ایجاد استثناهای سفارشی در پایتون

در پایتون، استثناها (Exceptions) ابزاری بسیار مهم برای مدیریت خطاها در برنامه‌ها هستند. معمولاً از استثناها برای نشان دادن وضعیت‌های غیرعادی یا خطاهایی که در حین اجرای برنامه رخ می‌دهند استفاده می‌شود. در موارد خاص، می‌توانید استثناهای سفارشی (Custom exceptions) خود را ایجاد کنید تا خطاهای خاصی که در برنامه شما رخ می‌دهند را مدیریت کنید.

1. تعریف کلاس‌های استثنا

برای ایجاد استثناهای سفارشی، کافی است که یک کلاس جدید بسازید که از کلاس پایه `Exception` ارث‌بری کند. این کلاس جدید می‌تواند حاوی پیام‌های سفارشی و ویژگی‌های اضافی برای مدیریت خطاها باشد.

ساختار یک استثنای سفارشی

```
class CustomError(Exception):
    def __init__(self, message):
        super().__init__(message)
        self.message = message

# استفاده از استثنای سفارشی
def check_age(age):
    if age < 18:
        raise CustomError("Age must be 18 or older.")
    return "Age is valid."

try:
    print(check_age(16))
except CustomError as e:
    print(f"Error: {e}")
```

در این مثال:

- کلاس `CustomError` از کلاس `Exception` ارث‌بری می‌کند.
- در متد `__init__` می‌توانید پیام یا ویژگی‌های دیگری به استثنا اضافه کنید.
- در صورت وقوع خطا، استثنا با پیام خاص خود پرتاب می‌شود.

2. استفاده از استثناهای سفارشی برای مدیریت خطاهای خاص

استفاده از استثناهای سفارشی به شما این امکان را می‌دهد که خطاها و شرایط خاصی که ممکن است در یک سیستم بزرگتر یا برنامه پیچیده رخ دهند را بهتر مدیریت کنید. به‌عنوان مثال، ممکن است بخواهید یک استثنا برای خطاهایی که به دلیل اتصال به پایگاه داده ایجاد می‌شوند یا یک استثنا برای مدیریت ورودی نامعتبر از کاربر ایجاد کنید.

مثال‌های کاربردی

1. **مدیریت خطای اتصال به پایگاه داده**

```
class DatabaseConnectionError(Exception):
    def __init__(self, message="Database connection failed."):
        super().__init__(message)

def connect_to_database():
    # فرض کنید اتصال موفقیت‌آمیز نیست
    raise DatabaseConnectionError("Unable to connect to the database.")

try:
    connect_to_database()
except DatabaseConnectionError as e:
    print(f"Database error: {e}")
```

1. **مدیریت ورودی نامعتبر**

```
class InvalidInputError(Exception):
    def __init__(self, message="Input is invalid."):
        super().__init__(message)

def process_input(user_input):
    if not user_input.isdigit():
        raise InvalidInputError("Input should be a number.")
    return int(user_input)

try:
    print(process_input("abc"))
except InvalidInputError as e:
    print(f"Error: {e}")
```

3. به‌کارگیری استثناهای سفارشی در سیستم‌های بزرگتر

در پروژه‌های بزرگ، استفاده از استثناهای سفارشی کمک می‌کند تا کد تمیزتر و قابل نگهداری‌تر باشد. این امکان به شما داده می‌شود که خطاها را به‌طور دقیق‌تر دسته‌بندی کنید و پیغام‌های خطای بیشتری را در اختیار کاربران قرار دهید.

مزایای استفاده از استثناهای سفارشی

- **کاهش پیچیدگی کد**: با داشتن استثناهای خاص برای انواع مختلف خطاها، برنامه‌نویسان می‌توانند مدیریت خطا را بهتر سازماندهی کنند.
- **خطایابی راحت‌تر**: استثناهای سفارشی به شما این امکان را می‌دهند که خطاها را به‌طور مشخص‌تری شناسایی کرده و پاسخ مناسب را ارسال کنید.
- **مستندسازی بهتر**: با استفاده از استثناهای سفارشی، سایر برنامه‌نویسان می‌توانند بفهمند که کدام خطاها ممکن است در برنامه رخ دهند.

جمع‌بندی

- با استفاده از کلاس‌های استثنا در پایتون، می‌توانید استثناهای سفارشی را ایجاد کنید که مناسب با نیازهای خاص برنامه‌تان باشند.
- استثناهای سفارشی به شما امکان می‌دهند که به‌صورت دقیق‌تری خطاهای مختلف را شناسایی کرده و آن‌ها را مدیریت کنید.

- استفاده از استثناهای سفارشی در برنامه‌های پیچیده باعث بهبود نگهداری، تست و خطایابی کد می‌شود.