文件读写

1. 文件操作基础

- 文件是计算机上存储数据的一种方式，可以用于保存文本、图像、音频等信息。
- 文件处理是许多应用程序和脚本中常见的任务，如数据存储、配置读写等。

2. 打开和关闭文件

- 使用 open() 函数来打开文件，它返回一个文件对象，用于进行读取或写入操作。
- 使用 with 语句可以确保在使用文件后自动关闭文件，以避免资源泄漏。

```
# 示例代码：打开和关闭文件

file_path = "example.txt"

with open(file_path, "r") as file:

        content = file.read()

        print(content)
```

3. 读取文件内容

- 使用不同的方法读取文件内容，取决于需求：read()、readline() 和 readlines()。

```
# 示例代码：读取文件内容

with open("example.txt", "r") as file:

        content = file.read()

        print(content)
```

```
with open("example.txt", "r") as file:

        line = file.readline()

        print(line)



with open("example.txt", "r") as file:

        lines = file.readlines()

        for line in lines:

                print(line.strip()) # 去除换行符
```

4. 写入文件内容

- 使用 write() 方法将内容写入文件，可覆盖原有内容。
- 使用 writelines() 方法写入多行内容。

```
# 示例代码：写入文件内容

new_content = "This is new content."

with open("example.txt", "w") as file:

        file.write(new_content)



new_lines = ["Line 1\n", "Line 2\n", "Line 3\n"]

with open("example.txt", "w") as file:

        file.writelines(new_lines)
```

5. 追加内容到文件

- 使用 open() 函数的模式参数来实现追加。
- 使用 a 模式来追加内容到文件。

```
# 示例代码：追加内容到文件

additional_content = "This is additional content."

with open("example.txt", "a") as file:

        file.write(additional_content)
```

6. 异常处理与文件

- 在文件操作时，可能会遇到异常，如文件不存在、权限问题等。
- 使用 try 和 except 块来捕获并处理异常。

```
# 示例代码：异常处理与文件

try:

        with open("nonexistent.txt", "r") as file:

                content = file.read()

                print(content)

except FileNotFoundError:

        print("File not found.")
```

7. 文件处理的最佳实践

- 使用 with 语句自动关闭文件，确保资源得到释放。
- 将文件操作封装为函数，提高代码的可读性和可维护性。

8. 实际应用案例

- 创建和维护日志文件，记录程序运行情况。
- 读取和修改配置文件，存储应用程序设置。
- 数据持久化与存储，保存应用程序状态。