পাইথনে এরর হ্যান্ডেল করা

আবু আশরাফ মাসনুন

পাইথন প্রোগ্রামিং পর্ব-৬

এরর হ্যান্ডলিং কী এবং কেনো

অন্য দশটি কাজের মতো প্রোগ্রামিংয়ের বেলায়ও অনেক ধরনের অনাহূত পরিস্থিতির মুখোমুখি হতে হয়। এর জন্য হয়তো আমরা আগে থেকে প্রস্তুত থাকি না। যেমন, আপনি একটি ফাংশন লিখেছেন, যার কাজ দুটো সংখ্যার ভাগফল নির্ণয় করা। ফাংশনটির জন্য আপনি নিচের মতো কিছু কোড লিখেছেন–

```
def divide(a,b): return a/b
```

এক লাইনের এ ফাংশনটি তার প্রথম প্যারামিটারকে দ্বিতীয় প্যারামিটার দিয়ে ভাগ করে ভাগফল রিটার্ন করে। এটি মোটামুটি সব ক্ষেত্রেই ঠিক মতো কাজ করবে। যদি ভুল করে আমরা b-এর মান ০ দেই, তবে কী হবে? শূন্য দিয়ে কি কোনো সংখ্যাকে ভাগ করা সম্ভব?

```
#!/usr/bin/env python
def divide(a,b): return a/b
print divide(2,0)
```

আমরা যদি নিচের কোড রান করি, তবে এ ধরনের আউটপুট পাব–

```
Lighthouse: ~/Codes/py
→ python test.py
Traceback (most recent call last):
  File “test.py”, line 5, in <module>
    print divide(2,0)
  File “test.py”, line 3, in divide
    def divide(a,b): return a/b
ZeroDivisionError: integer division or modulo by zero
```

অর্থাৎ পাইথন আমাদেরকে জানিয়ে দিচ্ছে, আমাদের দেয়া লজিক সে ঠিকমতো প্রয়োগ করতে পারেনি। বরং সে একটি এররের মুখোমুখি হয়েছে, যার নাম ZeroDivisionError।

এমন আরও অনেক উদাহরণ দেয়া যেতে পারে। আপনি একটি ওয়েবসাইটের কনটেন্ট ডাউনলোড করার জন্য একটি প্রোগ্রাম লিখলেন। প্রোগ্রামটি বেশ ভালোই কাজ করছে যতক্ষণ আপনি নেটে কানেক্টেড আছেন। কিন্তু ইন্টারনেট কানেকশন যদি না থাকে তাহলে? সে ক্ষেত্রে পাইথন আবারও এরর দেখাবে।

এ ধরনের অপ্রত্যাশিত এররগুলোকে ভালোভাবে কাটিয়ে ওঠাই এরর হ্যান্ডলিংয়ের কাজ। আমরা মূলত পাইথনকে বলে দেব এ ধরনের অপ্রত্যাশিত পরিস্থিতি কীভাবে এড়িয়ে চলা যায়।

এক্সেপশন বেসিক

পাইথন যখনই অপ্রত্যাশিত কিছুর মুখোমুখি হয়, তখনই জানিয়ে দেয় সে এটা প্রত্যাশা করেনি। পাইথন এসব ক্ষেত্রে রেইজ করে নানা এক্সেপশন। আর আমরা তাই পাইথনকে বলে দেব প্রথমে আমাদের লজিক ট্রাই করতে। সেটা যদি করার মতো না হয় তবে কী করতে হবে সেটাও বলে দেব এক্সেপ্ট কিওয়ার্ডটি ব্যবহার করে। শব্দগুলো দেখেই হয়তো বুঝতে পারছেন এগুলোর কাজ কী। আমাদের একটু আগে লেখা ফাংশনটিকে আমরা নিচের মতো করে পরিবর্তন করে নেই–

```
#!/usr/bin/env python
def divide(a,b): return a/b
try:
    print divide(2,0)
except:
    print “something went wrong”
Gevi ivb K‡i †`LybÑ
Lighthouse: ~/Codes/py
→ python test.py
something went wrong
```

এবার কিন্তু পাইথন জানত অপ্রত্যাশিত পরিস্থিতিতে কী করতে হবে। সে সেটাই করেছে। পাইথনে এরর হ্যান্ডলিংয়ের জন্য আমরা try…except…finally ব্লক ব্যবহার করি। প্রথমেই থাকে try ব্লক। এ ব্লকে থাকবে আমাদের মূল কোড। পাইথন এ কোড রান করবে। যদি কোনো এক্সেপশন থাকে, তবে পাইথন except ব্লকে থাকা কোড রান করবে। এছাড়া একটি অপশনাল finally ব্লক যোগ করা যেতে পারে। এরর থাকুক আর নাই থাকুক এ ব্লকের কোড রান করবেই। এ ব্লকটি কাজে লাগে যখন আমাদের অসমাপ্ত কাজ শেষ করতে হয়। ধরুন, আপনি একটি ডাটাবেজ কানেকশন ওপেন করেছিলেন। এরর থাকুক আর নাই থাকুক, এ কানেকশনটি ক্লোজ করতেই হবে। এ ধরনের কাজগুলো আমরা ফাইনাল ব্লকে বসাতে পারি।

বিশেষ বিশেষ এক্সেপশন হ্যান্ডল করা

except কিওয়ার্ডের পর আমরা নির্দেশ করে দিতে পারি কোন এক্সেপশনগুলো হ্যান্ডেল করতে চাই। যেমন, আগের উদাহরণটি আরও নির্দিষ্ট করে আমরা এভাবে লিখতে পারতাম–

```
def divide(a,b): return a/b
try:
    print divide(2,0)
except ZeroDivisionError:
    print “You can’t divide by zero”
```

এভাবে আমরা একাধিক except ব্লক আর নির্দিষ্ট এক্সেপশন ব্যবহার করে আলাদা এক্সেপশনের জন্য আলাদা ব্যবস্থা করতে পারি।

নিজের কোডে এক্সেপশন রেইজ করা

আমরা নিজেদের কোডেও এক্সেপশন রেইজ করতে পারি রেইজ কিওয়ার্ড ব্যবহার করে। যেমন–

```
def create_a_mess(): raise Exception(“It’s such a Mess!”)
create_a_mess()
```

প্রোগ্রামটি রান করুন–

```
Lighthouse: ~/Codes/py
→ python test.py
Traceback (most recent call last):
  File “test.py”, line 5, in <module>
    create_a_mess()
  File “test.py”, line 3, in create_a_mess
        def create_a_mess(): raise Exception(“It’s such a Mess!”)
Exception: It’s such a Mess!
```

দেখা যাচ্ছে, আমরা একটি এক্সেপশন রেইজ করেছি। পাইথনের এরর হ্যান্ডলিং কিংবা এক্সেপশন সম্পর্কে আরও বিস্তারিত জানতে অফিশিয়াল ডকুমেন্টশন দেখা যেতে পারে কজ

ফিডব্যাক : masnun@gmail.com