آموزش رایگان پایتون (python)- مدیریت خطاها به شکل تودرتو، فراخوانی خطاها، بلوک finally

در شماره گذشته آموزش پایتون یاد گرفتیم که چگونه از آرگومان‌های استثناءها برای مدیریت بهتر خطاها استفاده کنیم. همچنین با نحوه مدیریت خطاها از طریق بلوک‌های چندگانه و انفرادی except آشنا شدیم. در این شماره مبحث فوق را ادامه خواهیم داد.

برای مطالعه بخش نوزدهم آموزش رایگان پایتون اینجا کلیک کنید

بلوک‌های مدیریت خطای تودرتو

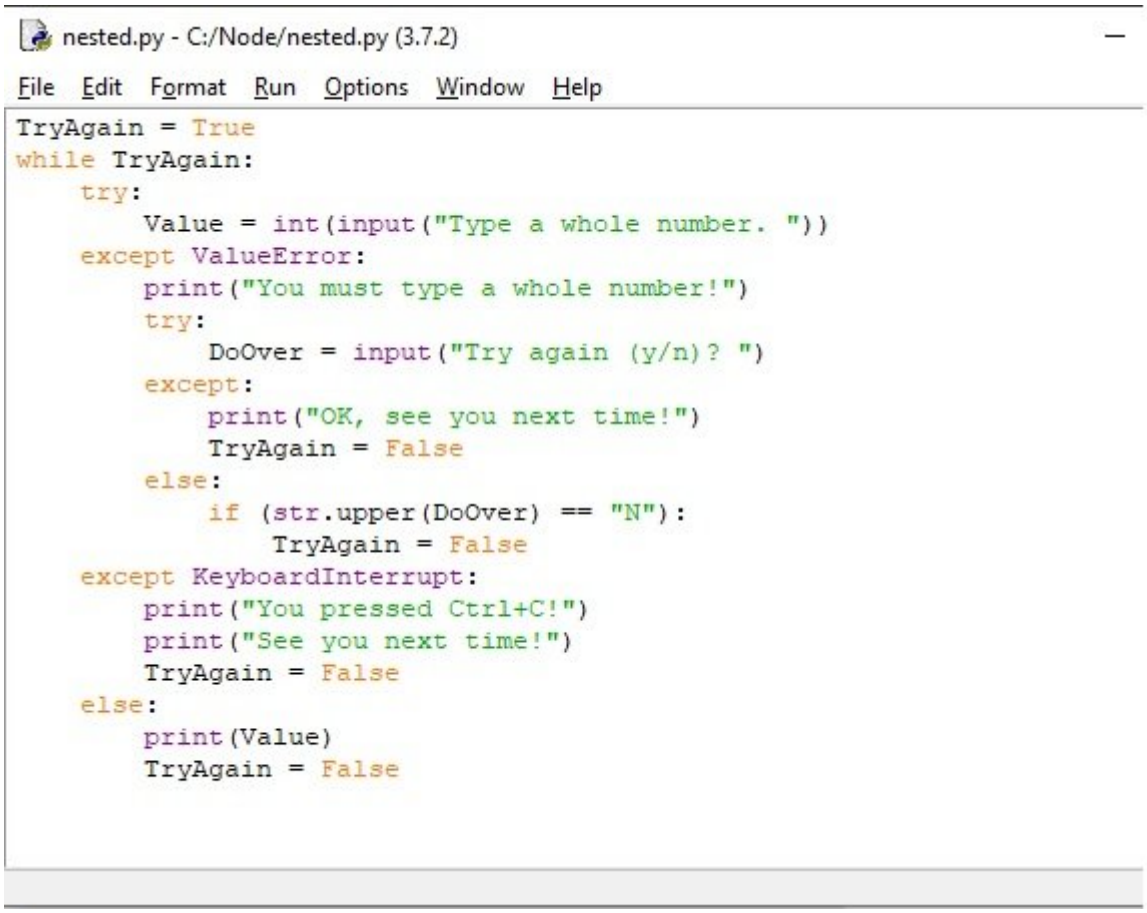
گاهی اوقات مجبور هستید یک روتین مدیریت خطا را درون بلوک دیگری قرار دهید. زمانی که از روتین‌های مدیریت خطاها به شکل تودرتو استفاده می‌کنید، پایتون ابتدا به سراغ داخلی‌ترین روتین رفته و پس از آن به سراغ روتین‌های قبل‌تر از آن خواهد رفت. پایتون به شما اجازه می‌دهد روتین‌های مدیریت خطاهای مطابق با نیاز خود را به شکل تودرتو استفاده کنید. اما چرا باید از روتین‌های مدیریت خطا به شکل تودرتو استفاده کرد؟ برخی موارد شما می‌خواهید اطلاعات ورودی از کاربر دریافت کرده و اطلاعات را درون حلقه‌ای استفاده کنید. پیش از آن‌که اطلاعات درون حلقه‌ای قرار گیرد، ابتدا باید از اصالت آن‌ها اطمینان حاصل کنید. اما چگونه می‌توانیم از بلوک‌های مدیریت خطاها به شکل متداخل استفاده کنیم؟ به مثال زیر دقت کنید:

