python基础6：文件

一、文件操作

1.【知道】文件的作用

- 文件的作用：**持久化存储**数据

2.【重点】文件基本操作

1. 文件操作流程
    1. 打开文件，或者新建立一个文件
    2. 读/写数据
    3. 关闭文件
2. 打开文件：`文件变量 = open("文件名字", "访问模式")`
3. 关闭文件：`文件变量.close()`

```
"""操作文件的流程：
1. 打开文件
2. 读或写文件
```

```
3. 关闭文件
"""

# 打开文件格式
# 注意点：文件名字，打开权限都是字符串格式
# 文件变量 = open(文件名字，访问模式)

# 'w'：只写方式打开文件，文件不存在创建，文件存在，清空文件内容
f = open("xxx.txt", "w")

# 关闭文件，为了释放资源
# 文件变量.close()
f.close()
```

2.1. 【记忆】文件自动关闭

- 格式:

```
with open('文件名称', '访问模式') as 文件变量名:
    # 文件操作
    pass
```

```
# 只要文件打开了，运行完with语句，自动关闭文件
# with open() as 文件变量别名:
#    文件的操作

with open("abc.txt", "w") as f:
    # 文件操作，执行完代码块，文件自动关闭
    pass
```

3.【重点】文件写操作

- 格式:

```
文件变量.write("需要写入的内容")
```

```
"""
# 1. 打开文件，只写方式打开
# 2. 写文件
# 3. 关闭文件
"""

# 1. 打开文件，只写方式打开
f = open("xxx.txt", "w")


# 2. 写文件
```

```
# 写文件格式：文件变量.write(所需的内容)
f.write("hello abc")
f.write(" hello python\n")
f.write(" hello tom")


# 3. 关闭文件
f.close()
```

4.【重点】文件读操作

4.1. read指定读取内容：

- `读取的内容 = 文件变量.read(读写字符长度)`，n 为读取的字符数，不设置则全部读取

```
# 1. 打开文件，只读方式打开，'r'
# 2. 读取文件内容
# 3. 关闭文件



# 1. 打开文件，只读方式打开，'r'
# 'r'：  打开文件，必须存在，不存在，报错崩溃
f = open("xxx.txt", "r")

# 2. 读取文件内容
```

```
# 格式:  内容变量 = 文件变量.read(读取的长度)
#        如果read的长度不指定，默认读取全部
ret = f.read(4)
print(ret)

ret = f.read(7)
print(ret)

ret = f.read()
print(ret)

# 3. 关闭文件
f.close()
```

扩展: encoding="编码方式"

```
# 1. 打开文件，只读方式打开，'r'
# 2. 读取文件内容
# 3. 关闭文件



# 1. 打开文件，只读方式打开，'r'
# 'r':  打开文件，必须存在，不存在，报错崩溃
# f = open("xxx.txt", "r")  # 默认windows的python使用open打开文件是GBK (国标简繁体中文编码)
```

```python
f = open("xxx.txt", "r", encoding="utf-8")  # encoding使用关键字传递参数,可以指定打开文件的编码方式使用utf-8(国际编码)


# f.write("hello")  # io.UnsupportedOperation: not writable

# 2. 读取文件内容
# 格式:  内容变量 = 文件变量.read(读取的长度)
#        如果read的长度不指定, 默认读取全部
ret = f.read(4) # 4: 4个字,包括中文字数
print(ret)

ret = f.read(7)
print(ret)

ret = f.read()
print(ret)

# 3. 关闭文件
f.close()
```

4.2. readlines读取所有行：

- `内容列表变量 = 文件变量.readlines()`

```
# 1. 打开文件，只读方式打开，'r'
# 2. 读取文件内容
# 3. 关闭文件


# 1. 打开文件，只读方式打开，'r'
# 'r'： 打开文件，必须存在，不存在，报错崩溃
f = open("xxx.txt", "r", encoding="utf-8")

# 2. 读取文件内容
# readlines: 读取所有的行，按行作为分隔条件
# 格式：内容列表变量 = 文件变量.readlines()
r_list = f.readlines()
print(r_list) # ['你好 张三\n', 'hello abc\n', 'hello python\n', 'hello tom\n', 'hello rose\n', 'hello mike']


# 通过for取出列表的所有元素
for row in r_list:
    # print(row)
```

```
    print(row, end="")  # print不加换行的输出

# 3. 关闭文件
f.close()
```

