نوشتن در فایل‌ها در پایتون

در پایتون برای نوشتن در فایل‌ها از تابع `()open` همراه با **حالت‌های نوشتن** (`wb`, `a`, `w`, **...**) استفاده می‌شود. دو روش اصلی برای نوشتن داده‌ها در فایل وجود دارد:

1. `()write` → برای نوشتن یک رشته.
2. `()writelines` → برای نوشتن چندین خط به‌صورت لیست.

---

۱. باز کردن فایل برای نوشتن (`w`)

🔹 حالت `"w"` فایل را **بازنویسی** می‌کند. اگر فایل از قبل وجود داشته باشد، محتوای آن پاک شده و اطلاعات جدید جایگزین می‌شود.

```
with open("output.txt", "w") as file:
    file.write("این یک متن نمونه است.\n")
    file.write("خط دوم در فایل نوشته شد.\n")
```

✅ **نکات:**

- اگر فایل **وجود نداشته باشد**، به‌صورت خودکار ساخته می‌شود.
- محتوای قبلی فایل **حذف شده و جایگزین می‌شود**.

---

۲. اضافه کردن به انتهای فایل (`a`)

🔹 حالت `"a"` **محتوای جدید را به انتهای فایل اضافه می‌کند** بدون اینکه اطلاعات قبلی پاک شود.

```
with open("output.txt", "a") as file:
    file.write("این متن جدید به انتهای فایل اضافه شد.\n")
```

✅ **نکات:**

- اگر فایل **وجود نداشته باشد**، ساخته می‌شود.
- اطلاعات جدید **به انتهای فایل اضافه می‌شوند** و چیزی حذف نمی‌شود.

---

۳. نوشتن چندین خط با `()writelines`

🔹 متد `()writelines` یک **لیست از رشته‌ها** را در فایل می‌نویسد.

```
lines = ["خط اول\n", "خط دوم\n", "خط سوم\n"]

with open("output.txt", "w") as file:
    file.writelines(lines)
```

✅ **نکته مهم:** `()writelines` **بین خطوط** `\n` **اضافه نمی‌کند**، بنابراین باید انتهای هر رشته `\n` قرار دهید.

---

۴. نوشتن در فایل‌های باینری (`wb`)

📌 اگر بخواهید فایل‌های غیرمتنی مانند تصاویر یا فایل‌های صوتی را ذخیره کنید، از حالت `wb` (write binary) استفاده کنید.

```
with open("binary_file.bin", "wb") as file:
    file.write(b'\x48\x65\x6C\x6C\x6F')  # نوشتن داده‌های باینری
```

✅ کاربرد:

- ذخیره **تصاویر، فایل‌های صوتی، و داده‌های رمزگذاری‌شده**.
- انتقال داده‌ها در **شبکه و ارتباطات**.

---

۵. نوشتن داده‌های ساختاریافته در فایل

📌 مثال: ذخیره اطلاعات به‌صورت JSON

```
import json

data = {"نام": "علی", "سن": 25, "شهر": "تهران"}

with open("data.json", "w", encoding="utf-8") as file:
    json.dump(data, file, ensure_ascii=False, indent=4)
```

✅ **مزیت:** فرمت JSON برای ذخیره‌سازی **داده‌های ساختاریافته** بسیار مناسب است.

---

۶. نوشتن داده‌های جدولی (CSV)

📌 مثال: ذخیره اطلاعات در فایل CSV

```
import csv

rows = [
    ["نام", "سن", "شهر"],
    ["علی", "25", "تهران"],
    ["رضا", "30", "مشهد"]
]

with open("data.csv", "w", newline='', encoding="utf-8") as file:
    writer = csv.writer(file)
    writer.writerows(rows)
```

✅ **مزیت:** مناسب برای **کار با داده‌های جدولی و ذخیره اطلاعات پایگاه داده**.

---

۷. ترکیب خواندن و نوشتن در فایل

📌 مثال: خواندن یک فایل، پردازش داده‌ها، و ذخیره در فایل جدید

```python
with open("input.txt", "r") as infile, open("output.txt", "w") as outfile:
    for line in infile:
        outfile.write(line.upper())  # تبدیل متن به حروف بزرگ و ذخیره در فایل جدید
```

---

۸. مدیریت بهینه حافظه هنگام نوشتن فایل‌های حجیم

📌 **نوشتن اطلاعات حجیم به فایل بدون اشغال زیاد حافظه**

```python
with open("large_output.txt", "w") as file:
    for i in range(1, 1000000):
        file.write(f"خط شماره {i}\n")
```

✅ **مزیت:** به‌جای ذخیره کل داده در حافظه، اطلاعات را مستقیماً در فایل می‌نویسد.

---

۹. جمع‌بندی

✅ `()write` → نوشتن یک رشته.
✅ `()writelines` → نوشتن چندین رشته از یک لیست.
✅ `"w"` → بازنویسی فایل، **محتوای قبلی پاک می‌شود**.
✅ `"a"` → افزودن به انتهای فایل، **محتوای قبلی حفظ می‌شود**.
✅ `"wb"` → نوشتن داده‌های باینری (عکس، صدا، ویدئو).
✅ `()json.dump` → ذخیره اطلاعات ساختاریافته (JSON).
✅ `()csv.writer` → نوشتن داده‌های جدولی.
✅ **مدیریت صحیح فایل باعث بهینه‌سازی حافظه و افزایش کارایی کد می‌شود!** 🚀