量子文件舱管理系统学习笔记

1.文件操作与目录管理：

这里之前学python的时候没太认真看，所以借此机会重新学了一遍。

(1)os模块：

os模块提供的一些操作目录的 函数

os模块提供的与目录相关的函数

getcwd()	返回当前的工作目录
listdir(path)	返回指定路径下的文件和目录信息
mkdir(path [,mode])	创建目录
makedirs(path1/path2… [,mode])	创建多级目录
rmdir(path)	删除目录
removedirs(path1/path2……)	删除多级目录
chdir（path）	把path设置为当前工作目录
walk(top[,topdown[,onerror]])	遍历目录树，该方法返回一个元组，包括所有路径名、所有目录列表和文件列表3个元素

os.path模块提供的一些操作目录的函数

os.path模块提供的与目录相关的函数

abspath(path)	用于获取文件或目录的绝对路径
exists(path)	用于判断目录或者文件是否存在，如果存在则返回True，否则返回False
join(path,name)	将目录与目录或者文件名拼接起来
splitext()	分离文件名和扩展名
basename(path)	从一个目录中提取文件名
dirname(path)	从一个路径中提取文件路径，不包括文件名
isdir(path)	用于判断是否为路径

注：(1)os.mkdir创建的是一级目录，需保证父目录一定存在；os.makedirs可以创建多级目录，即便路径中某些路径不存在，它都会创建完整路径

(2)

```
os.walk(top[, topdown][, onerror][, followlinks])
```

top：指定要遍历内容的根目录。

topdown：可选参数，用于指定遍历的顺序。若值为True，表示自上而下遍历（先遍历根目录），若值为 False，表示自下而上遍历。（默认值为True）

onerror：可选参数，用于指定错误处理方式，默认忽略，使用时指定一个错误处理函数即可。

followlinks：可选参数，默认情况下将参数值设为True。

返回值：返回三个元素(dirpath, dirnames, filenames)

（3）

```
os.path.splitext(filename)
```

它会将文件路径拆分为两部分：

1.主文件名

2.拓展名（以.开头的部分）（有多个.只识别最后一个点后的内容作为拓展名）

```
import os

filename = "data/alien_research.quantum"
root, ext = os.path.splitext(filename)

print(f"主文件名：{root}")  # 输出：data/alien_research
print(f"扩展名：{ext}")    # 输出：.quantum
```

(2)shutil模块：

1.移动文件或目录：shutil.move()

```
shutil.move(src, dst)
```

将文件或目录从一个位置移动到另一个位置。如果目标路径已经存在，`move()` 会覆盖目标（如果目标是文件）。如果目标是目录，会将源文件/目录移动到目标目录下。

2.复制文件：shutil.copy()和shutil.copy2()

```
shutil.copy(src, dst)   # 复制文件内容
shutil.copy2(src, dst)  # 复制文件内容和元数据
```

将源文件复制到目标位置。`copy()` 只复制文件内容，而 `copy2()` 会复制文件内容以及元数据（如修改时间等）。

3.递归删除目录：shutil.rmtree()

```
shutil.rmtree(path)
```

递归地删除指定路径下的目录及其所有内容（文件和子目录）。这非常适合清空一个目录及其子目录。

4.复制整个目录：shutil.copytree()

```
shutil.copytree(src, dst)
```

递归地复制整个目录，包括目录中的所有文件和子目录。

5.压缩与解压：shutil.make_archive()和shutil.unpack_archive()

```
shutil.make_archive(base_name, format, root_dir, base_dir)
```

用于创建归档文件（如 `.zip` 或 `.tar`）以及解压归档文件

（3）时间戳与文件重命名：

在本题中，为了防止未知文件文件名出现冲突，所以引入Python中的datetime模块来生成时间戳。

时间戳通常使用当前的日期和时间（精确到秒甚至毫秒），将其转化为字符串格式，然后附加到文件名中。

1.datetime.now()获取当前时间：

```
import datetime

current_time = datetime.datetime.now()
print(current_time)
```

```
2025-02-12 14:35:47.123456
```