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.

```
TryAgain= True

while TryAgain:

    try:

        Value=int(input("Type a whole number. "))

    except ValueError:
```

```
        print("You must type a whole number!")


        try:
            DoOver=input("Try again (y/n)? ")
        except:
                print("OK see you next time!")
                TryAgain=False
        else:
            if (str.uppe(DoOver) == "N"):
                TryAgain=False


except KeyboardInterrupt:
            print("You pressed Ctrl+C")
            print("see you next time!")
            TryAgain=False
else:
            print(Value)
            TryAgain=False
```

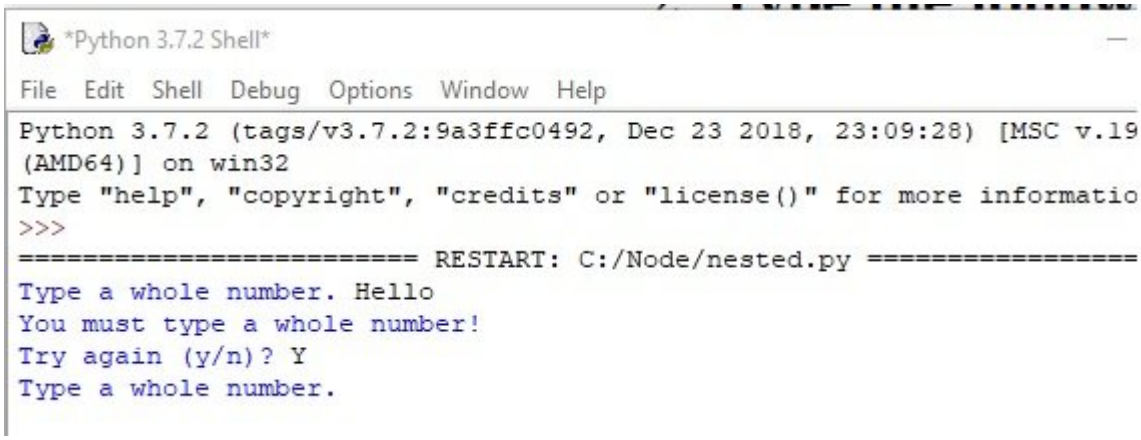


قطعه کد بالا با یک حلقه شروع به کار می‌کند. (در مقاله آینده به سراغ مبحث حلقه‌سازی در پایتون خواهیم رفت.) در حال حاضر این نکته را بدانید که حلقه‌ها برای انجام یکسری وظایف مشابه استفاده می‌شود. برنامه‌های کاربردی در اغلب موارد حلقه‌ها را به شکلی که در بالا مشاهده می‌کنید استفاده می‌کنند، زیرا در یک برنامه کاربری به ویژه زمانی که قرار است یک ورودی از کاربر دریافت شود، حلقه‌ها مادامی که کاربر مقدار درستی را وارد نکند تکرار می‌شوند. حلقه بالا الگویی ساده داشته و برای تکرار مجموعه‌ای از دستورات مادامی که ورودی کاربر به اجرای حلقه پایان ندهد تکرار می‌شوند. زمانی که این حلفه اجرا می‌شود، برنامه از کاربر درخواست می‌کند که یک عدد کامل را وارد کند. این عدد می‌تواند هر مقدار صحیحی باشد. اگر کاربر یک مقدار غیر صحیح را وارد کند یا کلیدهای Ctrl+C یا هر کلیدی که باعث به وجود آمدن وقفه‌ای شود را فشار دهد، کدی که برای مدیریت خطا تعریف شده اجرا شده و به این مسئله رسیدگی می‌کند. اما اگر کاربر مقدار صحیحی را وارد کند، برنامه مقدار وارد شده را چاپ کرده و مقدار متغیر TryAgain را به False تنظیم می‌کند تا اجرای حلقه پایان بپذیرد. خطای ValueError زمانی که کاربر اشتباهی انجام دهد فعال خواهد شد. زیرا شما دو نکته را نمی‌دانید. اول آن‌که کاربر مقدار صحیحی را وارد خواهد کرد یا خیر و دوم آن‌که کاربر دوست دارد برنامه ادامه پیدا کند یا به اجرای برنامه خاتمه دهد. به همین دلیل است که از او درباره ادامه کار سوال می‌کنید. البته دقت کنید اگر کاربر بیش از یک ورودی را وارد کند، خطای دیگری رخ خواهد داد. بلوک try-except داخلی به منظور مدیریت ورودی دوم کاربر استفاده شده‌اند. دقت کنید در بلوک دوم ما از تابع str.upper() استفاده کردیم تا بتوانیم ورودی کاربر که یک کاراکتر است را برای هر دو حالت بزرگ و کوچک y یا Y پردازش کنیم. زمانی که از کاربر درخواست می‌کنید کاراکتری را وارد کند، ایده خوبی است که هر دو حالت بزرگ و کوچک کاراکترها یا رشته‌ها را مقایسه کنید تا مدیریت خطاها به سادگی امکان‌پذیر باشد.

3. برنامه را ذخیره کرده و از منوی run گزینه Run Module را انتخاب کنید. برنامه از کاربر درخواست می‌کند تا یک عدد کامل را وارد کند.

