آموزش رایگان پایتون (python)- آشنایی با مفهوم مدیریت خطاها در پایتون

در شماره گذشته آموزش پایتون با نحوه به‌کارگیری فرمان if...elif آشنا شدیم و به مثال‌های مختلفی اشاره کردیم که اجازه می‌دهند سناریوهای پیچیده تصمیم‌گیری را با استفاده از if...elif پیاده‌سازی کنید. در این شماره قصد داریم به سراغ مبحث مدیریت خطاها در پایتون برویم.

برای مطالعه بخش پانزدهم آموزش رایگان پایتون اینجا کلیک کنید

قبل از آن‌که به سراغ مبحث مدیریت خطاها برویم، اجازه دهید به نکته دیگری در ارتباط با عملگرهای شرطی اشاره کنیم. پایتون به شما اجازه می‌دهد برای ارزیابی یک مقدار یا عبارت از مدل دیگری از عملگرها استفاده کنید که به عملگرهای سه تایی شهرت دارند. عملگرهای سه تایی که به نام عملگرهای شرطی نیز شناخته می‌شوند، عملگرهایی هستند که یک مقدار را بر مبنای یک وضعیت درست یا غلط ارزیابی می‌کنند. این عملگرها به شما اجازه می‌دهند به جای چند دستور if-else از یک خط برای ارزیابی مقادیر استفاده کنید. ترکیب نحوی این فرمان به شرح زیر است:

[on_true] if [expression] else [on_false]

برای آن‌که با ترکیب نحوی بالا بهتر آشنا شوید، اجازه دهید با ذکر مثالی این مسئله را برای شما روشن کنیم. قطعه کد زیر نحوه به‌کارگیری عملگر سه‌تایی را نشان می‌دهد. در قطعه کد زیر اگر مقدار متغیر a کوچک‌تر از متغیر b باشد مقدار متغیر a درون متغیر min قرار گرفته و فرمان print مقدار موجود در متغیر min را چاپ می‌کند.

```
# Program to demonstrate conditional operator

a, b = 10, 20

# Copy value of a in min if a < b else copy b

min = a if a < b else b

  print(min)
```

در قطعه کد بالا مقادیر 10 و 20 به دو متغیر a و b اختصاص داده شده‌اند و در خط دوم بررسی شده است که اگر مقدار متغیر a کوچک‌تر از b بود، آن‌گاه مقدار متغیر a درون متغیر min قرار گیرد. تصویر زیر خروجی قطعه کد بالا را نشان می‌دهد.

