Python程序设计：文件I/O

2023-2024

王斌辉 副教授

南开大学软件学院

文件 I/O

- 文件 (file object)：由文本I/O 、 二进制I/O和原始I/O生成的对象
- 操作文件I/O的流程：
  - 打开文件
  - 读、写文件
    - 读：将文件内容读入内存
    - 写：将内存内容写入文件
  - 关闭文件

- 每类文件对象具有各种功能：可以只读、只写或读写
- 文件对象允许任意随机访问(向前或向后寻找任何位置)，或仅允许顺序访问(例如在套接字或管道情况下)

	函数/方法	说 明
1	open(函数)	打开文件，并且返回文件操作对象
2	read	将文件内容读取到内存
3	write	将指定内容写入文件
4	close	关闭文件

类文件对象 file-like Object

- 像open()函数返回的这种有个read()方法的对象，称为file-like Object
- 除了file外，还可以是内存的字节流，网络流，自定义流等等
- file-like Object不要求从特定类继承，只要写个read()方法就行
- StringIO就是在内存中创建的file-like Object，常用作临时缓冲

文本 I/O 操作

- 操作 str 对象，数据的编码和解码是透明的，并可选择转换特定平台的换行符

```
# 1.打开文件
file = open("read.txt", "r", encoding="utf-8")
# 2.读取文件内容
text = file.read()
print(text)
# 3.关闭文件
file.close()
```

字符	功能说明
'r'	读取（默认）
'w'	写入，并先截断文件
'x'	写模式，新建一个文件（排它性创建），如果文件已存在则失败
'a'	打开文件用于写入，如果文件存在则在末尾追加
'b'	二进制模式
't'	文本模式（默认）
'+'	打开用于更新（读取与写入）

- 内存中文本流也可以作为 StringIO 对象使用

```
f = io.StringIO("some initial text data")
```

二进制 I/O 操作

- 也称为缓冲I/O，操作 bytes 对象，如图片、视频等。其不执行编码、解码或换行转换
- 创建二进制流的最简单方法是使用 open()，并在模式字符串中指定 'b'

```
f = open("myfile.jpg", "rb")
```

- 内存中二进制流也可以作为 BytesIO 对象使用

```
f = io.BytesIO(b"some initial binary data: \x00\x01")
```

- 其它库模块可以提供额外的方式来创建文本或二进制流。参见 socket.socket.makefile() 的示例

原始 I/O

- 原始 I/O（也称为 非缓冲 I/O）通常用作二进制和文本流的低级构建块
- 用户代码直接操作原始流的用法非常罕见。不过，可以通过在禁用缓冲的情况下以二进制模式打开文件来创建原始流

```
f = open("myfile.jpg", "rb", buffering=0)
```

- RawIOBase 的文档中详细描述了原始流的API

文件 I/O 打开

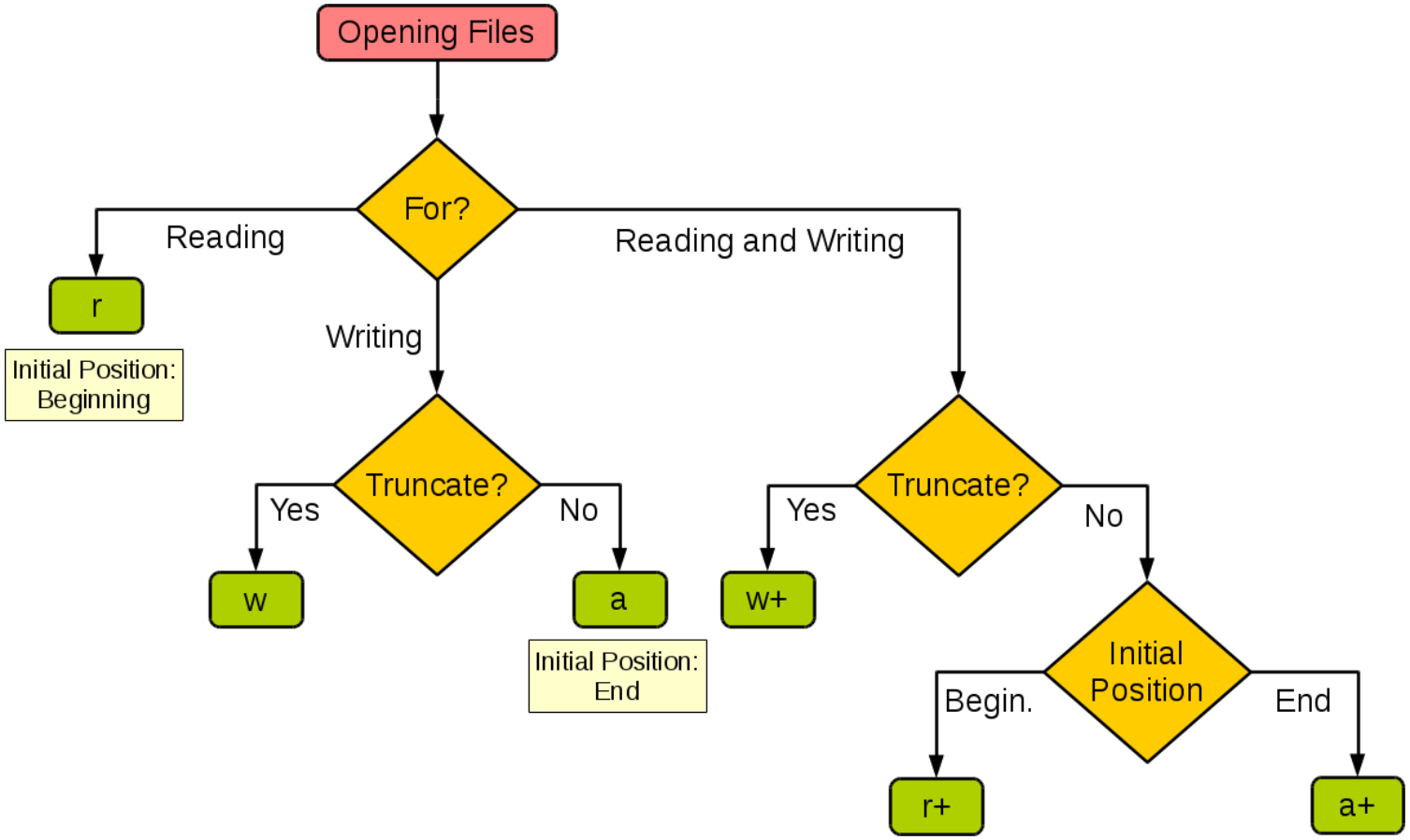
```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
```