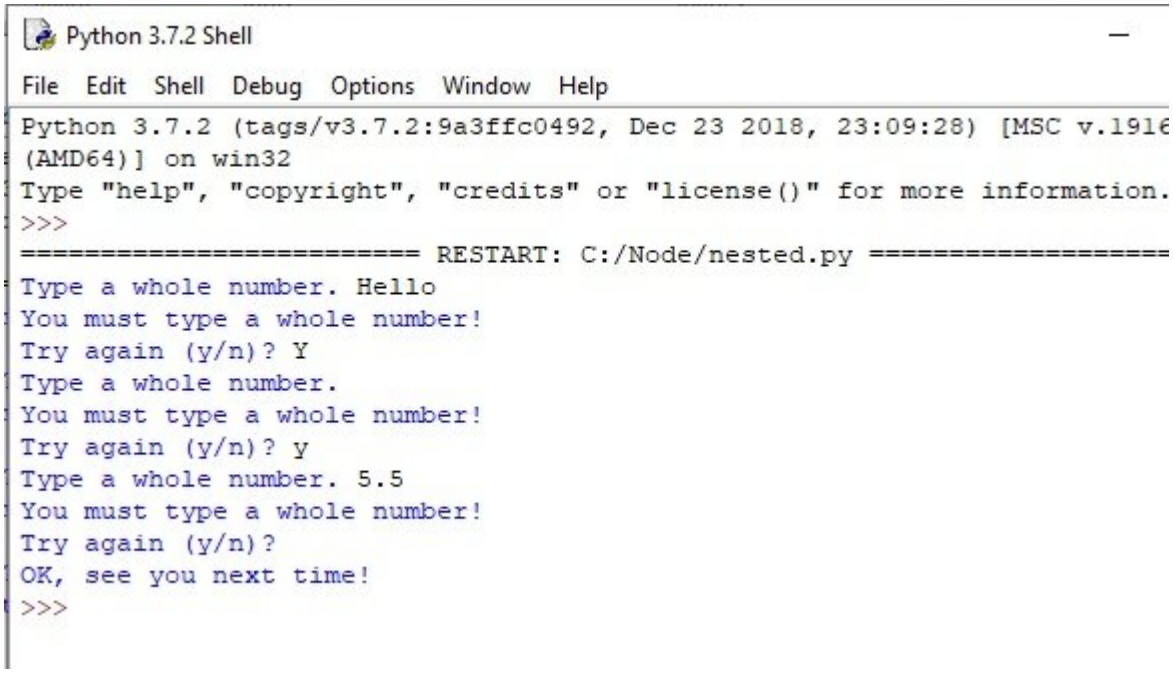
4. Hello را تایپ کرده و کلید اینتر را فشار دهید. برنامه یک خطای مدیریت شده نشان داده و برعکس برنامه‌هایی که تاکنون نوشته‌اید از شما سوال می‌کند که آیا دوست دارید دومرتبه مقدار دیگری را وارد کنید. این سوال مجدد به این دلیل ظاهر می‌شود که ما از حلقه While استفاده کرده‌ایم.

5. کلید Y را فشار داده تا برنامه دومرتبه از شما درخواست کند عدد کاملی را وارد کنید.



6. عدد 5.5 را تایپ کرده و کلید اینتر را فشار دهید. برنامه دومرتبه پیغام خطای کنترل شده‌ای نشان داده و از شما سوال می‌کند که دوست دارید دومرتبه عددی را وارد کنید.

7. کلیدهای Ctrl+C یا هر کلیدی را برای ساخت یک وقفه فشار دهید. دقت کنید پیغام خطای مدیریت شده‌ای که نشان داده می‌شود از درون حلقه مدیریت خطای داخلی ظاهر می‌شود. برنامه برای رسیدگی به خطا هیچ‌گاه به سراغ بلوک مدیریت خطای خارجی نخواهد رفت، زیرا بلوک اداره کننده خطای داخلی این کار را مدیریت می‌کند.



