

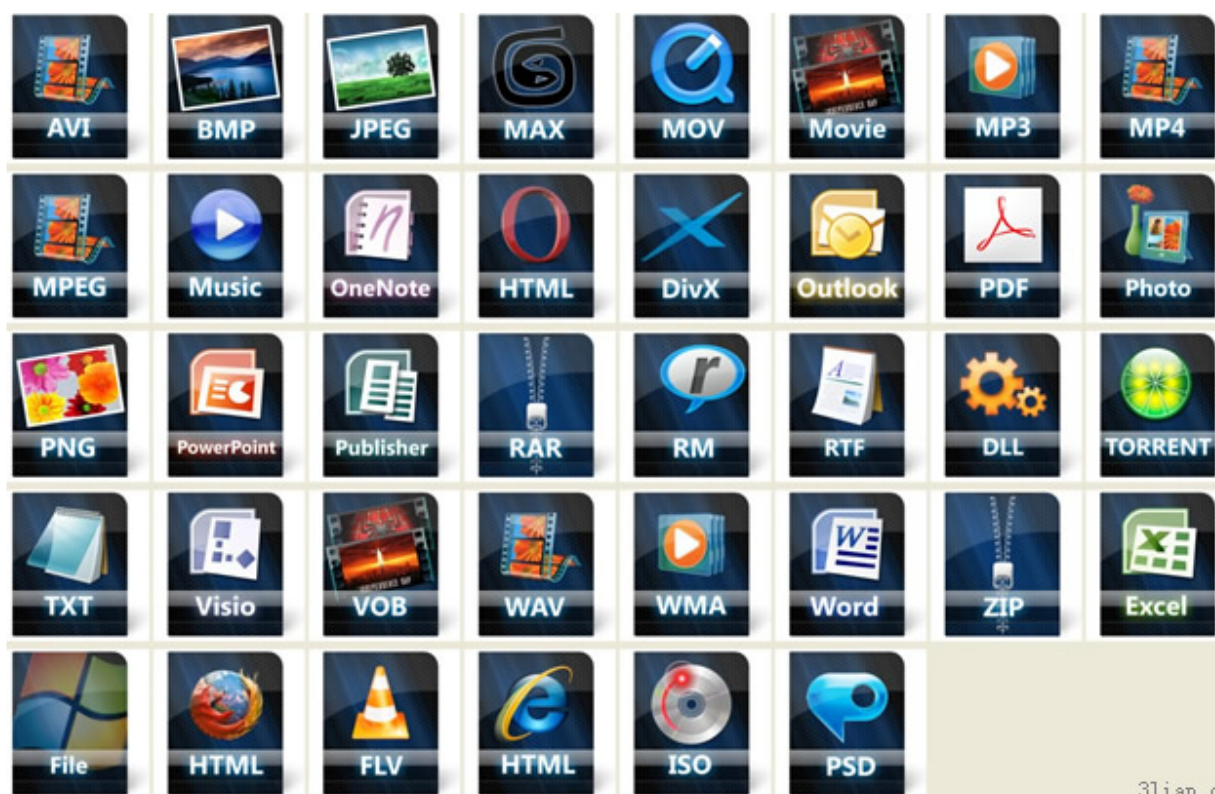
课程：文件操作

目标

- 文件操作的作用
- 文件的基本操作
 - 打开
 - 读写
 - 关闭
- 文件备份
- 文件和文件夹的操作

一. 文件操作的作用

思考：什么是文件？



3lian.com

思考：文件操作包含什么？

答：打开、关闭、读、写、复制....

思考：文件操作的的作用是什么？

答：读取内容、写入内容、备份内容.....

总结：文件操作的作用就是把一些内容(数据)存储存放起来，可以让程序下一次执行的时候直接使用，而不必重新制作一份，省时省力。

二. 文件的基本操作

2.1 文件操作步骤

1. 打开文件
2. 读写等操作
3. 关闭文件

注意：可以只打开和关闭文件，不进行任何读写操作。

2.1.1 打开

在python，使用open函数，可以打开一个已经存在的文件，或者创建一个新文件，语法如下：

```
1 | open(name, mode)
```

name：是要打开的目标文件名的字符串(可以包含文件所在的具体路径)。

mode：设置打开文件的模式(访问模式)：只读、写入、追加等。

2.1.1.1 打开文件模式

模式	描述
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。
w	打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
w+	打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会追加模式。如果该文件不存在，创建新文件用于读写。
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。

2.1.1.2 快速体验

```
1 f = open('test.txt', 'w')
```

注意：此时的 `f` 是 `open` 函数的文件对象。

2.1.2 文件对象方法

2.1.2.1 写

- 语法

```
1 对象对象.write('内容')
```

- 体验

```
1 # 1. 打开文件
2 f = open('test.txt', 'w')
3
4 # 2. 文件写入
5 f.write('hello world')
6
7 # 3. 关闭文件
8 f.close()
```

注意：

1. `w` 和 `a` 模式：如果文件不存在则创建该文件；如果文件存在，`w` 模式先清空再写入，`a` 模式直接末尾追加。
2. `r` 模式：如果文件不存在则报错。

2.1.2.2 读

- `read()`

```
1 文件对象.read(num)
```

`num`表示要从文件中读取的数据的长度（单位是字节），如果没有传入`num`，那么就表示读取文件中所有的数据。

- `readlines()`

`readlines`可以按照行的方式把整个文件中的内容进行一次性读取，并且返回的是一个列表，其中每一行的数据为一个元素。

```
1 f = open('test.txt')
2 content = f.readlines()
3
4 # ['hello world\n', 'abcdefg\n', 'aaa\n', 'bbb\n', 'ccc']
5 print(content)
6
7 # 关闭文件
8 f.close()
```

