文件

文件类型

按照文件中数据的组织形式，文件分为文本文件和二进制文件

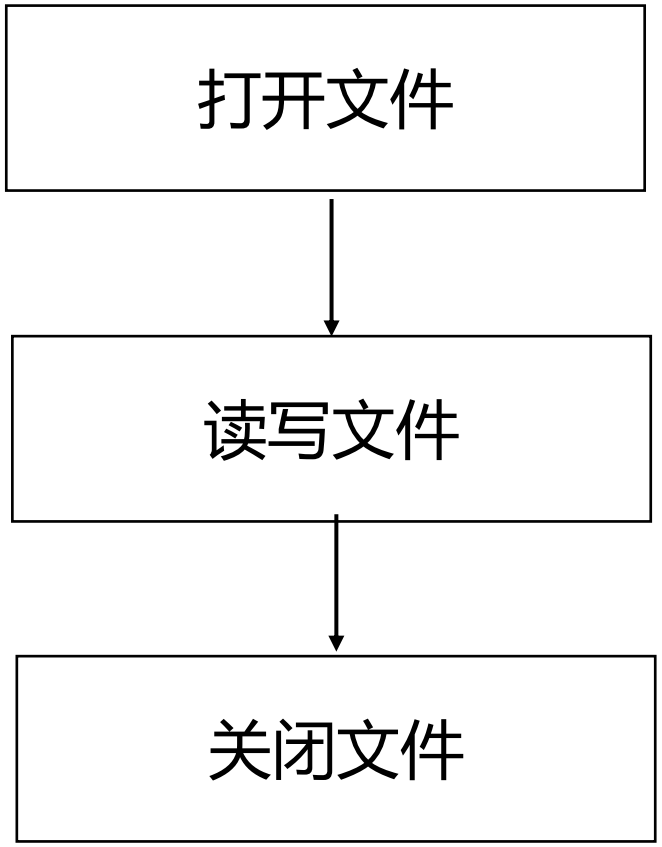
- 文本文件：存储常规字符串，由若干文本行组成。常规字符串指文本编辑器能正常显示、编辑且人能够直接阅读和理解的字符串

