کار با فایل‌های متنی در پایتون: فراتر از خواندن

در درس‌نامه قبلی، با اصول خواندن اطلاعات از فایل‌های متنی آشنا شدیم. اکنون، به بررسی نحوه نوشتن، ویرایش و مدیریت فایل‌ها می‌پردازیم.

نوشتن در فایل: خلق محتوای جدید

- تابع `write()` : ابزاری قدرتمند برای افزودن متن به فایل.

```
f = open("output.txt", 'w') # Creates or overwrites "output.txt"
f.write("Welcome to the world of Python file handling!\n")
f.write("This is a new line of text.")
f.close()
```

نکته: حالت `'w'` (write) محتوای قبلی فایل را پاک می‌کند. برای افزودن متن به انتهای فایل، از حالت `'a'` (append) استفاده کنید.

مثال: برنامه‌ای که لیستی از نام‌ها را از کاربر دریافت کرده و در فایل `names.txt` ذخیره می‌کند.

```
names = []
while True:
    name = input("Enter a name (or 'done' to finish): ")
    if name.lower() == 'done':
        break
    names.append(name)

with open("names.txt", 'w') as f:
    for name in names:
        f.write(name + '\n')
```

اعمال تغییرات در فایل: ذخیره و پایداری

تغییرات شما تا زمانی که به صراحت ذخیره نشوند، در فایل اعمال نمی‌شوند.

- تابع `flush()` : برای ذخیره فوری تغییرات در فایل.

```
f = open("log.txt", 'a')
f.write("New log entry...\n")
f.flush() # Ensures the log entry is written immediately
```

مدیریت فایل: بستن و اطمینان

بستن فایل پس از اتمام کار، منابع سیستم را آزاد کرده و از بروز خطا جلوگیری می‌کند.

- تابع `close()` : برای بستن فایل.

```
f = open("data.txt", 'r')
# ... read data ...
f.close()
```

استفاده از دستور `with` بهترین روش برای مدیریت فایل است، زیرا فایل به طور خودکار پس از اتمام کار بسته می‌شود.

```
with open("config.txt", 'r') as f:
    config_data = f.read()
    # ... process config_data ...
```

خواندن و نوشتن همزمان: تعامل پویا

حالت `'r+'` امکان خواندن و نوشتن همزمان در فایل را فراهم می‌کند.

```
with open("data.txt", 'r+') as f:
    content = f.read()
    f.write("\nAdditional data...")
```

تولیدکننده‌ها در فایل‌ها: مدیریت حافظه برای فایل‌های بزرگ

برای پردازش فایل‌های حجیم، از تولیدکننده‌ها (Generators) استفاده کنید.

```
def process_large_file(filepath):
    with open(filepath, 'r') as f:
        for line in f:
            yield process_line(line)

for result in process_large_file("huge_data.csv"):
    # ... process result ...
```