اجرای استثناءها

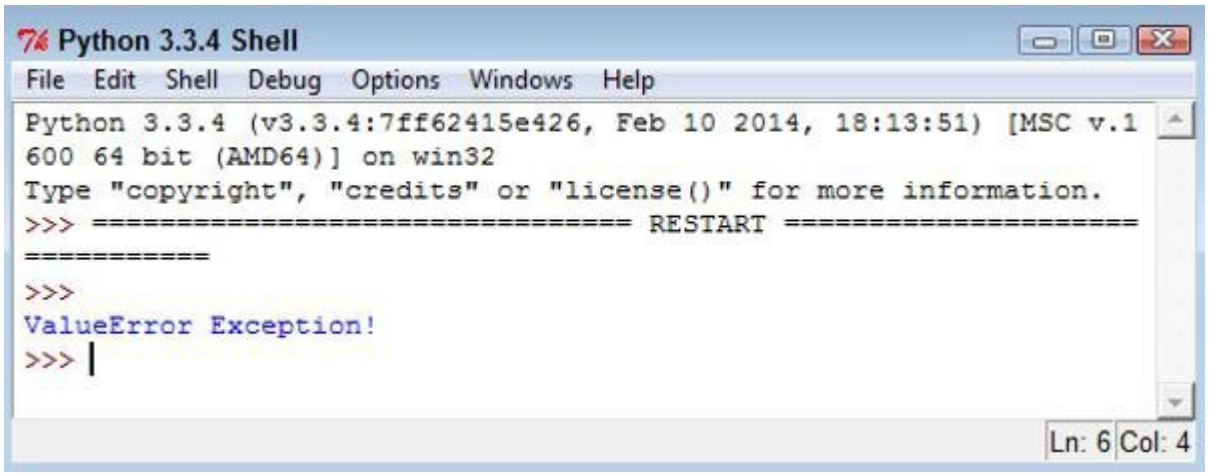
در چند شماره گذشته به مثال‌هایی اشاره کردیم که همگی واکنشی به یک اشتباه فاحش بودند. اشتباهاتی که ممکن است از سوی هر کاربری رخ دهد. زمانی که مشکلی برای یک برنامه کاربردی رخ می‌دهد، رخدادی برای واکنش به آن خطا اجرا می‌شود. با این حال، در برخی موارد شما شما به درستی نمی‌دانید در زمان اجرای یک برنامه زمانی که خطایی رخ می‌دهد چگونه باید آن‌را مدیریت کنید. ممکن است در سطح خاصی موفق نشوید یک خطا را به درستی مدیریت کرده و ترجیح می‌دهید که این خطا در سطح دیگری مدیریت شود. به‌طور خلاصه، در برخی موارد، مجبور است خود یک خطا را به شیوه دستی فراخوانی کنید. قطعه کد زیر نشان می‌دهد چگونه در برنامه کاربردی خود یک خطا فراخوانی کرده و به آن پاسخ دهید.