- `readline()`

`readline()`一次读取一行内容。

```
1 f = open('test.txt')
2
3 content = f.readline()
4 print(f'第一行: {content}')
5
6 content = f.readline()
7 print(f'第二行: {content}')
8
9 # 关闭文件
10 f.close()
```

```
C:\Users\黑马程序员\AppData\Local\Programs\Python\Python37\python3.exe
第一行: hello world

第二行: abcdefg

Process finished with exit code 0
```

2.1.2.3 seek()

作用：用来移动文件指针。

语法如下：

```
1 文件对象.seek(偏移量, 起始位置)
```

起始位置：

- 0：文件开头
- 1：当前位置
- 2：文件结尾

2.1.3 关闭

```
1 文件对象.close()
```

三. 文件备份

需求：用户输入当前目录下任意文件名，程序完成对该文件的备份功能(备份文件名为xx[备份]后缀，例如：test[备份].txt)。

3.1 步骤

1. 接收用户输入的文件名
2. 规划备份文件名
3. 备份文件写入数据

3.2 代码实现

1. 接收用户输入目标文件名

```
1 | old_name = input('请输入您要备份的文件名：')
```

2. 规划备份文件名

2.1 提取目标文件后缀

2.2 组织备份的文件名，xx[备份]后缀

```
1 | # 2.1 提取文件后缀点的下标
2 | index = old_name.rfind('.')
3 |
4 | # print(index) # 后缀中的下标
5 |
6 | # print(old_name[:index]) # 源文件名（无后缀）
7 |
8 | # 2.2 组织新文件名 旧文件名 + [备份] + 后缀
9 | new_name = old_name[:index] + '[备份]' + old_name[index:]
10 |
11 | # 打印新文件名（带后缀）
12 | # print(new_name)
```

3. 备份文件写入数据

3.1 打开源文件 和 备份文件

3.2 将源文件数据写入备份文件

3.3 关闭文件

```
1 | # 3.1 打开文件
2 | old_f = open(old_name, 'rb')
3 | new_f = open(new_name, 'wb')
4 |
5 | # 3.2 将源文件数据写入备份文件
6 | while True:
7 |     con = old_f.read(1024)
8 |     if len(con) == 0:
9 |         break
```

```
10     new_f.write(con)
11
12     # 3.3 关闭文件
13     old_f.close()
14     new_f.close()
```

3.3 思考

如果用户输入 `.txt`，这是一个无效文件，程序如何更改才能限制只有有效的文件名才能备份？

答：添加条件判断即可。

```
1  old_name = input('请输入您要备份的文件名：')
2
3  index = old_name.rfind('.')
4
5
6  if index > 0:
7      postfix = old_name[index:]
8
9  new_name = old_name[:index] + '[备份]' + postfix
10
11  old_f = open(old_name, 'rb')
12  new_f = open(new_name, 'wb')
13
14  while True:
15      con = old_f.read(1024)
16      if len(con) == 0:
17          break
18      new_f.write(con)
19
20  old_f.close()
21  new_f.close()
```

四. 文件和文件夹的操作

在Python中文件和文件夹的操作要借助os模块里面的相关功能，具体步骤如下：

1. 导入os模块

```
1  import os
```

2. 使用 `os` 模块相关功能

```
1 | os.函数名()
```

4.1 文件重命名

```
1 | os.rename(目标文件名, 新文件名)
```

4.2 删除文件

```
1 | os.remove(目标文件名)
```

4.3 创建文件夹

```
1 | os.mkdir(文件夹名字)
```

4.4 删除文件夹

```
1 | os.rmdir(文件夹名字)
```

4.5 获取当前目录

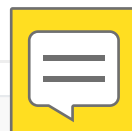
```
1 | os.getcwd()
```

4.6 改变默认目录

```
1 | os.chdir(目录)
```

4.7 获取目录列表

```
1 | os.listdir(目录)
```



五.应用案例

需求：批量修改文件名，既可添加指定字符串，又能删除指定字符串。

- 步骤

1. 设置添加删除字符串的标识
2. 获取指定目录的所有文件
3. 将原有文件名添加/删除指定字符串，构造新名字
4. `os.rename()`重命名

- 代码

```
1  import os
2
3  # 设置重命名标识：如果为1则添加指定字符，flag取值为2则删除指定字符
4  flag = 1
5
6  # 获取指定目录
7  dir_name = './'
8
9  # 获取指定目录的文件列表
10 file_list = os.listdir(dir_name)
11 # print(file_list)
12
13
14 # 遍历文件列表内的文件
15 for name in file_list:
16
17     # 添加指定字符
18     if flag == 1:
19         new_name = 'Python-' + name
20     # 删除指定字符
21     elif flag == 2:
22         num = len('Python-')
23         new_name = name[num:]
24
25     # 打印新文件名，测试程序正确性
26     print(new_name)
27
28     # 重命名
29     os.rename(dir_name+name, dir_name+new_name)
```

六. 总结

- 文件操作步骤

- 打开

```
1 | 文件对象 = open(目标文件, 访问模式)
```

- 操作

- 读

```
1 | 文件对象.read()
2 | 文件对象.readlines()
3 | 文件对象.readline()
```

- 写

```
1 | 文件对象.write()
```

- seek()

- 关闭

```
1 | 文件对象.close()
```

- 主访问模式

- w: 写, 文件不存在则新建该文件
 - r: 读, 文件不存在则报错
 - a: 追加

- 文件和文件夹操作

- 重命名: `os.rename()`
 - 获取当前目录: `os.getcwd()`
 - 获取目录列表: `os.listdir()`