4.3. readline一次读取一行：

- `内容变量 = 文件变量.readline()`

```
# 1. 打开文件，只读方式打开，'r'
# 2. 读取文件内容
# 3. 关闭文件


# 1. 打开文件，只读方式打开，'r'
# 'r'： 打开文件，必须存在，不存在，报错崩溃
f = open("xxx.txt", "r", encoding="utf-8")

# 2. 读取文件内容
# readlines： 读取所有的行
# readline:  一次读取一行
# readline格式：内容变量 = 文件变量.readline()
# ret = f.readline()
```

```
# print(ret)
#
# ret = f.readline()
# print(ret)
#
# ret = f.readline()
# print(ret)

while True:
    ret = f.readline()
    # if ret == "": # "" 表示假
    if not ret:      # not 假 --> 真
        break
    print(ret)

# 3. 关闭文件
f.close()
```

5.【记忆】打开文件详解

5.1 【记忆】访问模式`r`,`w`,`a`的区别

- 只读方式打开文件，文件不存存在，报错

```
# 文件打开访问模式
# 文件变量 = open(文件名字，访问模式)
```

```
# 'r', 只读方式打开文件，文件不存存在，报错
# 注意1: open 默认打开方式就是"r"
# f = open("abc.txt")

# 注意2: "r": 文件不存在,会报错
# f = open("abc.txt", "r")  #
FileNotFoundError: [Errno 2] No such file
or directory: 'abc.txt'
f = open("abc.txt", "r")

# 注意3: "r": 只读打开文件,不能写
# f.write("hello world")  #
io.UnsupportedOperation: not writable

ret = f.read()
print(ret)

f.close()
```

- 只写方式打开文件，文件不存在新建，文件存在清空文件内容

```
# 文件打开访问模式
# 文件变量 = open(文件名字，访问模式)

# 'w'，只写方式打开文件，文件不存在新建，文件存在清空文件内容
# 1.文件不存在新建
# 2.文件存在清空文件内容
f = open("abc.txt", "w")

# 3."w": 只写打开,不能读
# ret = f.read()  # io.UnsupportedOperation: not readable

f.close()
```

- 追加方式打开文件，文件不存在新建，文件存在写光标则放在文件末尾，写数据直接写在文件末尾

```
# 追加方式打开文件，文件不存在新建，文件存在写光标放在文件末尾，写数据直接写在文件末尾

# 1. 'a'，追加方式打开文件
# 1) 文件不存在新建
# 2) 文件存在写光标放在文件末尾
f = open("abc.txt", "a")
```

```
# 3) "a": 不能读
f.read()  # io.UnsupportedOperation: not readable

# 2. 写数据
f.write("hello python")

# 3. 关闭文件
f.close()
```

5.2 【记忆】绝对路径和相对路径的区别

- 绝对路径：是指文件在硬盘上真正存在的路径，是电脑完整的路径

```
# 1. 绝对路径下，打开文件，如果绝对路径E:/Code/PyCode不存在，打开失败
# print("\\")   # 字符串中\ 表示转义字符，\n: 换行 \t: tab键, .... \\: \字符
#
print("C:\\Users\\35477\\Desktop\\python40\\day06\\"+"abc.txt")

# 绝对路径：文件在磁盘中的完整路径，从盘符算起的路径.
```

```
# f =
open("C:\\Users\\35477\\Desktop\\python40
\\day06\\"+"abc.txt", "r")
f =
open("C:/Users/35477/Desktop/python40/day
06/"+"abc.txt", "r")


ret = f.read()
print(ret)

f.close()
```

- 相对路径：相对于自己的目标文件位置
  - `1.txt`：等价于`./1.txt`，当前路径下的`1.txt`
  - `../1.txt`：上一级路径下的`1.txt`

```
# 2. 相对路径下打开文件，相对于当前的目标文件，
就是当前py文件所在路径
# 2.1 当前路径下，如果文件不存在，新建文件
# f = open("abc.txt", "r")
f = open("./abc.txt", "r")  # ./表示当前路
径

ret = f.read()
```

```
print(ret)

f.close()

# 2.2 上一级路径下，如果文件不存在，新建文件
f = open("../abc.txt", "w")  # ../表示当前路径的上一级路径

f.close()
```

6.【应用】文件备份案例

- 实现思路：

```
1. 只读方式打开源文件(旧文件)
2. 只写方式打开新的备份文件(新文件)
3. while True:
    4. 一次读取一点源文件的内容
    5. 如果读取的内容，往写文件中写
    6. 如果没有读到内容，文件已经读取完毕，break跳出循环
7. 关闭文件
```