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

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

```
try:
    raise ValueError
except ValueError:
    print("ValueError Exception!")
```

شما ممکن است در دنیای واقعی یک چنین کدی را هیچ‌گاه ننویسید، اما این قطعه کد به شما نشان می‌دهد که چگونه یک استثناء را به شیوه دستی و در ابتدایی‌ترین سطح خود فراخوانی کنید. در قطعه کد بالا، دستور فراخوانی کننده خطا درون بلوک try...except قرار گرفته است. در یک فراخوانی ساده، نام استثناء همراه با کلمه کلیدی raise مشخص می‌شود. در فراخوانی دستی شما می‌توانید آرگومان‌ها را به عنوان بخشی از خروجی که قرار است اطلاعات اضافی ارائه کند استفاده کنید. دقت کنید که این بلوک try....except فاقد آرگومان است، زیرا پس از فراخوانی قرار نیست کار دیگری انجام شود. شما به ندرت از فراخوانی دستی استفاده می‌کنید، اما باید با نحوه فراخوانی دستی آشنا باشید. دقت کنید در چنین شرایطی به‌کارگیری بخش else جنبه اختیاری داشته و ضروری نیست. اما حداقل باید یک بلوک except را تعریف کنید. با اجرای برنامه خروجی زیر را مشاهده می‌کنید.



ارسال اطلاعات خطا برای یک فراخوانی‌کننده

پایتون یک مدیریت خطای فوق‌العاده انعطاف‌پذیر در اختیارتان قرار می‌دهد که اجازه می‌دهد اطلاعات خطا را برای یک فراخوانی‌کننده خطا ارسال کنید، فارغ از این‌که چه خطایی را استفاده کرده‌اید. به عبارت دقیق‌تر برای کدی که به شکل دستی خطایی را فراخوانی کرده، جزییات خطا را ارسال کنید. ارسال اطلاعات اضافی به شما کمک می‌کند در زمان نمایش یک خطای تولید شده اطلاعات اضافی را نمایش دهید که به شما یا فردی که روی یک پروژه کار می‌کند کمک می‌کنند اطلاعات بیشتری در مورد خطایی که رخ داده است به دست آورید. برای روشن شدن بحث به مثال زیر دقت کنید.

