جزوه برنامه نویسی پیشرفته

امیرحسین قلی زاده

دانشگاه علوم و فنون آریان ترم دوم ۱۴۰۳ – ۱۴۰۴

جزوه آموزشی – جلسه اول: توابع پیشرفته و مدیریت خطاها

استاد: امیرحسین قلی زاده | **دانشگاه:** علوم و فنون آریان | **نیمسال تحصیلی:** نیمسال دوم ۱۴۰۳ - ۱۴۰۴

فهرست مطالب

جزوه آموزشی – جلسه اول: توابع پیشرفته و مدیریت خطاها ۱
مقدمه ۲
توابع بازگشتی ۲
تعریف و مفهوم بازگشت ۲
اصول مهم توابع بازگشتی: ۲
ساختار یک تابع بازگشتی ۳
مثال: تابع فاکتوریل ۳
توابع لامبدا ۴
تعریف و کاربرد ۴
نحو تعریف تابع لامبدا: ۴
مثال‌های کاربردی ۴
مدیریت خطاها (Exception Handling) ۵
مفهوم استثناها ۵
ساختار try-except ۵
تمرینات و پروژه‌های کلاسی ۶
تکلیف شخصی ۶
تکلیف گروهی ۷
نتیجه‌گیری ۷
پایان جزوه ۷

مقدمه

در این جلسه، به بررسی مباحث پیشرفته در برنامه‌نویسی پایتون می‌پردازیم. دو مبحث اصلی مورد بحث عبارتند از:

- **توابع پیشرفته**: شامل توابع بازگشتی (Recursive Functions) و توابع لامبدا (Lambda Functions).
- **مدیریت خطاها**: تکنیک‌هایی برای کنترل و مدیریت خطاها در برنامه به منظور افزایش پایداری و کارایی.

هدف از این جلسه، آشنا شدن دانشجویان با مفاهیم فوق و کاربردهای آن‌ها در پروژه‌های واقعی است.

توابع بازگشتی

تعریف و مفهوم بازگشت

توابع بازگشتی توابعی هستند که در داخل خودشان دوباره فراخوانی می‌شوند. این روش به خصوص در مسائلی که به صورت تقسیم و حل (Divide and Conquer) قابل تفکیک هستند، بسیار مفید است.

اصول مهم توابع بازگشتی:

- **حالت پایه (Base Case)**: شرطی که فراخوانی بازگشتی متوقف شود.
- **فراخوانی بازگشتی (Recursive Call)**: قسمتی از تابع که خود تابع را با ورودی جدید فراخوانی می‌کند.

ساختار یک تابع بازگشتی

یک تابع بازگشتی معمولاً به صورت زیر نوشته می‌شود:

```
def recursive_function(parameters):
    # شرایط پایان: حالت پایه
    if base_condition:
        return result
    else:
        # فراخوانی بازگشتی با تغییر پارامترها
        return recursive_function(modified_parameters)

```

مثال: تابع فاکتوریل

فاکتوریل یک عدد، حاصل ضرب تمام اعداد صحیح مثبت کوچکتر یا مساوی آن عدد است. تعریف ریاضی فاکتوریل به صورت زیر است:

$$\mathbf{n!} = \begin{cases} \mathbf{1} & \text{اگر} \quad \boldsymbol{n = 0} \\ \boldsymbol{n \times (n-1)!} & \text{اگر} \quad \boldsymbol{n > 0} \end{cases}$$

کد نمونه با استفاده از تابع بازگشتی:

```
def factorial(n):
    # حالت پایه: اگر n برابر 0 باشد
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# تست تابع
print(factorial(5))  # خروجی: 120

```

نکته: هنگام کار با توابع بازگشتی باید مراقب شرایط خاتمه (Base Case) باشید تا از بروز خطای بیش از حد فراخوانی (Recursion Error) جلوگیری شود.

توابع لامبدا

تعریف و کاربرد

توابع لامبدا توابع ناشناس (بدون نام) در پایتون هستند. این توابع به‌طور مختصر تعریف شده و معمولاً در مواقعی که یک تابع کوچک و یک‌بار مصرف نیاز است، به کار می‌روند.