- 普通版本：

```
"""
# 1. 只读方式打开源文件                abc.txt
```

```
# 2. 只写方式打开新的备份文件        abc[备份].txt
# 3. 一次性读取源文件的内容          read()
# 4. 读取的内容，往写文件中写        write()
# 5. 关闭文件                       close()

# 注意：一次性读取源文件内容需要大量内存开销,可能造成内存不足,无法拷贝.
"""

# 1. 只读方式打开源文件              abc.txt
f_src = open("abc.txt", "r")

# 2. 只写方式打开新的备份文件        abc[备份].txt
f_dest = open("abc[备份].txt", "w")

# 3. 一次性读取源文件的内容          read()
data = f_src.read()

# 4. 读取的内容，往写文件中写        write()
f_dest.write(data)

# 5. 关闭文件                       close()
f_src.close()
f_dest.close()
```

- 分片读取

```
"""
# 1. 只读方式打开源文件(旧文件)
# 2. 只写方式打开新的备份文件(新文件)
# 3. while True:
    # 4. 一次读取一点源文件的内容
    # 5. 如果读取的内容，往写文件中写
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
# 7. 关闭文件

# 注意：一次读取一点源文件的内容,节约内存开销.
"""

# 1. 只读方式打开源文件(旧文件)
old_file = open("abc.txt", "r")

# 2. 只写方式打开新的备份文件(新文件)
new_file = open("abc[备份].txt", "w")

# 3. while True:
while True:
    # 4. 一次读取一点源文件的内容 (一次读1024个
字符)
    ret = old_file.read(1024)
```

```
    # 5. 如果读取的内容，往写文件中写
    new_file.write(ret)
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
    # " " : 有空格字符的字符串
    # "" : 空字符串
    if ret == "":    # if not ret:
        break

# 7. 关闭文件
old_file.close()
new_file.close()
```

- 获取备份文件名

```
# 拷贝的文件名，不要写死，需要用户输入 input
# 新文件的名字变成：旧文件名[备份].txt

# 1. 字符串找.的位置，rfind，  r为right，从右往
左找
file_name = "a.b.c.txt"

pos = file_name.rfind(".")
# print(pos)

# 2. 通过切片提取所要的字符串
```

```
l_file_name = file_name[:pos]
r_file_name = file_name[pos:]
# print(l_file_name)
# print(r_file_name)
new_file_name = l_file_name + '[备份]' +
r_file_name
print(new_file_name)
```

- 修改文件名的拷贝版本

```
# 拷贝的文件名，不要写死，需要用户输入 input
# 新文件的名字变成：旧文件名[复制].txt

"""
# a. 输入需要拷贝的文件名 old_file_name =
input()
# b. 找文件名字右边的第一个点，通过切片组装成： 旧
文件名[备份].后缀

# 1. 只读方式打开源文件(旧文件)
# 2. 只写方式打开新的备份文件(新文件)
# 3. while True:
    # 4. 一次读取一点源文件的内容
    # 5. 如果读取的内容，往写文件中写
```

```
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
# 7. 关闭文件
"""

# a. 输入需要拷贝的文件名 old_file_name =
input()
old_file_name = input("输入需要拷贝的文件名:
")
# b. 找文件名字右边的第一个点，通过切片组装成：  旧
文件名[备份].后缀
pos = old_file_name.rfind(".")
l_file_name = old_file_name[:pos]
r_file_name = old_file_name[pos:]
new_file_name = l_file_name + "[备份]" +
r_file_name

# 1. 只读方式打开源文件(旧文件)
old_file = open(old_file_name, "r")
# 2. 只写方式打开新的备份文件(新文件)
new_file = open(new_file_name, "w")
# 3. while True:
while True:
    # 4. 一次读取一点源文件的内容
    ret = old_file.read(1024)
    # 5. 如果读取的内容，往写文件中写
```

```
    new_file.write(ret)
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
    if ret == "":
        break
# 7. 关闭文件
old_file.close()
new_file.close()
```