2.strftime(format)格式化时间：

该方法可以将datetime对象转化为所需要的格式，可以用不同的格式字符来表示日期喝时间的各个部分。

```
timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
print(timestamp)
```

```
20250212143547
```

3.文件重命名：

在处理多个文件时，为了避免文件名冲突，可以把时间戳附加到文件名中，比如：

```
import datetime
import os

filename = "alien_research.quantum"
timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")

# 生成新的文件名
new_filename = f"{timestamp}_{filename}"
print(new_filename)
```

(4)json模块：

json.dumps()方法用于将 Python 对象转换为 JSON 格式的字符串

json.load()方法用于从json文件中读取数据

2.代码实现流程：

1.定义文件拓展名与目标目录的映射关系：

先定义映射关系，后续在文件分类的函数中可以根据文件的拓展名对文件进行分类。

```
srtucture = {
    '.quantum': ('quantum_core', 'SECTOR-7G'),
    '.holo': ('hologram_vault', 'CHAMBER-12F'),
    '.exo': ('exobiology_lab', 'POD-09X'),
    '.chrono': ('temporal_archive', 'VAULT-00T')
}
```

2.定义用于文件分类的函数：

```
# 定义文件分类的函数
def classify_files(directory):
    # 遍历目录中的文件
    for file in os.listdir(directory):
        file_path = os.path.join(directory, file) # 获取文件的完整路径

        if not os.path.isfile(file_path): # 如果路径不是文件，跳过
            continue

        ext = os.path.splitext(file)[1] # 将文件名拆分为文件名和拓展名两部分，获取文件拓
展名
        dest_info = structure.get(ext)
        if dest_info:
            dest_dir = os.path.join(*dest_info)
        else:
            dest_dir = "quantum_quarantine"


        if not dest_info: # 如果是未知类型的文件
            new_file_name =
f"ENCRYPTED_{datetime.now().strftime('%Y%m%d%H%M%S')}_{file}" # 进行重命名（前缀加
ENCRYPTED_）并加上时间戳
```

```
            dest_path = os.path.join(directory, dest_dir, new_file_name) # 生成目
标路径
        else: # 如果是已知类型的文件
            new_file_name = f"{datetime.now().strftime('%Y%m%d%H%M%S')}_{file}" #
同样为文件命名，添加时间戳
            dest_path = os.path.join(directory,dest_dir,new_file_name) # 生成目标路
径

        os.makedirs(os.path.dirname(dest_path), exist_ok=True) # 创建目标文件夹，若
已存在则不做任何操作

        shutil.move(file_path, dest_path) # 将文件移动到目标目录
        print(f"移动 {file} 至 {dest_path}") # 打印文件移动的信息
```

这里先根据文件的拓展名将文件按照第一步定义的structure进行分类，分类之后的文件处理方式有两种，分别是已知类型的文件和未知类型的文件。

`structure.get(ext, "quantum_quarantine/")`：

- `dict.get(key, default_value)`

  方法会：

  1. **如果 `ext` 在 `structure` 字典中，返回对应的值**（如 `('.hologram_vault', 'CHAMBER-12F')`）。
  2. **如果 `ext` 不在 `structure` 中，返回默认值 `"quantum_quarantine/"`**。