- 二进制文件：存储字节串（bytes）。无法用普通文本处理软件进行编辑，通常无法被人直接阅读和理解。需要使用专门的软件进行解码后读取、显示、修改或执行

内容

- 文件的基本操作
- 二进制文件的序列化操作
- 文件的路径
- 文件级操作及目录操作
- 综合案例

文件的基本操作流程

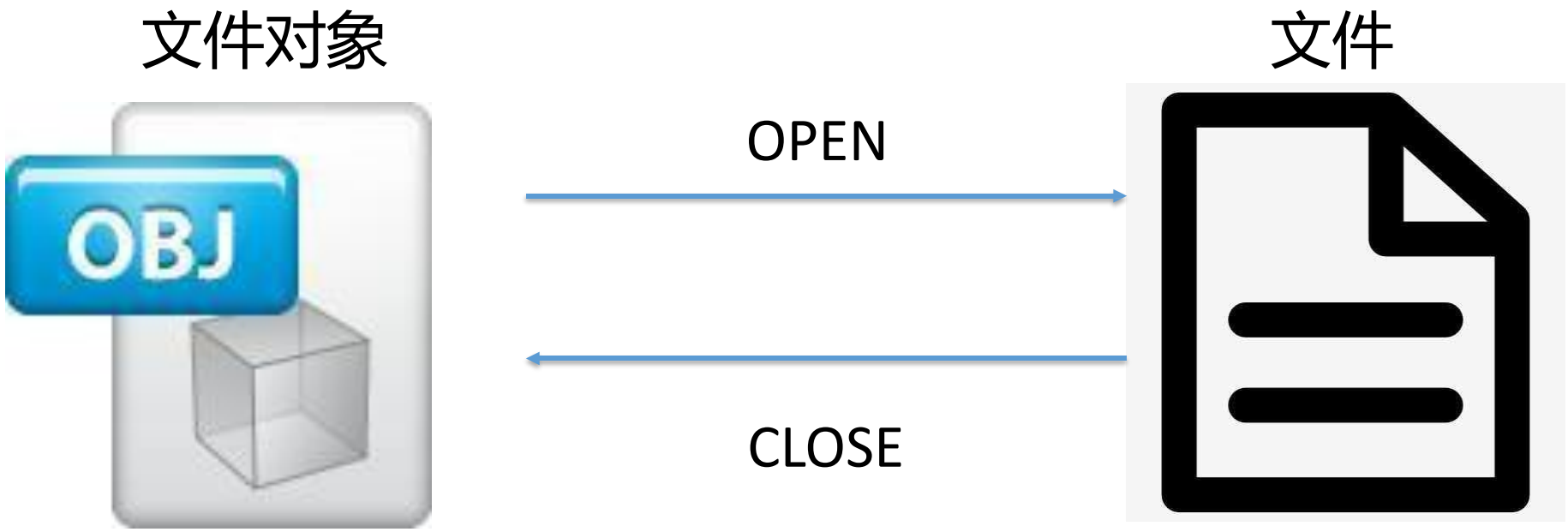


#打开文件，创建文件对象

#读取、写入、删除、修改

#关闭文件，保存文件内容

文件操作的实质



打开指定文件

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

文件打开模式

模式	说明
r	只读方式打开文件，若文件不存在则报错，默认模式
w	只写方式打开文件，若文件存在则覆盖
a	追加方式打开文件，在文件尾部追加写入
x	独占方式创建文件，若文件存在则报错
t	文本模式（需与前面模式组合使用），默认模式
b	二进制模式（需与前面模式组合使用）
+	读写模式（需与前面模式组合使用）

文本文件读出操作

以‘r’、‘r+’、‘w+’、‘a+’方式打开的文本文件，可执行读出操作

方法	说明
fp.read([size])	读出size个字符的内容作为结果返回，不指定size表示读出全部内容
fp.readline([size])	读出一行内容作为结果返回，指定size时与read()方法相同
fp.readlines()	把每行文本作为一个字符串存入列表中，返回该列表
fp.readable()	测试当前文件是否可读，返回True或False

文本文件读出操作

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

```
>>> f1 = open("tinyGrass.txt",'r')
>>> s1=f1.read(6)
>>> print(s1,end="***")
Tiny g***
>>> s2=f1.readline()
>>> print(s2,end="****")
rass,
****
>>> s3=f1.readlines()
>>> for s in s3:
        print(s)
your steps are small,
but you possess the earth
under your tread.
```

tinyGrass.txt

文本文件读出操作

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

tinyGrass.txt

```
f1 = open('tinyGrass.txt')
for s in f1:
    print(s)
f1.close()
```

```
Tiny grass,

your steps are small,

but you possess the earth

under your tread.
```

文本文件写入操作

以‘w’、‘a’、‘x’、‘w+’、‘a+’、‘x+’、‘r+’方式打开的文本文件，可执行写入操作

方法	说明
fp.write (s)	把s的内容写入文件
fp.writelines(s)	把字符串列表写入文本文件，不添加换行符
fp.flush()	强制立即将缓冲区的内容写入文件
fp.writable()	测试当前文件是否可写，返回True或False

文本文件写入操作

```
>>> s1 = "Twinkle, twinkle, little star"
>>> s2 = "How I wonder what you are"
>>> s3=["Up above the world so high","Like a diamond in the sky."]

>>> f2=open("littleStar.txt","w")

>>> f2.write(s1)
29
>>> f2.write(s2)
25

>>> f2.writelines(s3)

>>> f2.close()
```

文本文件写入操作

```
>>> f2=open("littleStar.txt")
>>> s4=f2.readlines()
>>> for s in s4:
        print(s)

Twinkle, twinkle, little starHow I wonder what you areUp above the world so high
Like a diamond in the sky.
```

文件内容的随机访问

方法	说明
fp.seek (offset[,whence])	把文件指针移动到新的位置，offset表示相对于whence的位移（字节数）。whence的值：0——从文件头开始；1——从当前位置开始；2——从文件尾开始。默认为0
fp.tell()	返回文件指针的当前位置
fp.truncate([size])	删除从当前指针位置到文件末尾的内容。如果指定了size，则不论指针在什么地方，只留下前size个字节，其余内容一律删除
fp.seekable()	测试当前文件是否支持随机访问，返回True或False

文件内容的随机访问

随机访问函数可以同时应用于以文本模式或二进制模式打开的文件，但在应用于文本模式打开的文件时，会受到限制

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

```
>>> f3 = open("tinyGrass.txt","r")
>>> f3.read(6)
'Tiny g'
>>> f3.tell()
6
>>> f3.seek(1,1)

Traceback (most recent call last):
  File "<pyshell#109>", line 1, in <module>
    f3.seek(1,1)
io.UnsupportedOperation: can't do nonzero cur-relative seeks
```

文件内容的随机访问

随机访问函数可以同时应用于以文本模式或二进制模式打开的文件，但在应用于文本模式打开的文件时，会受到限制

Tiny grass,
your steps are small,
but you possess the earth
under your tread.

```
>>> f3 = open("tinyGrass.txt","rb")
>>> f3.read(6)
b'Tiny g'
>>> f3.tell()
6
>>> f3.seek(1,1)
7
>>> f3.read(3)
b'ass'
```

```
>>> f3.tell()
10
>>> f3.seek(-5,1)
5
>>> f3.read(5)
b'grass'
```

