عنصری که به دنبال آن هستید را درون یک فهرست پیدا کنید. برای آشنایی با تکنیک مرتب‌سازی عناصر درون یک فهرست به مثال زیر دقت کنید:

IDLE.1 را باز کرده و از منوی File گزینه New File را انتخاب کنید.

2. برنامه را ذخیره کرده و از منوی Run گزینه New Module را انتخاب کنید. خروجی این برنامه همانند شکل زیر است.



قطعه کد بالا با ساخت یک آرایه از رنگ‌ها به نام Colors کار خود را آغاز می‌کند. رنگ‌ها به صورت غیر مرتب در فهرست وارد شده‌اند. در ادامه قطعه کد بالا رنگ‌ها را به همان ترتیبی که درون فهرست قرار دارند چاپ می‌کند. در قطعه کد بالا دقت کنید ما از عبارت end= در فرمان چاپ استفاده کردیم تا مطمئن شویم رنگ‌ها همگی در یک خط چاپ می‌شوند. پس از چاپ رنگ‌ها از تابع sort برای مرتب‌سازی عناصر فهرست استفاده کردیم. پس از فراخوانی این تابع دومرتبه از حلقه for برای چاپ مقادیر استفاده کردیم. اکنون مقادیر به شکل مرتب شده چاپ می‌شوند.

برخی موارد مجبور هستید فرآیند مرتب‌سازی فهرست‌ها را به شکل معکوس انجام دهید. برای این‌کار **پایتون** تابع reverse را ارائه کرده است. برنامه زیر را در **پایتون** وارد کرده و اجرا کنید تا تفاوت دو قطعه کد را مشاهده کنید.

```
Colors = ["Red", "Orange", "Yellow", "Green", "Blue"]
```

```
for Item in Colors:
    print(Item, end=" ")
print()
Colors.sort()
Colors.revese()
for Item in Colors:
    print(Item, end=" ")
print()
```

کار با شی شمارنده (Counter)

گاهی اوقات تنها یک منبع داده‌ای دارید و به سادگی می‌دانید چه اتفاقاتی درون منبع رخ می‌دهد. (شبیه به فهرستی که ایجاد می‌کنید و عناصر مشخصی درون آن قرار دارند.). زمانی که فهرست کوتاهی دارید، بدون مشکل عناصر داخل فهرست قابل شمارش هستند. اما زمانی که یک فهرست بلندی در اختیار دارید، شمارش دقیق عناصر کار خیلی مشکلی است. به‌طور مثال، رمان طولانی همچون جنگ و صلح را تصور کنید که قصد شمارش کلماتی را دارید که در آن رمان آورده شده‌اند. این کار بدون استفاده از کامپیوترها امکان‌پذیر نخواهد بود. شی Counter به شما اجازه می‌دهد تا عناصر را به سرعت شمارش کنید. علاوه بر این، به‌کارگیری این شی ساده است. این شی کاربردهای مختلفی دارد که در این مقاله با نحوه استفاده از آن روی فهرست‌ها آشنا می‌شوید. در مثال زیر، فهرستی ایجاد با عناصر تکراری ایجاد می‌کنیم و سپس تعداد عناصر درون فهرست را شمارش خواهیم کرد.

1. IDLE را باز کرده و از منوی File گزینه New File را انتخاب کنید.

2. قطعه کد زیر را درون پنجره جدید وارد کنید.

```
from collections import Counter
MyList = [1, 2, 3, 4, 1, 2, 3, 1, 2, 1, 5]
ListCount = Counter(MyList)
print(ListCount)
for ThisItem in ListCount.items():
    print("Item: ", ThisItem[0]," Appears: ", ThisItem[1])
    print("The value 1 appears {0} times.".format(ListCount.get(1)))
```

برای آن‌که از شی Counter استفاده کنید، ابتدا باید ماژول collections را درون برنامه خود وارد کنید. این ماژول دنباله‌ها و مجموعه‌های دیگر نیز درون این ماژول قرار دارند. قطعه کد بالا با ساخت یک فهرست به نام MyList آغاز می‌شود. در ادامه عناصر عددی تکراری درون آن وارد می‌شود. فهرست ایجاد شده درون یک شی Counter به نام ListCount قرار می‌گیرد. شی Counter به روش‌های مختلفی قابل ساخت است، اما روش فوق ساده‌تر از سایر روش‌ها است. دقت کنید شی Counter و فهرست هیچ‌گونه ارتباطی با یکدیگر ندارند. زمانی که محتوای یک فهرست را تغییر می‌دهید، مجبور هستید شی Counter را دومرتبه ایجاد کنید، زیرا به‌طور خودکار تغییرات در شی اعمال نخواهند شد. ساده‌ترین راهکار برای این منظور فراخوانی متد clear و سپس فراخوانی متد update است تا شی

counter داده‌های جدید را دریافت کند. قطعه کد بالا ListCount را به روش‌های مختلفی چاپ می‌کند. اولین خروجی Counter است که بدون هیچ‌گونه دستکاری چاپ می‌شود. دومین خروجی عناصر منحصر به فرد MyList را با تعداد تکرارهای هر عنصر چاپ می‌کند.برای به دست آوردن یک عنصر و تعداد تکرارهایی که ظاهر می‌شود، شما باید تابع Item را فراخوانی کنید. سرانجام مثال نشان می‌دهد که چگونه برای به دست آوردن تعداد عناصر از تابع get استفاده کنید.

3. برنامه را ذخیره کرده و از منوی Run گزینه New Module را انتخاب کنید.

```
========================== RESTART: C:/Node/MyList.py ===========
Counter({1: 4, 2: 3, 3: 2, 4: 1, 5: 1})
Item:  1  Appears:  4
The value 1 appears 4 times.
Item:  2  Appears:  3
The value 1 appears 4 times.
Item:  3  Appears:  2
The value 1 appears 4 times.
Item:  4  Appears:  1
The value 1 appears 4 times.
Item:  5  Appears:  1
The value 1 appears 4 times.
>>>
```

دقت کنید اطلاعاتی که درون counter قرار می‌گیرند در قالب جفت مقدار و کلید ذخیره می‌شوند. هدف ما در این قطعه کد این بود که نشان دهیم چگونه می‌توانید عناصر درون یک فهرست را پیدا و شمارش کنید.

در نهایت

در این سری از آموزش‌های مقدماتی **پایتون** سعی کردیم، مباحث اصلی و کلیدی برنامه‌نویسی به زبان **پایتون** را نشان دهیم. بدیهی است، مباحث دیگری همچون کلاس‌ها، کار با سایر مجموعه،ها، مرتب‌سازی داده‌ها در فایل‌ها، ارسال ایمیل، کار با منابع پیشرفته و.... نیز وجود داشت که امیدواریم در سری‌های بعدی این آموزش آن‌ها را پی بگیریم. از همه خوانندگانی که با دنبال کردن این آموزش و ارایه دیدگاه‌های خود ما را یاری کردند سپاسگزاری می‌کنیم.

تاریخ انتشار:
23 اسفند 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/programming/14800/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86-python-%D8%B3%D8%A7%D8%AE%D8%AA%D8%8C-%D9%88%DB%8C%D8%B1%D8%A7%DB%8C%D8%B4-%D9%88-%D9%85%D8%AF%DB%8C%D8%B1%DB%8C%D8%AA-%D9%81%D9%87%D8%B1%D8%B3%D8%AA%E2%80%8C%D9%87%D8%A7-%D8%AF%D8%B1-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86