08. 万恶之源-文件操作

本节主要内容:

1. 初识文件操作
2. 只读(r, rb)
3. 只写(w, wb)
4. 追加(a, ab)
5. r+读写
6. w+写读
7. a+写读(追加写读)
8. 其他操作方法
9. 文件的修改以及另一种打开文件句柄的方式

主要内容:

一. 初识文件操作

使用python来读写文件是非常简单的操作. 我们使用open()函数来打开一个文件, 获取到文件句柄. 然后通过文件句柄就可以进行各种各样的操作了. 根据打开方式的不同能够执行的操作也会有相应的差异.

打开文件的方式: r, w, a, r+, w+, a+, rb, wb, ab, r+b, w+b, a+b 默认使用的是r(只读)模式

二. 只读操作(r, rb)

```
f = open("护士少妇嫩模.txt",mode="r", encoding="utf-8")
content = f.read()
print(content)
f.close()
```

需要注意encoding表示编码集. 根据文件的实际保存编码进行获取数据, 对于我们而言. 更多的是utf-8.

rb. 读取出来的数据是bytes类型, 在rb模式下. 不能选择encoding字符集.

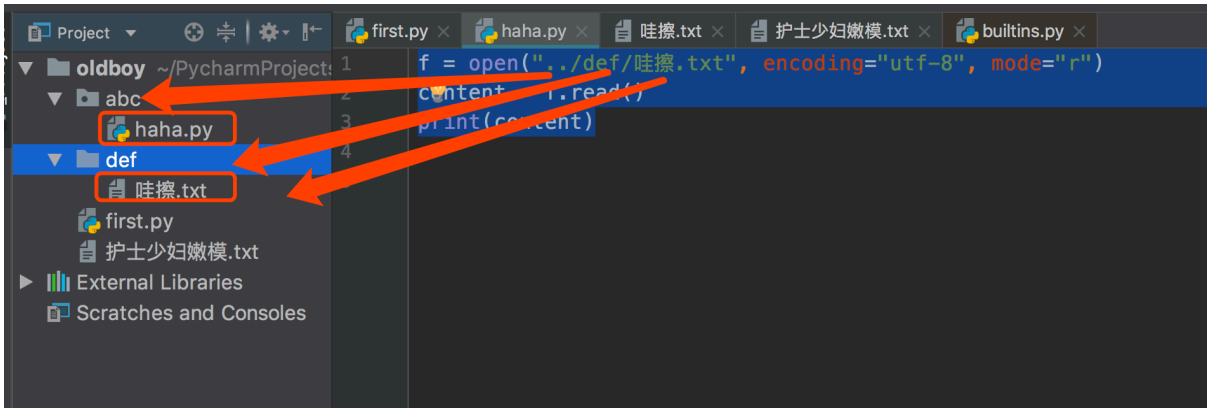
```
f = open("护士少妇嫩模.txt",mode="rb" )
content = f.read()
print(content)
f.close()

结果:
b'\xe6\xaf\x85\xe5\x93\xa5, \xe5\xa4\xaa\xe7\x99\xbd,
wuse\n\xe5\x91\xb5\xe5\x91\xb5\n\xe6\x97\xa5\xe5\xa4\xa9'
```

rb的作用: 在读取非文本文件的时候. 比如读取MP3. 图像. 视频等信息的时候就需要用到rb. 因为这种数据是没办法直接显示出来的. 在后面我们文件上传下载的时候还会用到. 还有.

我们看的直播. 实际上都是这种数据.

绝对路径和相对路径:

1. 绝对路径:从磁盘根目录开始一直到文件名.
2. 相对路径:同一个文件夹下的文件. 相对于当前这个程序所在的文件夹而言. 如果在同一个文件夹中. 则相对路径就是这个文件名. 如果在上一层文件夹. 则要../



我们更推荐大家使用相对路径. 因为在我们把程序拷贝给别人使用的时候. 直接把项目拷贝走就能运行. 但是如果用绝对路径. 那还需要拷贝外部的文件.

