第 5 章
文件操作与管理

5.1 实验一：读写 TXT 格式文件

一、实验目的

- 掌握使用 Python 读写 TXT 格式文件的几种方法。
- 熟练掌握在程序设计中读写 TXT 格式文件的方法。

二、相关知识

在 Python 中，可以使用内置函数 open()读写文本文件。

1. 打开文件

使用 open()函数打开一个 TXT 格式文件。它接收 2 个参数：文件路径和打开模式。打开模式可以是‘r’（只读模式）、‘w’（写入模式）、‘a’（追加模式）或‘x’（创建模式）。

2. 读取文件内容

一旦打开了文件，可以使用 read()方法读取文件的全部内容，或者使用 readline()方法逐行读取文件。

3. 写入文件内容

如果以写入模式打开文件，可以使用 write()方法写入文本内容。可以一次写入多行文本，并使用\n 来表示换行。

4. 关闭文件

在完成读写文件后，应该关闭文件释放资源，可以使用 close()方法关闭文件。

5. 使用上下文管理器

使用上下文管理器（with 语句）能够确保文件等资源在使用完毕后自动释放，简化代码结构并提高程序的健壮性。另外，在读取或写入文件时注意文件路径和权限，以及及时关闭文件来避免资源泄露。

三、实验要求

通过 Python 编写程序，实现读取给定的 TXT 格式文件并进行一些简单的文本处理操作，最后将处理结果写入新的 TXT 格式文件。

四、实验内容

（1）创建一个名为 input.txt 的 TXT 格式文件，并在其中添加一些文本内容，至少包含多行文本。

（2）编写 Python 程序，读取 input.txt 文件的文本内容。

（3）对读取的文本内容进行以下处理操作。

① 统计文件中的行数、单词数和字符数。

② 将每一行的文本进行逆序处理，即将每行文本反转。

③ 将所有文本转换为小写字母形式。

④ 过滤掉所有非字母字符（如标点符号、数字等）。

⑤ 创建一个名为 output.txt 的新 TXT 格式文件，并将处理后的文本内容写入其中。

（4）在输出文件中，每个处理结果应位于新的一行，并包含相应的标识符或描述，以便清晰展示每个处理操作的结果。

五、参考代码

本实验参考代码如下：

```
def process_text(text):
    reversed_text = text[::-1]
    lowercase_text = reversed_text.lower()
    filtered_text = ''.join(ch for ch in lowercase_text if ch.isalpha())
    return filtered_text

#读取 input.txt 文件的内容
with open('input.txt', 'r') as input_file:
    lines = input_file.readlines()

#统计行数、单词数和字符数
num_lines = len(lines)
num_words = sum(len(line.split()) for line in lines)
num_chars = sum(len(line) for line in lines)

#输出统计结果到终端
print(f"Total lines: {num_lines}")
print(f"Total words: {num_words}")
print(f"Total characters: {num_chars}")
print()

#处理文本内容并输出到终端
for line in lines:
    processed_line = process_text(line.strip())
    print(f"Original text: {line.strip()}")
    print(f"Processed text: {processed_line}")
    print()
```

```
#写入处理后的文本内容到 output.txt 文件
with open('output.txt', 'w') as output_file:
    output_file.write(f"Total lines: {num_lines}\n")
    output_file.write(f"Total words: {num_words}\n")
    output_file.write(f"Total characters: {num_chars}\n\n")
    output_file.write("Processed text:\n")
    for line in lines:
        processed_line = process_text(line.strip())
        output_file.write(f"Original text: {line.strip()}\n")
output_file.write(f"Processed text: {processed_line}\n\n")
```