在这一步处理完之后，就可以通过dest_info区分两种不同的文件，对这两种文件进行对应的处理（文件命名，添加时间戳，生成目标路径）。

最后创建目标文件夹并进行文件移动。

3.定义全息日志生成函数：

```
# 定义生成日志文件的函数
def generate_log(directory):
    # 日志头部
    header = [
        "┌──────────────────────────────────┐",
        "|  Xia-III 空间站数据分布全息图  |",
        "└──────────────────────────────────┘",
        ""
    ]
    # 日志尾部
    footer = [
        "",
        f"🤖 SuperNova · 地球标准时  {datetime.now().strftime('%Y-%m-
%dT%H:%M:%S')}",
        "⚠️ 警告：请勿直视量子文件核心"

    ]

    # 日志文件的路径
    log_file = os.path.join(directory,"log.txt")

    # 打开日志文件进行写入操作
```

```
    with open(log_file, "w",encoding="utf-8") as log:
        log.write("\n".join(header) + "\n")  # 写入头部

        # 写入日志中间部分信息
        # 遍历所有文件
        for root,_,files in os.walk(directory):
            level = root.replace(directory, "").count(os.sep) # 计算当前目录的层级
            indent = "|   " * level + "├─ " # 计算当前层级所对应的缩进
            log.write(f"{indent}🚀 {os.path.basename(root)}\n") # 写入当前目录的名称
            for file in files: # 遍历当前目录下的文件
                file_indent = "|   " * (level + 1) + "├─ " # 计算当前文件所对应的缩进
                symbol = "🔮" if "ENCRYPTED" not in file else "⚠️" # 如果文件没有
“ENCRYPTED”则用🔮标记，否则用⚠️
                log.write(f"{file_indent}{symbol} {file}\n") # 写入文件信息
        log.write("\n".join(footer) + "\n")  # 写入尾部

    return log_file # 返回生成的日志文件的路径
```

这里分为三个部分，日志的头部、尾部和中部，头部和尾部相对简单直接定义即可。

中部需根据不同文件层级计算对应的缩进，对已知类型和未知类型的文件可以通过之前添加的"ENCRYPTED"来区分，从而添加不同的标志。

4.定义生成JSON格式的日志文件：

```
def generate_json_log(directory, log_file):
    # 读取日志文件内容
    with open(log_file, "r", encoding="utf-8") as f:
        log_content = f.readlines()  # 读取所有行

    # 生成 JSON 格式的日志
    json_log = json.dumps({"log_content": log_content}, indent=4,
ensure_ascii=False)

    json_log_file = os.path.join(directory, "log.json")  # JSON 文件路径
    with open(json_log_file, "w", encoding="utf-8") as log:
        log.write(json_log)

    print(f"生成JSON日志的路径为：{json_log_file}")  # 打印 JSON 日志的路径
```

这里使用JSON.dumps()将原先的日志内容转换为JSON格式。

```
ensure_ascii=False
```

这是因为默认情况下，`json.dumps()` 会将所有非 ASCII 字符转义成 Unicode 码,这一行代码可以使非 ASCII 字符会保留原始形态，让JSON日志的可读性更强。