- 二进制版本拷贝文件[扩展]

```
# 拷贝的文件名，不要写死，需要用户输入 input
# 新文件的名字变成：旧文件名[复制].txt

"""
# a. 输入需要拷贝的文件名 old_file_name =
input()
# b. 找文件名字右边的第一个点，通过切片组装成： 旧
文件名[备份].后缀

# 1. 只读方式打开源文件(旧文件)
# 2. 只写方式打开新的备份文件(新文件)
# 3. while True:
    # 4. 一次读取一点源文件的内容
    # 5. 如果读取的内容，往写文件中写
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
```

```
# 7. 关闭文件
"""

# a. 输入需要拷贝的文件名 old_file_name = input()
old_file_name = input("输入需要拷贝的文件名:")
# b. 找文件名字右边的第一个点，通过切片组装成：旧文件名[备份].后缀
pos = old_file_name.rfind(".")
l_file_name = old_file_name[:pos]
r_file_name = old_file_name[pos:]
new_file_name = l_file_name + "[备份]" + r_file_name

# 1. 只读方式打开源文件(旧文件)
old_file = open(old_file_name, "rb")    # rb: 二进制方式只读打开(不用考虑编码)
# 2. 只写方式打开新的备份文件(新文件)
new_file = open(new_file_name, "wb")    # wb: 二进制方式只写打开(不用考虑编码)
# 3. while True:
while True:
    # 4. 一次读取一点源文件的内容
    ret = old_file.read(1024)
    # 5. 如果读取的内容，往写文件中写
```

```
    new_file.write(ret)
    # 6. 如果没有读到内容，文件已经读取完毕，
break跳出循环
    if not ret:  # ret: 读到为空，表示
False(假), not False 表示  True(真)
        break
# 7. 关闭文件
old_file.close()
new_file.close()
```

7.【记忆】文件相关操作

- 文件重命名：`os.rename(旧的文件名，新的文件名)`
- 改变默认目录：`os.chdir(改变的路径)`
- 获取目录列表：`目录列表变量 = os.listdir(指定某个目录)`
- 判断文件是否存在：`os.path.exists(需要判断的文件)`

```
"""
1. 导入模块，只需导入一次即可
import os
```

```
2. 使用os中的方法，完成功能
os.方法名()
"""
import os

# 1. 给文件重新命名
# 格式： os.rename(旧的文件名，新的文件名)
# os.rename("abc.txt", "abc-最终版.txt")

# 2. 删除文件
# 格式： os.remove(待删除的文件名)
# os.remove("abc[备份].txt")

# 3. 创建文件夹，只能创建文件夹，不能创建普通文件
# 格式： os.mkdir(文件夹的名字)
# os.mkdir("张三")

# 4. 删除文件夹，只能删除空的文件夹
# 格式：os.rmdir(待删除文件夹的名字)
# os.rmdir("张三")

# 5. 获取当前工作的路径
# 格式：路径变量 = os.getcwd()
# C:\Users\35477\Desktop\python40\day06
path = os.getcwd()  # current work dir
```

```
print(path)  #
C:\Users\35477\Desktop\python40\day06

# 6. 改变路径
# 格式：os.chdir(改变的路径)
# 切换到上一级路径
os.chdir("../")
path2 = os.getcwd()
print(path2)

os.chdir(path)

path2 = os.getcwd()
print(path2)


# 7. 【重点】获取某个目录的文件信息，获取文件夹或文件的名字
# 格式：目录列表变量 = os.listdir(指定某个目录)
#       如果不指定目录，默认当前路径
file_list = os.listdir()
print(file_list)

file_list2 = os.listdir("pytest")
print(file_list2)
```

```
# 8. 判断文件是否存在
# 语法格式：os.path.exists(需要判断的文件)
# 文件存在返回True，文件不存在返回False
ret = os.path.exists("info.txt")
print(ret)
```

7.1. 【应用】批量修改文件名

- 批量给文件添加前缀

```
"""
# 1. 获取pytest的目录信息，返回文件名的列表
# 2. 切换到目标路径
# 3. for遍历文件名的列表取出某个元素
# 4. 对这个元素重命名
"""
import os

# 1. 获取pytest的目录信息，返回文件名的列表
name_list = os.listdir("pytest")
# print(name_list)    # ['lily.txt',
'mike.txt', 'rock.txt', 'tom.txt',
'yoyo.txt']
# 2. 切换到目标路径
os.chdir("pytest")
```

```
# 3. for遍历文件名的列表取出某个元素
for name in name_list:
    # 4. 对这个元素重命名
    new_name = "[黑马出品]-" + name
    os.rename(name, new_name)

os.chdir("../")
```