مدیریت خطاها

اکثر کدهایی که درون برنامه‌های کاربردی نوشته می‌شوند ترکیبی پیچیده دارند. زمانی که برنامه شما ناگهان فریز شده و بدون دلیل متوقف می‌شود، نشان می‌دهد که برنامه شما دارای یک خطا است. خطاها در بیشتر موارد بدون آن‌که هیچ‌گونه نشانه‌ای داشته باشند ظاهر می‌شوند. یک محاسبه اشتباه روی مجموعه‌ای از اعداد باعث می‌شود خروجی اشتباهی که انتظارش را نداشتید دریافت کنید. خروجی اشتباه نه تنها ممکن است اطلاعات اشتباهی در اختیار کاربر قرار دهد، بلکه ممکن است ناگهان به اجرای یک برنامه خاتمه دهد. زبان‌های برنامه‌نویسی راهکاری در اختیار ما قرار داده‌اند که مانع از آن می‌شود که اجرای یک برنامه ناگهان خاتمه پیدا کند. زمانی که خطایی در یک برنامه رخ می‌دهد که مانع از آن می‌شود که یک برنامه روند طبیعی اجرای خود را پشت سر بگذارد در این حالت می‌گوییم استثنایی رخ داده است. استثناءها جزء جدایی‌ناپذیر دنیای برنامه‌نویسی هستند و ممکن است هر لحظه رخ دهند، پس اگر برنامه شما با استثناءیی روبرو شد، نگران باشید و فقط به فکر تشخیص و برطرف کردن آن باشید. فرآیند شناسایی و پردازش یک استثناء مدیریت استثناءها یا مدیریت خطا نام دارد. زمانی که خطا را تشخیص دادید، در مرحله بعد باید آن خطا را در بلوک مخصوصی (catch) به دام انداخته و اجازه ندهید اجرای یک برنامه را متوقف کند. به دام انداختن یا گرفتن خطا به معنای بررسی احتمال بروز خطا بوده و باید کدهای خاصی برای مواجه شدن با یک خطا در برنامه قرار گیرد. به‌طور مثال، فرض کنید باید محتوا یک فایل متنی را خوانده و اطلاعات آن‌را درون برنامه کاربردی خود وارد کنید. اگر فایل روی دیسک وجود داشته باشد، برنامه شما فایل را باز کرده، محتوای آن‌را خوانده و در برنامه استفاده می‌کند، اما اگر فایل روی دیسک وجود نداشته باشد، آن‌گاه برنامه شما یک استثناء تولید می‌کند. در چنین شرایطی برنامه شما باید آمادگی آن‌را داشته باشد که در صورت عدم وجود فایل چه تهمیداتی در نظر بگیرید. اما چگونه می‌توانیم بلوک‌های مدیریت خطاها را درون برنامه کاربردی خود وارد کنیم؟

گاهی اوقات کدهای درون یک برنامه باعث بروز خطا می‌شوند. زمانی که این اتفاق رخ دهد، برنامه شما در اصطلاح برنامه‌نویسان یک استثناء را پرتاپ می‌کند (throw an exception). در برخی از موارد تشخیص و شناسایی خطاها به راحتی امکان‌پذیر بوده و با یک مدیریت ساده و نمایش پیغام خطا مشکل برطرف می‌شود. در حالات دیگر شما مجبور هستید از تکنیک‌های پیشرفته خطایابی برای شناسایی مشکلات استفاده کنید. پایتون دارای اشیا ژنریک سفارشی قدرتمندی است که به شما اجازه می‌دهد به درستی علت بروز مشکلات را تشخیص دهید. به‌طور مثال، برنامه شما ممکن است به شکل ویژه و سفارشی از بانک‌های اطلاعاتی پشتیبانی کند، در این‌گونه مواقع و در زمان بروز مشکلات خاص **پایتون** نمی‌تواند از طریق اشیا عمومی تشخیص خطا به شما کمک کند و در نتیجه شما مجبور هستید خودتان از تکنیک‌ها و روش‌های موجود برای واکاوی مشکل استفاده کنید. نکته مهمی که در زمان مدیریت استثناءها به شکل محلی باید به آن دقت کنید، به مکان قرارگیری کدها و همچنین نحوه تعامل کدهای برنامه با چنین کدهایی اشاره دارد. به‌طور مثال یک تابع برای انجام پردازش‌های خود به ورودی نیاز دارد که این ورودی به درستی

دریافت نشده یا پردازش نشده است و در صورت ارسال این مقدار برای یک تابع یک استثنا به وجود خواهد آمد. در این حالت باید درون تابع دستورات مناسبی برای پاسخ‌گویی به این وضعیت قرار گیرد.

چرا زمانی که کدنویسی می‌کنید پایتون منظور شما را درک نمی‌کند؟

توسعه‌دهندگان اغلب ترجیح می‌دهند از زبان‌هایی استفاده کنند که به آن‌ها اجازه می‌دهد به بهترین شکل خطاها را شناسایی کرده و برطرف کنند. زمانی که دستورات و کدهای برنامه‌نویسی را تایپ می‌کنید، نه کامپیوتر و نه پایتون نمی‌دانند که منظور شما از تایپ کدها چیست. آن‌ها فقط دستورالعمل‌هایی که وارد کرده‌اید را بر مبنای منطق برنامه شما دنبال کرده و اجرا می‌کنند. به‌طور مثال زمانی که قصد دارید فایلی که درون یک پوشه قرار دارد را پاک کنید و برای این منظور از یک حلقه استفاده می‌کنید، باید برای **پایتون** به شکل دقیق مشخص کنید که قرار است چه فایلی باید پاک شود، در غیر این صورت ممکن است **پایتون** همه فایل‌های درون یک پوشه را پاک کند.