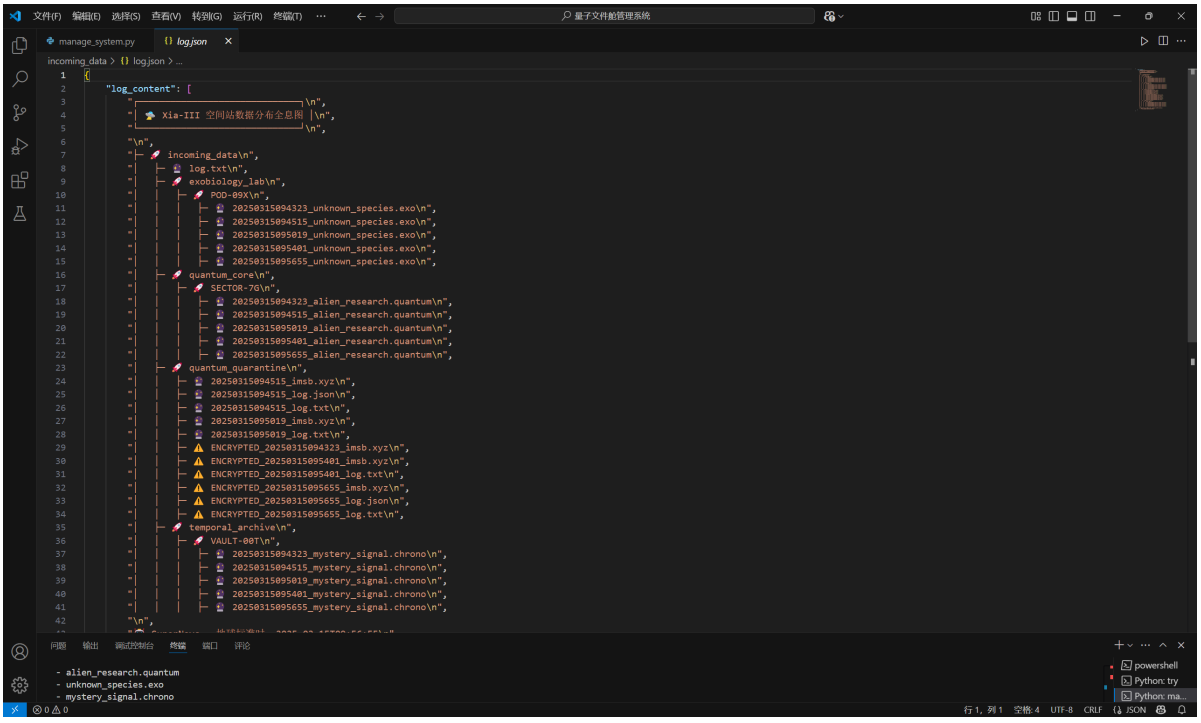
5.生成一个测试文件：

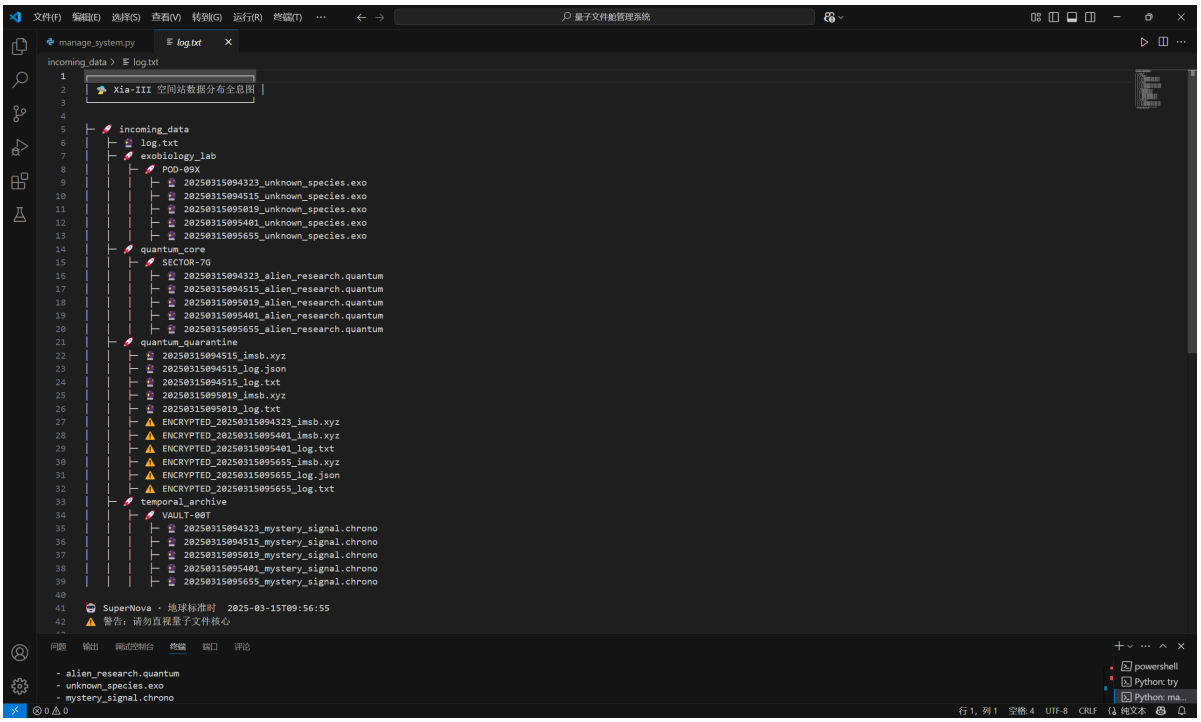
```
def generate_test_input(directory):
    # 使用题干中给的示例文件名
    files = [
        "alien_research.quantum",
        "unknown_species.exo",
```

```
        "mystery_signal.chrono",
        "imsb.xyz"
    ]

    # 创建目录（如果不存在）
    os.makedirs(directory, exist_ok=True)

    # 生成文件
    for file in files:
        file_path = os.path.join(directory, file)
        with open(file_path, 'w') as f:
            f.write(f"Sample content for {file}\n")

    print(f"已在 '{directory}' 中生成以下文件：")
    for file in files:
        print(f" - {file}")
```