- 批量恢复文件名

```
"""
# 1. 获取pytest的目录信息，返回文件名的列表
# 2. 切换到目标路径
# 3. for遍历文件名的列表取出某个元素
# 4. 对这个元素重命名
"""
import os

# 1. 获取pytest的目录信息，返回文件名的列表
name_list = os.listdir("pytest")
# 2. 切换到目标路径
os.chdir("pytest")
# 3. for遍历文件名的列表取出某个元素
for name in name_list:
    # 4. 对这个元素重命名
    tmp_str = "[黑马出品]-"
```

```
    _len = len(tmp_str)
    new_name = name[_len:]

    os.rename(name, new_name)

os.chdir("../")
```

8.【记忆】字符串、容器类型相互转换

函数	说明
str(容器变量)	将 容器变量 转换为一个字符串
eval(字符串内容)	返回传入字符串内容的结果，字符串里面看到像是什么，就转换成什么

```
# 字符串、容器类型相互转换
# str(容器变量)：将 容器变量 转换为一个字符串
# eval(字符串内容)：返回传入字符串内容的结果，字符串里面看到是什么，就转换成什么

# 列表
```

```
user_list = [{"name": "rose", "age": 20,
"sex": "male"},
             {"name": "tom", "age": 21,
"sex": "male"}]

# 列表转字符串
my_str = str(user_list)
print(type(my_str), my_str)
# f = open("abc.txt", "w")
# f.write(user_list)  # TypeError: write()
argument must be str, not list
# f.close()

# 字符串
my_str2 = "[{'name': 'xiaoming', 'age':15,
'sex':'male'}]"
# 字符串转列表
# eval(),  看到像什么，就转换什么
my_list = eval(my_str2)
print(type(my_list), my_list,
type(my_list[0]), my_list[0])
```

8.1. 【记忆】把学生名片保存到文件中

```
"""
# 1. 通过with 只写方式打开文件，with里面的语句执行完，自动处理关闭
# 2. 将列表转换为字符串后，再往文件中写
"""

user_list = [{"name": "rose", "age": 20, "sex": "male"},
             {"name": "tom", "age": 21, "sex": "male"}]

# 1. 通过with 只写方式打开文件，with里面的语句执行完，自动处理关闭
with open("info.txt", "w") as file:
    # 2. 将列表转换为字符串后，再往文件中写
    file.write(str(user_list))
```

8.2. 【记忆】把学生名片信息从文件中读取写入列表中

```
"""
# 1. 定义空列表，用户后续做学生信息保存
# 2. 通过with打开文件，只读方式打开文件
```

```
# 3. 文件变量.read()读取所有内容，返回内容是字符串类型
# 4. 把读取内容通过eval转换成列表，给前面的列表赋值
# 5. 在with的外面，打印列表内容
"""

# 1. 定义空列表，用户后续做学生信息保存
# user_list = []
# 2. 通过with打开文件，只读方式打开文件
with open("info.txt", "r") as file:
    # 3. 文件变量.read()读取所有内容，返回内容是字符串类型
    data = file.read()
    # 4. 把读取内容通过eval转换成列表，给前面的列表赋值
    user_list = eval(data)
# 5. 在with的外面，打印列表内容
print(type(user_list), user_list)
```

9.【应用】文件版学生名片管理系统

9.1 【应用】学生名片信息保存到文件的实现思路

1. 通过with 只写方式打开文件，with里面的语句执行完，自动处理关闭
2. 将列表转换为字符串后，再往文件中写

```
# 1. 名片信息保存到文件
def save_stu():
    # 1.1 菜单增加保存信息的提示
    # 1.2 主逻辑增加 名片信息保存到文件 的选择
    # 1.3 名片信息保存到文件 函数的设计
    with open("stu.txt", "w") as file:
        file.write(str(user_list))
```

9.2 【应用】从文件中读取学生名片信息到名片系统中

0. 判断文件是否存在，如果不存在，中断函数
1. 通过with打开文件，只读方式打开文件
2. 文件变量.read()读取所有内容，返回内容是字符串类型
3. global 声明全局变量列表
4. 把读取内容通过eval转换成列表，给全局变量列表赋值

```
# 2. 加载数据
def load_stu():
    # 2.1 程序启动时，循环的前面，调用加载数据函数
    # 2.2 判断文件是否存在，不存在，中断函数
    if os.path.exists("stu.txt"):
        # 2.3 文件存在的时候，加载数据
        # 2.4 通过with打开文件，只读方式打开文件
        with open("stu.txt", "r") as file:
            # 2.5 文件变量.read()读取所有内容，返回内容是字符串类型
            data = file.read()
            # 2.6 global 声明user_list全局变量
            global user_list
            # 2.7 把读取内容通过eval转换成列表，给全局变量的列表赋值
            user_list = eval(data)
```