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

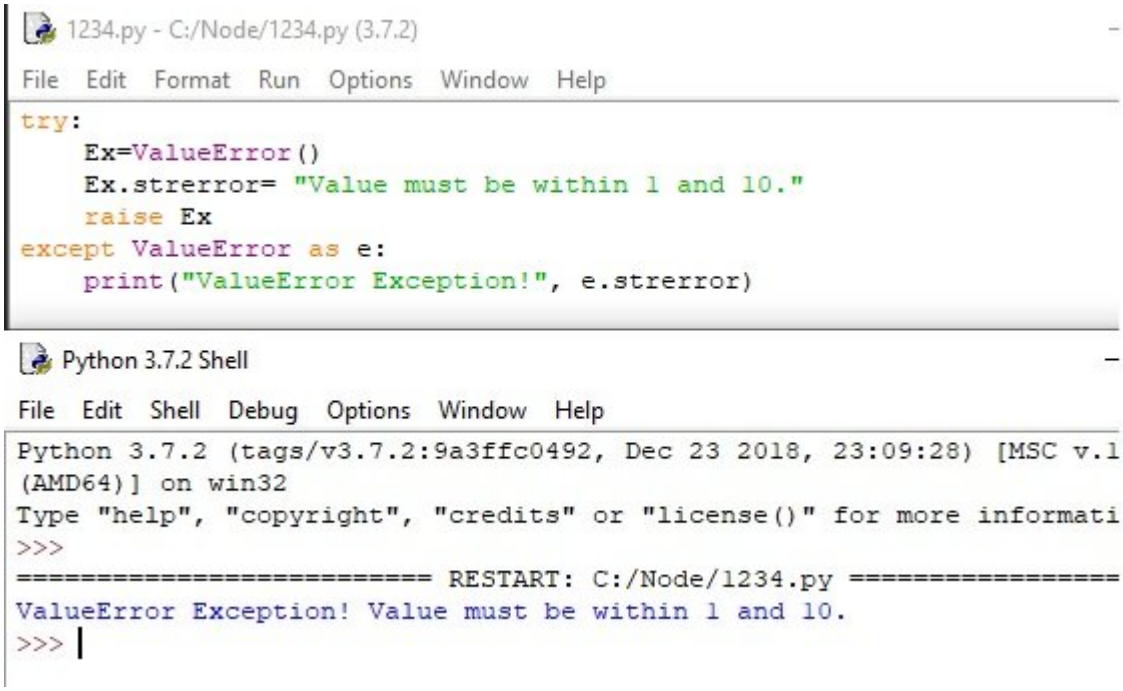
2. در پنجره جدید قطعه کد زیر را وارد کنید.

```
try:
    Ex=ValueError()
    Ex.strerror= "Value must be within 1 and 10."
```

```
    raise Ex

except ValueError as e:

    print("ValueError Exception!", e.strerror)
```

استثناء/خطای ValueError در حالت عادی خصلتی به نام strerror ندارد، اما شما می‌توانید به سادگی با تعریف و مقداردهی چنین خصلتی را به استثناء فوق اضافه کنید. زمانی که مثال فوق را اجرا می‌کنید، برنامه یک خطای مدیریت شده تولید کرده که بلوک except برای اداره کردن خطا فراخوانی می‌شود، اما برای دسترسی به اطلاعات خطا باید از خصلت e استفاده کنید. در ادامه فرمان print از متغیر e و خصلت strerror برای نمایش خطای مدیریت شده استفاده می‌کند.

3. برنامه را ذخیره کرده و از منوی Run گزینه Run Module را انتخاب کنید. خروجی قطعه کد بالا در تصویر زیر نشان داده شده است.



تکنیک فوق از آن جهت کمک کننده است که اجازه می‌دهد در زمان فراخوانی خطاها به شیوه دستی اطلاعات اضافی و تکمیلی خود را اضافه کنید.