六、运行结果

本实验程序运行结果如图 5-1 和图 5-2 所示。

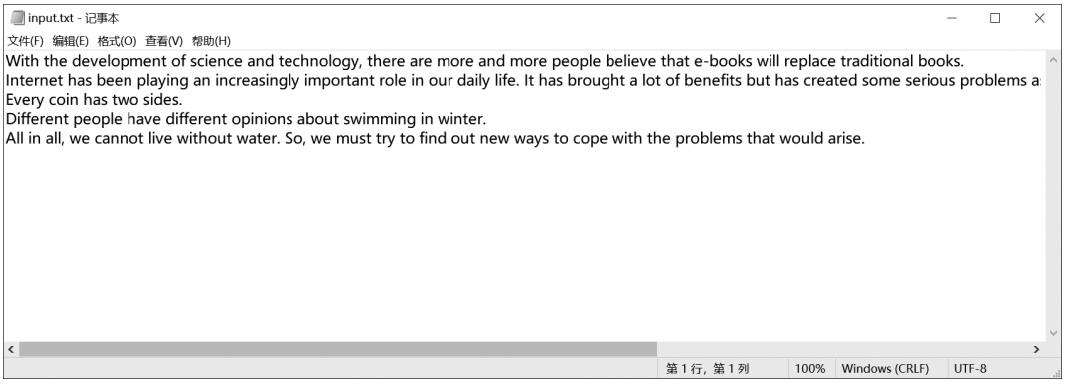

图 5-1　input.txt 文件

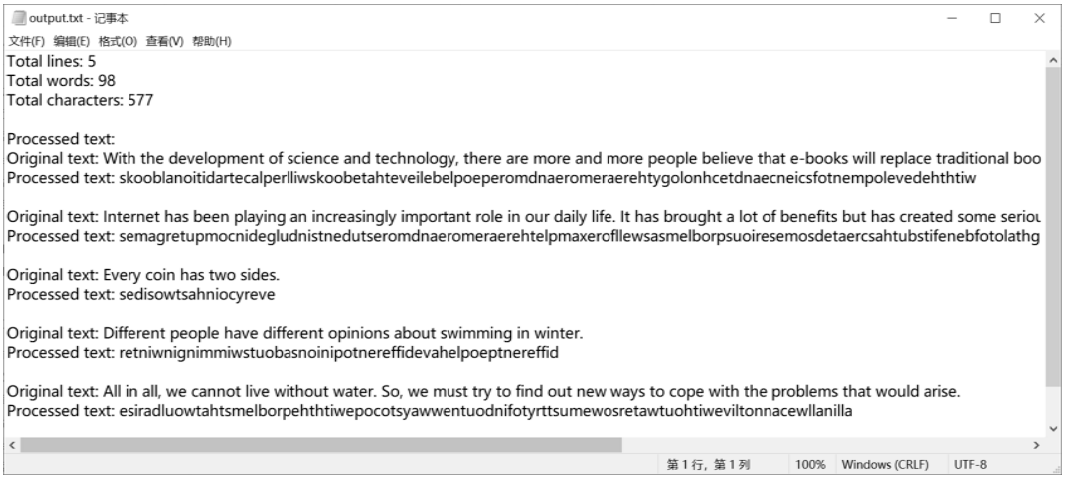

图 5-2　output.txt 文件

5.2 实验二：读写 JSON 格式文件

一、实验目的

- 掌握使用 Python 读写 JSON 格式文件的几种方法。
- 熟练掌握在程序设计中读写 JSON 格式文件的方法。

二、相关知识

在 Python 中，通常使用内置的 json 模块来读写 JSON 格式文件。

1. 导入模块

首先需要导入 json 模块，这个模块提供了处理 JSON 数据的方法。

2. 读取 JSON 格式文件

使用 json.load()方法来读取 JSON 格式文件。它接收一个已打开的文件对象作为参数，并返回解析后的 JSON 数据。

3. 写入 JSON 格式文件

如果要将 Python 对象写入 JSON 格式文件，可以使用 json.dump()方法。它接收两个参数：Python 对象和一个已打开的文件对象，将 Python 对象以 JSON 格式写入文件。使用 indent 参数指定缩进级别。

4. 读取和写入字符串

除了读取和写入文件，json 模块还提供了处理 JSON 字符串的方法。可以使用 json.loads()方法将 JSON 字符串解析为 Python 对象，使用 json.dumps()方法将 Python 对象转换为 JSON 格式的字符串。

三、实验要求

通过 Python 编写程序，实现读取给定的 JSON 格式文件并进行一些简单的数据处理操作，最后将处理结果写入新的 JSON 格式文件。

四、实验内容

（1）创建一个名为 input.json 的 JSON 格式文件，并在其中添加一些 JSON 数据，包含不同类型的字段和值（如字符串、数字、布尔值、列表、嵌套对象等）。

（2）编写 Python 程序，读取 input.json 文件的 JSON 数据，对读取的 JSON 数据进行以下处理操作。

① 统计 JSON 中的元素数量，并将统计结果输出到终端。

② 提取特定字段的值，并进行一些处理操作，如大小写转换、字符串拼接、数值计算等。

③ 创建一个新的 JSON 对象，将处理后的数据组织起来。

（3）创建一个名为 output.json 的新 JSON 格式文件，并将处理后的 JSON 数据写入其中。

（4）在输出文件中，包含原始 JSON 数据和处理后的 JSON 数据，并使用适当的标识

符描述，以便清晰展示每个处理操作的结果。

五、参考代码

本实验参考代码如下：

```
import json

#读取 input.json 文件的数据
with open('input.json', 'r') as input_file:
    data = json.load(input_file)

#统计 JSON 元素数量
num_elements = len(data)

#提取字段并进行处理操作
field_value = data['students']
for item in field_value:
    processed_value = item['name'].upper() + ' Processed'
#创建新的 JSON 对象
output_data ={
    'num_elements': num_elements,
    'processed_value': processed_value,
    'original_data': data
}

#写入处理后的 JSON 数据到 output.json 文件
with open('output.json', 'w') as output_file:
    json.dump(output_data, output_file, indent=4)
```