读取文件的方法:

1. read() 将文件中的内容全部读取出来. 弊端: 占内存. 如果文件过大.容易导致内存崩溃

```
f = open("../def/哇擦.txt", mode="r", encoding="utf-8")
content = f.read()
print(content)

结果:
友谊地久天长,
爱一点,
可惜我是水瓶座
一生中最爱
```

2. read(n) 读取n个字符. 需要注意的是. 如果再次读取. 那么会在当前位置继续去读而不是从头读, 如果使用的是rb模式. 则读取出来的是n个字节

```
f = open("../def/哇擦.txt", mode="r" encoding="utf-8")
content = f.read(3)
print(content)

结果:
```

```
友谊地

f = open("../def/哇擦.txt", mode="rb")
content = f.read(3)
print(content)

结果:
b'\xe5\x8f\x8b'


f = open("../def/哇擦.txt", mode="r", encoding="utf-8")
content = f.read(3)
content2 = f.read(3)
print(content)
print(content2)

结果:
友谊地
久天长
```

3. readline() 一次读取一行数据, 注意: readline()结尾, 注意每次读取出来的数据都会有一个\n 所以呢. 需要我们使用strip()方法来去掉\n或者空格

```
f = open("../def/哇擦.txt", mode="r", encoding="utf-8")
content = f.readline()
content2 = f.readline()
content3 = f.readline()
content4 = f.readline()
content5 = f.readline()
content6 = f.readline()
print(content)
print(content2)
print(content3)
print(content4)
print(content5)
print(content6)

结果:

友谊地久天长,

爱一点,

可惜我是水瓶座

一生中最爱
```

4. readlines()将每一行形成一个元素, 放到一个列表中. 将所有的内容都读取出来. 所以也是. 容易出现内存崩溃的问题.不推荐使用

```
f = open("../def/哇擦.txt", mode="r", encoding="utf-8")
lst = f.readlines()
print(lst)
for line in lst:
    print(line.strip())
```

5. 循环读取. 这种方式是组好的. 每次读取一行内容.不会产生内存溢出的问题.

```
f = open("../def/哇擦.txt", mode="r", encoding="utf-8")
for line in f:
    print(line.strip())
```

注意: 读取完的文件句柄一定要关闭 f.close()

三. 写模式(w, wb)

写的时候注意. 如果没有文件. 则会创建文件, 如果文件存在. 则将原件中原来的内容删除, 再写入新内容

```
f = open("小娃娃", mode="w", encoding="utf-8")
f.write("金毛狮王")
f.flush()    # 刷新. 养成好习惯
f.close()
```

尝试读一读

```
f = open("小娃娃", mode="w", encoding="utf-8")
f.write("金毛狮王")
f.read()    # not readable 模式是w. 不可以执行读操作
f.flush()
f.close()
```

wb模式下. 可以不指定打开文件的编码. 但是在写文件的时候必须将字符串转化成utf-8的bytes数据

```
f = open("小娃娃", mode="wb")
f.write("金毛狮王".encode("utf-8"))
f.flush()
f.close()
```

四. 追加(a, ab)

只要是a或者ab, a+ 都是在文件的末尾写入. 不论光标在任何位置.

在追加模式下. 我们写入的内容会追加在文件的结尾.

```
f = open("小娃娃", mode="a", encoding="utf-8")
f.write("麻花藤的最爱")
f.flush()
f.close()
```

ab模式自己试一试就好了

五. 读写模式(r+, r+b)

对于读写模式. 必须是先读. 因为默认光标是在开头的. 准备读取的. 当读完了之后再进行写入. 我们以后使用频率最高的模式就是r+

正确操作是:

```
f = open("小娃娃", mode="r+", encoding="utf-8")
content = f.read()
f.write("麻花藤的最爱")
print(content)
f.flush()
f.close()

结果:
正常的读取之后, 写在结尾
```

错误操作:

```
f = open("小娃娃", mode="r+", encoding="utf-8")
f.write("哈哈")
content = f.read()
print(content)
f.flush()
f.close()

结果: 将开头的内容改写成了"哈哈", 然后读取的内容是后面的内容.
```

所以记住: r+模式下. 必须是先读取. 然后再写入

六. 写读(w+, w+b)

先将所有的内容清空. 然后写入. 最后读取. 但是读取的内容是空的, 不常用

```
f = open("小娃娃", mode="w+", encoding="utf-8")
f.write("哈哈")
content = f.read()
print(content)
f.flush()
f.close()
```

有人会说. 先读不就好了么? 错. w+ 模式下, 一开始读取不到数据. 然后写的时候再将原来的内容清空. 所以, 很少用.