3.代码运行结果：

```
已在 'incoming_data' 中生成以下文件：
 - alien_research.quantum
 - unknown_species.exo
 - mystery_signal.chrono
 - imsb.xyz
移动 alien_research.quantum 至 incoming_data\quantum_core\SECTOR-7G\20250315095655_alien_research.quantum
移动 log.txt 至 incoming_data\quantum_quarantine\ENCRYPTED_20250315095655_log.txt
移动 mystery_signal.chrono 至 incoming_data\temporal_archive\VAULT-00T\20250315095655_mystery_signal.chrono
移动 unknown_species.exo 至 incoming_data\exobiology_lab\POD-09X\20250315095655_unknown_species.exo
生成JSON日志的路径为：incoming_data\log.json
```




文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T)
量子文件舱管理系统
manage_system.py
log.txt
incoming_data > log.txt
Xia-III 空间站数据分布全息图
├─ incoming_data
│ ├─ log.txt
│ ├─ exobiology_lab
│ │ ├─ POD-09X
│ │ │ ├─ 20250315094323_unknown_species.exo
│ │ │ ├─ 20250315094515_unknown_species.exo
│ │ │ ├─ 20250315095019_unknown_species.exo
│ │ │ ├─ 20250315095401_unknown_species.exo
│ │ │ ├─ 20250315095655_unknown_species.exo
│ ├─ quantum_core
│ │ ├─ SECTOR-7G
│ │ │ ├─ 20250315094323_alien_research.quantum
│ │ │ ├─ 20250315094515_alien_research.quantum
│ │ │ ├─ 20250315095019_alien_research.quantum
│ │ │ ├─ 20250315095401_alien_research.quantum
│ │ │ ├─ 20250315095655_alien_research.quantum
│ ├─ quantum_quarantine
│ │ ├─ 20250315094515_imsb.xyz
│ │ ├─ 20250315094515_log.json
│ │ ├─ 20250315094515_log.txt
│ │ ├─ 20250315095019_imsb.xyz
│ │ ├─ 20250315095019_log.txt
│ │ ├─ ENCRYPTED_20250315094323_imsb.xyz
│ │ ├─ ENCRYPTED_20250315095401_imsb.xyz
│ │ ├─ ENCRYPTED_20250315095401_log.txt
│ │ ├─ ENCRYPTED_20250315095655_imsb.xyz
│ │ ├─ ENCRYPTED_20250315095655_log.json
│ │ ├─ ENCRYPTED_20250315095655_log.txt
│ ├─ temporal_archive
│ │ ├─ VAULT-00T
│ │ │ ├─ 20250315094323_mystery_signal.chrono
│ │ │ ├─ 20250315094515_mystery_signal.chrono
│ │ │ ├─ 20250315095019_mystery_signal.chrono
│ │ │ ├─ 20250315095401_mystery_signal.chrono
│ │ │ ├─ 20250315095655_mystery_signal.chrono
SuperNova · 地球标准时 2025-03-15T09:56:55
警告：请勿直视量子文件核心
问题 输出 调试控制台 终端 端口 评论
- alien_research.quantum
- unknown_species.exo
- mystery_signal.chrono
powershell
Python: try
Python: ma...
行 1，列 1 空格: 4 UTF-8 CRLF 纯文本