六、运行结果

本实验程序运行结果如图 5-3 和图 5-4 所示。

input.json

```
{
  "students": [
    {
      "id": 1,
      "name": "John Doe",
      "age": 20,
      "grades": [80, 85, 90]
    },
    {
      "id": 2,
      "name": "Jane Smith",
      "age": 19,
      "grades": [90, 95, 88]
    },
    {
      "id": 3,
      "name": "Mike Johnson",
      "age": 21,
      "grades": [75, 82, 78]
    }
  ],
  "university": "ABC University",
  "location": "New York"
}
```

图 5-3　input.json

output.json

```
{
    "num_elements": 3,
    "processed_value": "MIKE JOHNSON Processed",
    "original_data": {
        "students": [
            {
                "id": 1,
                "name": "John Doe",
                "age": 20,
                "grades": [
                    80,
                    85,
                    90
                ]
            },
            {
                "id": 2,
                "name": "Jane Smith",
                "age": 19,
                "grades": [
                    90,
                    95,
                    88
                ]
            },
```

图 5-4　output.json

5.3 习题

一、选择题

1. 在 Python 中，用于打开文件的函数是（　　）。

A. open()　　B. read()　　C. write()　　D. close()

2. 可以在打开文件时实现读取和写入功能的是（　　）。

A. "r"　　B. "w"　　C. "a"　　D. "r+"

3. 使用 Python 进行文件写入操作时，可以将内容写入文件的函数是（　　）。

A. write()　　B. read()　　C. open()　　D. append()

4. 若要以追加模式打开文件，应该使用的模式参数是（　　）。

A. "r"　　B. "w"　　C. "a"　　D. "x"

5. 在 Python 中，可以读取整个文件内容的方法是（　　）。

A. read()　　B. readline()　　C. readlines()　　D. open()

6. 可以用于逐行读取文件内容的方法是（　　）。

A. read()　　B. readline()　　C. readlines()　　D. open()

7. 在进行文件操作时，关闭一个已经打开文件的方法是（　　）。

A. close()　　B. open()　　C. write()　　D. read()

8. 使用 Python 进行文件复制的最简单的方法是（　　）。

A. 使用os.copy()函数

B. 使用shutil.copy()函数

C. 打开一个文件并将其内容写入另一个文件

D. 使用copyfile()函数

9. 在 Python 中，要删除一个文件，可以使用（　　）。

A. delete()函数　　B. remove()函数　　C. erase()函数　　D. discard()函数

10. 在 Python 中，用于检查文件是否存在的方法是（　　）。

A. file_exists()　　B. file_exists()　　C. exists()　　D. isfile()

11. 若要创建一个新的目录，可以使用的函数是（　　）。

A. create_directory()　　B. new_directory()

C. make_dir()　　D. mkdir()

12. 在 Python 中，可以获取文件的绝对路径的函数是（　　）。

A. get_path()　　B.　abs_path()　　C. get_abs_path()　　D. os.path.abspath()

13. 在 Python 中，用于重命名文件的方法是（　　）。

A. rename()　　B. rename_file()　　C. move()　　D. move_file()

14. 在 Python 中，获取文件大小（以字节为单位）的函数是（　　）。

A. get_size()　　B. size()　　C. file_size()　　D. os.path.getsize()

15. 若要遍历一个目录中的所有文件和子目录，可以使用的函数是（　　）。

A. list_files()　　B. get_files()　　C. traverse_dir()　　D. os.walk()

16. 在 Python 中，判断给定路径是否为一个文件夹的方法是（　　）。

A. is_folder()　　B. is_dir()　　C. is_directory()　　D. is_folder()

17. 若要将一个文件移动到另一个位置，可以使用的函数是（　　）。

A. move()　　B. move_file()　　C. rename()　　D. rename_file()

18. 在进行文件操作时，获取文件访问时间的方法是（　　）。

A. access_time()　　B. get_access_time()

C. modification_time()　　D. getmtime()

19. 在 Python 中，用于创建一个空文件的方法是（　　）。

A. create_file()　　B. new_file()　　C. touch()　　D. open()

20. 若要判断一个路径是否为绝对路径，可以使用的函数是（　　）。

A. is_absolute()　　B. is_relative()　　C. get_abs_path()　　D. os.path.isabs()

二、填空题

1. 文件可分为_____和_____两种类型。

2. _____方法主要用于从文件中按行读取存储的内容，当读取的内容超过文本中存储的行数时，该函数读取到的内容为_____。_____方法能够一次将文本内容读取到文件对象中。

3. 创建好文本文件后，可以通过_____方法和_____方法分别对文件进行打开和关闭操作。