باگ چیست؟

هر زمان خطایی رخ دهد، مردم می‌گویند که برنامه دارای یک باگ است. باگ‌ها در حقیقت خطاهای کدنویسی هستند که شما می‌توانید با استفاده از دیباگر آن‌ها را برطرف کنید. یک دیباگر یک ابزار ویژه است که به شما اجازه می‌دهد بر روند اجرای یک برنامه کنترل کاملی داشته باشید و دستورات را خط به خط اجرا کرده، وضعیت متغیرها را بررسی کرده و به‌طور کلی یک برنامه را تجزیه و تحلیل کنید.

خطاها به دلایل مختلفی رخ می‌دهند، بی دقتی برنامه‌نویس در بررسی یک حالت خاص و ورودی‌های اشتباهی که از سوی کاربران وارد می‌شود از شایع‌ترین علل بروز خطا در یک برنامه هستند. دقت کنید که **پایتون** نمی‌تواند زمانی که ورودی اشتباهی در اختیار یک برنامه قرار می‌گیرد این مسئله را بررسی کند در نتیجه این شما هستید که در زمان درج ورودی‌های غیرمعتبر باید این مسئله را تشخیص داده و واکنش درستی نسبت به آن داشته باشید. دقت کنید که **پایتون** مفهوم داده‌های خوب یا بد را متوجه نمی‌شود، **پایتون** به سادگی داده‌های وارد شونده را بر مبنای یکسری قواعد از پیش تنظیم شده پردازش می‌کند. خلاقیت جزء خصایص **پایتون** یا هیچ زبان برنامه‌نویسی دیگری نیست، این ویژگی تنها در اختیار توسعه‌دهندگان قرار دارد. زمانی که یک خطای شبکه رخ می‌دهد یا کاربر کار غیر معمولی انجام می‌دهد، **پایتون** نمی‌تواند راه‌حلی برای برطرف کردن مشکل ارائه کند. **پایتون** فقط و فقط کدها را پردازش می‌کند. در این حالت اگر شما به عنوان یک برنامه‌نویس کدهای درستی را برای مدیریت یک خطا ننوشته باشید، برنامه به احتمال زیاد کرش کرده و به شکل تصادفی پایان می‌پذیرد. برنامه‌نویسان هیچ‌گاه نمی‌توانند همه علل بروز مشکلات را پیش‌بینی کرده و برای آن‌ها راه‌حلی ارائه کنند، به همین دلیل است که اشتباهات در برنامه‌های پیچیده بیش‌تر است. قبل از آن‌که شروع به کدنویسی برای پاسخ‌گویی به خطاها کنید در اولین گام باید منبع بروز مشکل را شناسایی کنید. در حالت کلی خطاها به دو گروه زیر طبقه‌بندی می‌شوند.

خطاهایی که در یک زمان خاص رخ می‌دهند

خطاهایی که از نوع خاصی هستند.

در شماره آینده آموزش **پایتون** توضیحات بیشتری درباره هر یک از این طبقه‌بندی‌ها ارائه خواهیم کرد.

تاریخ انتشار:

02 اسفند 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/programming/14673/%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%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-%D9%85%D9%81%D9%87%D9%88%D9%85-%D9%85%D8%AF%DB%8C%D8%B1%DB%8C%D8%AA-%D8%AE%D8%B7%D8%A7%D9%87%D8%A7-%D8%AF%D8%B1-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86