- 打开 file 并返回对应的 file object。 如果该文件不能被打开，则引发 OSError
- file 是一个 path-like object，表示将要打开的文件的路径（绝对路径或者相对当前工作目录的路径），也可以是要封装文件对应的整数类型文件描述符。如果给出的是文件描述符，则当返回的 I/O 对象关闭时它也会关闭，除非将 closefd 设为 False 。
  - 读：将文件内容读入内存
  - 写：将内存内容写入文件
- buffering 是一个可选的整数，用于设置缓冲策略。 传入 0 来关闭缓冲（只允许在二进制模式下），传入 1 来选择行缓冲（只在文本模式下可用），传入一个整数 > 1 来表示固定大小的块缓冲区的字节大小。更多注意事项>>
- errors 是一个可选的字符串参数，用于指定如何处理编码和解码错误(不能在二进制模式下使用)
- newline determines how to parse newline characters from the stream

文本 I/O 操作

模式	r	r+	w	w+	a	a+
读	√	√		√		√
写		√	√	√	√	√
创建			√	√	√	√
覆盖			√	√		
指针在开始	√	√	√	√		
指针在结尾					√	√

文本 I/O 操作


Opening Files
For?
Reading
Writing
Reading and Writing
r
Initial Position: Beginning
Truncate?
Yes
No
w
a
Initial Position: End
Truncate?
Yes
No
w+
Initial Position
Begin.
End
r+
a+

文件操作

- open 函数默认以 只读方式 打开文件，并且返回文件对象，其相关属性：

```
f = open("文件名", "访问方式")
```

属性	描述
file.closed	返回true如果文件已被关闭，否则返回false
file.mode	返回被打开文件的访问模式
file.name	返回文件的名称
file.softspace	如果用print输出后，必须跟一个空格符，则返回false。否则返回true

文本 I/O 方法

序号	方法	描述
1	file.close()	关闭文件。关闭后文件不能再进行读写操作。
2	file.flush()	刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件, 而不是被动的等待输出缓冲区写入。
3	file.fileno()	返回一个整型的文件描述符(file descriptor FD 整型), 可以用在如os模块的read方法等一些底层操作上。
4	file.isatty()	如果文件连接到一个终端设备返回 True，否则返回 False。
5	file.next()	返回文件下一行。
6	file.read([size])	从文件读取指定的字节数，如果未给定或为负则读取所有。
7	file.readline([size])	读取整行，包括 "\n" 字符。
8	file.readlines([sizeint])	读取所有行并返回列表，若给定sizeint>0，则是设置一次读多少字节，这是为了减轻读取压力。
9	file.seek(offset[, whence])	设置文件当前位置
10	file.tell()	返回文件当前位置。
11	file.truncate([size])	截取文件，截取的字节通过size指定，默认为当前文件位置
12	file.write(str)	将字符串写入文件，返回的是写入的字符长度。
13	file.writelines(sequence)	向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符。

文件/目录

- os模块执行文件、目录操作：
  - 创建、重命名、删除、修改路径、查看目录内容
  - 文件操作

序号	方法名	说明	示例
1	rename	重命名文件	os.rename(源文件名, 目标文件名)
2	remove	删除文件	os.remove(文件名)

  - 目录操作

序号	方法名	说明	示例
1	listdir	目录列表	os.listdir(目录名)
2	mkdir	创建目录	os.mkdir(目录名)
3	rmdir	删除目录	os.rmdir(目录名)
4	getcwd	获取当前目录	os.getcwd()
5	chdir	修改工作目录	os.chdir(目标目录)
6	path.isdir	判断是否是目录	os.path.isdir(文件路径)

文件/目录

- Python 4 大文件处理库（os、shutil、pathlib、glob）：