七. 追加读(a+)

a+模式下, 不论先读还是后读. 都是读取不到数据的.

```
f = open("小娃娃", mode="a+", encoding="utf-8")
f.write("马化腾")
content = f.read()
print(content)

f.flush()
f.close()
```

还有一些其他的带b的操作. 就不多赘述了. 就是把字符换成字节. 仅此而已

八. 其他相关操作

1. seek(n) 光标移动到n位置，注意, 移动的单位是byte. 所以如果是UTF-8的中文部分要是3的倍数.

   通常我们使用seek都是移动到开头或者结尾.

   移动到开头: seek(0)

   移动到结尾: seek(0,2) seek的第二个参数表示的是从哪个位置进行偏移, 默认是0，表示开头, 1表示当前位置, 2表示结尾

```
f = open("小娃娃", mode="r+", encoding="utf-8")
f.seek(0)   # 光标移动到开头
content = f.read()  # 读取内容, 此时光标移动到结尾
print(content)
f.seek(0)   # 再次将光标移动到开头
f.seek(0, 2)    # 将光标移动到结尾
content2 = f.read()  # 读取内容. 什么都没有
print(content2)

f.seek(0)   # 移动到开头
f.write("张国荣")  # 写入信息. 此时光标在9  中文3 * 3个 = 9

f.flush()
f.close()
```

2. tell() 使用tell()可以帮我们获取到当前光标在什么位置

```
f = open("小娃娃", mode="r+", encoding="utf-8")
```

```
f.seek(0)   # 光标移动到开头
content = f.read()  # 读取内容，此时光标移动到结尾
print(content)
f.seek(0)   # 再次将光标移动到开头
f.seek(0, 2)    # 将光标移动到结尾
content2 = f.read()  # 读取内容. 什么都没有
print(content2)

f.seek(0)   # 移动到开头
f.write("张国荣")  # 写入信息. 此时光标在9  中文3 * 3个 = 9

print(f.tell())  # 光标位置9

f.flush()
f.close()
```

3. truncate() 截断文件

```
f = open("小娃娃", mode="w", encoding="utf-8")
f.write("哈哈")   # 写入两个字符
f.seek(3)   # 光标移动到3，也就是两个字中间
f.truncate()    # 删掉光标后面的所有内容
f.close()


f = open("小娃娃", mode="r+", encoding="utf-8")
content = f.read(3)  # 读取12个字符
f.seek(4)
print(f.tell())
f.truncate()    # 后面的所有内容全部都删掉
# print(content)
f.flush()
f.close()
```

深坑请注意: 在r+模式下. 如果读取了内容. 不论读取内容多少. 光标显示的是多少. 再写入或者操作文件的时候都是在结尾进行的操作.

所以如果想做截断操作. 记住了. 要先挪动光标. 挪动到你想要截断的位置. 然后再进行截断

关于truncate(n), 如果给出了n. 则从开头开头进行截断, 如果不给n, 则从当前位置截断. 后面的内容将会被删除

九. 修改文件以及另一种打开文件的方式

文件修改: 只能将文件中的内容读取到内存中, 将信息修改完毕, 然后将源文件删除, 将新文件的名字改成老文件的名字.

```
# 文件修改
```

```
import os

with open("小娃娃", mode="r", encoding="utf-8") as f1,\
     open("小娃娃_new", mode="w", encoding="UTF-8") as f2:
    content = f1.read()
    new_content = content.replace("冰糖葫芦", "大白梨")
    f2.write(new_content)
os.remove("小娃娃")    # 删除源文件
os.rename("小娃娃_new", "小娃娃")     # 重命名新文件
```

弊端: 一次将所有内容进行读取. 内存溢出. 解决方案: 一行一行的读取和操作

```
import os

with open("小娃娃", mode="r", encoding="utf-8") as f1,\
     open("小娃娃_new", mode="w", encoding="UTF-8") as f2:
    for line in f1:
        new_line = line.replace("大白梨", "冰糖葫芦")
        f2.write(new_line)
os.remove("小娃娃")    # 删除源文件
os.rename("小娃娃_new", "小娃娃")     # 重命名新文件
```