ساخت و به‌کارگیری استثناءهای سفارشی

پایتون به شما اجازه می‌دهد تا بلوک‌های مدیریت خطای سفارشی خاص خود را ایجاد کرده و از آن‌ها استفاده کنید. این مدل استثناءها انعطاف‌پذیری بالایی در اختیارتان قرار می‌دهد و حتا اجازه می‌دهد استثناءها را مطابق با نیاز کاری خود ویرایش کنید. در پاراگراف پیشین به شما نشان دادیم که چگونه می‌توانید اطلاعاتی را برای یک فراخوانی کننده استثناء ارسال کرده و استثناء ValueError را همراه با داده‌های اضافی ویرایش کنید. با این حال، گاهی اوقات، مجبور هستید یک استثناء سفارشی را ایجاد کنید، زیرا برنامه شما ممکن است خطای خاصی را تولید کند. به‌طور مثال، نام یک استثناء ممکن است به بیننده اطلاع مشخص و روشنی درباره خطا ندهد یا زمانی که با بانک‌های اطلاعاتی و سرویس‌های ویژه کار می‌کنید یک مدیریت خطای مشخص در اختیارتان قرار نداشته باشد. ما اکنون به سراغ مبحث پیاده‌سازی و به‌کارگیری استثناءهای سفارشی نمی‌رویم تا مفهوم کلاس‌ها را بررسی کنیم.

به‌کارگیری بلوک finally

به‌طور معمول، شما در نظر دارید هر استثنایی که ناگهان رخ می‌دهد را مدیریت کنید. با این‌حال، گاهی اوقات

نمی‌توانید همه مشکلات را درست کنید و برنامه شما به شکل ناگهانی خاتمه می‌یابد. در این نقطه، هدف شما برطرف کردن مشکلی است که باعث می‌شود کاربرانی که از برنامه شما استفاده می‌کنند داده‌های خود را از دست بدهند. بهترین راهکاری که برای مدیریت خطاها در اختیار شما قرار دارد، حفظ داده‌هایی است که ممکن است ناگهان از دست بروند. بلوک finally به شما اجازه می‌دهد یک استراتژی در زمان خرابی برنامه طراحی کنید. شما می‌توانید از بلوک finally برای اجرای دستوراتی که در آخرین لحظات اجرا خواهند شد استفاده کنید. به‌طور معمول بلوک finally شامل اطلاعات مختصری هستند که فقط دستورات حیاتی درون آن‌ها قرار می‌گیرد. بستن فایل‌هایی که باز هستند، خارج کردن کاربر از برنامه و سایر کارهایی که پیش از خاتمه برنامه انجام آن‌ها ضروری است در این بلوک قرار می‌گیرند. برای روشن شدن مطلب به مثال زیر دقت کنید:

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

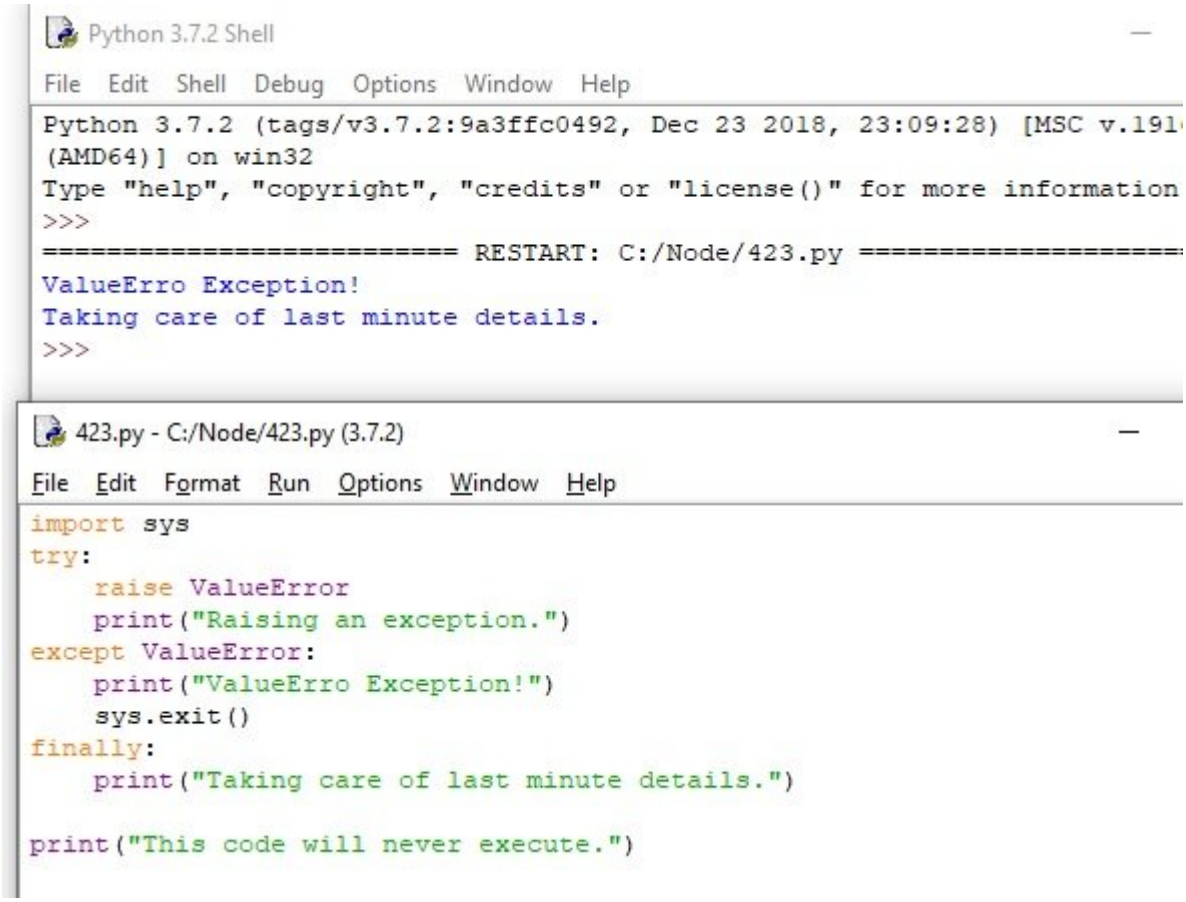
2. قطعه کد زیر را درون پنجره جدید وارد کنید.

```
import sys

try:

    raise ValueError

    print("Raising an exception.")

except ValueError:

    print("ValueErro Exception!")

    sys.exit()

finally:

    print("Taking care of last minute details.")



print("This code will never execute.")
```

قطعه کد بالا، استثناء ValueError را فراخوانی می‌کند. بلوک except زمانی که استثناء فوق رخ دهد اجرا می‌شود. فراخوانی فرمان sys.exit به معنای آن است که برنامه از همه بلوک‌هایی که برای مدیریت استثناءها طراحی شده خارج خواهد شد. در چنین شرایطی فرمان print که درون بلوک finally قرار دارد اجرا شده اما فرمان print انتهای برنامه هیچ‌گاه اجرا نخواهد شد.

دقت کنید که بلوک finally در هر شرایطی اجرا خواهد شد. به عبارت دیگر، مهم نیست برنامه شما استثنایی تولید کرده یا خیر. دستوراتی که درون بلوک finally قرار می‌گیرند در هر حالتی اجرا شده و به همین دلیل است که برنامه‌نویسان ترجیح می‌دهند دستورات مهم یک برنامه را پیش از خاتمه برنامه درون این بلوک قرار دهند.

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



در شماره آینده آموزش **پایتون** به سراغ مبحث حلقه‌ها خواهیم رفت.

تاریخ انتشار:
12 اسفند 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/programming/14726/%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-%D9%85%D8%AF%DB%8C%D8%B1%DB%8C%D8%AA-%D8%AE%D8%B7%D8%A7%D9%87%D8%A7-%D8%A8%D9%87-%D8%B4%DA%A9%D9%84-%D8%AA%D9%88%D8%AF%D8%B1%D8%AA%D9%88%D8%8C-%D9%81%D8%B1%D8%A7%D8%AE%D9%88%D8%A7%D9%86%DB%8C-%D8%AE%D8%B7%D8%A7%D9%87%D8%A7%D8%8C