نحو تعریف تابع لامبدا:

```
lambda arguments: expression
```

مثال‌های کاربردی

۱. جمع دو عدد:

```
add = lambda x, y: x + y
print(add(3, 4))  # خروجی: 7
```

۲. فیلتر کردن لیست:

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # خروجی: [2, 4, 6]
```

۳. مرتب‌سازی بر اساس کلید سفارشی:

```
students = [
    {'name': 'Ali', 'grade': 85},
    {'name': 'Sara', 'grade': 92},
    {'name': 'Reza', 'grade': 78}
]
# سازی بر اساس نمره‌مرتب
sorted_students = sorted(students, key=lambda student: student['grade'])
print(sorted_students)
```

مدیریت خطاها (Exception Handling)

مفهوم استثناها

در برنامه‌نویسی، خطاها یا استثناها (Exceptions) می‌توانند در زمان اجرا رخ دهند. مدیریت صحیح استثناها باعث افزایش پایداری برنامه و ارائه پیام‌های مناسب به کاربر می‌شود.

ساختار try-except

بلاک‌های try-except برای کنترل خطاها به کار می‌روند. ساختار کلی آن به صورت زیر است:

```
try:
    # بلوک کد مشکوک به بروز خطا
    risky_code()
except SpecificError as e:
    # کدی که در صورت وقوع خطای مشخص اجرا می‌شود
    handle_error(e)
else:
    #  کدی که اگر خطایی رخ ندهد اجرا می‌شود(اختیاری)
    code_if_no_error()
finally:
    #  کدی که همیشه اجرا می‌شود(اختیاری)
    final_cleanup()
```

مثال: استفاده از مدیریت استثنا در تابع فاکتوریل

در این مثال، از مدیریت استثنا برای کنترل ورودی‌های نادرست (مانند اعداد منفی یا ورودی‌های غیر عددی) استفاده می‌کنیم:

```
def factorial(n):
    """
    محاسبه فاکتوریل یک عدد صحیح غیر منفی به روش بازگشتی
    """
    # بررسی ورودی: باید عدد صحیح غیر منفی باشد
    if not isinstance(n, int) or n < 0:
        raise ValueError("ورودی باید یک عدد صحیح غیر منفی باشد.")

    # شرط خروج از بازگشت
    if n == 0:
        return 1

    return n * factorial(n - 1)
```

```
14.
15.
16. #  از استفاده try-except برای مدیریت خطاها
17. try:
18.     user_input = input("لطفاً یک عدد صحیح غیر منفی وارد کنید: ")
19.
20.     if not user_input.isdigit():
21.         raise ValueError("ورودی باید یک عدد صحیح غیر منفی باشد.")
22.
23.     user_input = int(user_input)
24.     result = factorial(user_input)
25.
26.     print(f"فاکتوریل{user_input}  برابر است با{result}")
27.
28. except ValueError as ve:
29.     print(f"خطا: {ve}")
30. except RecursionError:
31.     print("خطا: مقدار ورودی بیش از حد بزرگ است و باعث خطای بازگشتی شده است.")
32. except Exception as e:
33.     print("ای رخ داده استخطای ناشناخته:", e)19.
34.
```

توضیح:

- ابتدا ورودی کاربر دریافت شده و به عدد صحیح تبدیل می‌شود.
- در صورت وارد کردن ورودی نامعتبر (مثلاً یک رشته یا عدد منفی)، خطای ValueError ایجاد شده و در بلوک except مربوطه مدیریت می‌شود.

تمرینات و پروژه‌های کلاسی

تکلیف شخصی

- **موضوع:** نوشتن یک تابع بازگشتی برای محاسبه فاکتوریل
- **شرایط:**
  - از مدیریت استثنا استفاده کنید تا ورودی‌های نامعتبر (مانند اعداد منفی یا ورودی‌های غیر عددی) را کنترل کنید.

- در صورت وقوع خطا، پیام مناسبی نمایش دهید.

تکلیف گروهی

- **موضوع:** ساخت یک پروژه گروهی ساده که:
  - از کاربر ورودی دریافت کند.
  - در صورت ورود داده‌های نامعتبر، خطا را مدیریت کرده و پیام مناسبی نمایش دهد.
- **راهنمایی:**
  - می‌توانید یک فرم ساده برای دریافت اطلاعات از کاربر ایجاد کنید.
  - از توابع و مدیریت خطاها برای اعتبارسنجی ورودی‌ها استفاده کنید.

نتیجه‌گیری

در این جلسه، با مفاهیم توابع پیشرفته (بازگشتی و لامبدا) و مدیریت خطاها در پایتون آشنا شدیم. یادگیری این مفاهیم به شما کمک می‌کند تا برنامه‌هایی با ساختار منظم‌تر، کدهای کوتاه‌تر و پایدارتری بنویسید. همچنین، مدیریت صحیح خطاها از بروز مشکلات ناخواسته در زمان اجرا جلوگیری کرده و تجربه کاربری بهتری ارائه می‌دهد.

پایان جزوه

امیدواریم این آموزش مفید واقع شده باشد. در صورت داشتن هرگونه سوال، در کلاس یا از طریق پلتفرم‌های ارتباطی با استاد مشورت کنید.

با آرزوی موفقیت برای شما دانشجویان عزیز!