二进制文件的读写操作

- 对文本文件的读写操作方法同样可用于二进制文件，格式相同
- 文本文件可以用打开二进制文件的方式打开（‘rb’、‘wb’、‘ab’等），并可以使用一般的读写方法进行读写操作，在未出现编码方式错误的前提下，这些操作可以正常进行
- 对于真正的二进制文件，虽然也可以使用一般的读写方法进行读写操作，但直接读取的内容是无法理解的，同样也难以进行有意义的写操作


```
>>> f2 = open("wordhelp.docx","rb")
```

```
>>> b1 = f2.readline()
>>> print(b1)
b'PK\x03\x04\x14\x00\x06\x00\x08\x00\x00\x00!\x00\x81\xff\x9cN\xc1\x01\x00\x00\x
9f\t\x00\x00\x13\x00\x08\x02[Content_Types].xml \xa2\x04\x02(\xa0\x00\x02\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xcc\x96\xcbn\xdb0\x10E\xf7\x05\xfa\x0f
\x02\xb7\x85E\'-\x8a\xa2\xb0\x9cE\x1f\xcb6@S\xa0[\x9a\x1c\xc9L\xf9\x029N\xe2\xbf
\xefP\xb2\x85 P"7\xb6\x821\x04H\x9c{\xef\xe1\x03C-.\xee\xac)n &\xed]\xc5\xce\xca
9+\xc0I\xaf\xb4k*\xf6\xfb\xea\xfb\xec\x13+\x12\n'
```

文件的关闭

fp.close()

在进行文件操作时，如果文件读写后，未能正确关闭一个文件，有可能导致文件损坏、文件内容丢失等问题

出现异常时无法正常关闭文件

```
f4 = open("test.txt","r+")
exit(-1)  #模拟程序异常退出
f4.close() #此句永远无法执行到
```

程序执行过程中异常退出时，无法执行后面的close语句

文件操作异常处理

```
#demo0701.py
import os
try:
    file1 = open(“test.dat”,“w”)
    file1.write(“10000”)
except FileNotFoundError:
    print("File not found error\n")
except PermissionError:
    print("Permission error\n")
except BaseException:
    print("Other unexpected error")
finally:
    file1.close()
```

文件读写操作可能发生异常，通过异常处理的try—except—finally处理异常

捕获三种异常：文件不存在、没有足够的操作权限、其它异常

无论文件操作是否正常，都关闭文件

文件操作的简化方式：with语句

```
with open(file, mode= ‘r’ , buffering=-1,
          encoding=None, errors=None,
          newline=None, closfd=True,
          opener=None) as fp:
```

<文件操作>

with关键字用于对资源进行访问的场合，会确保不管在使用过程中是否发生异常，都会在使用结束后执行必要的清理操作，释放资源，自动关闭文件等

文件对象常用属性

属性	说明
buffer	文件的缓冲区对象
closed	文件是否关闭
fileno	文件的文件号，一般不需要
mode	文件的打开模式
name	文件名称

```
>>> f4.name
'littleStar.txt'
>>> f4.closed
False
>>> f4.buffer
<_io.BufferedReader name='littleStar.txt'>
>>> f4.mode
'r'
>>> f4.fileno
<built-in method fileno of _io.TextIOWrapper object at 0x00000200133FD110>
```

内容

- 文件的基本操作
- 二进制文件的序列化操作
- 文件的路径
- 文件级操作及目录操作
- 综合案例

Python对象的序列化（Serialization）

- 序列化：将内存对象的状态信息转换为可以存储或传输的形式的过程
- 反序列化：将文件中存储或网络传输的对象转换为内存对象的过程

- 常用的Python序列化模块（方法）：pickle, json，marshal, shelve, struct

作用：内存对象的持久保存和跨越机器/平台的传输

pickle模块

- pickle是Python专用的序列化模块，不能跨平台
- pickle可实现对Python所有数据类型（对象）结构的二进制序列化和反序列化过程

pickle基本方法

方法	说明
pickle.dump (obj,file[,...])	将obj封存后的对象写入已打开的文件file
pickle.dumps(obj[, ...])	将obj封存后的对象作为字节类型直接返回，而不是将其写入文件
pickle.load(file[...])	从已打开的文件file中读取封存后的对象，重建其中特定对象的层次结构并返回
pickle.loads(data[...])	从data中读取封存后的对象，重建其中的对象层次结构并返回。data必须是类字节对象

pickle封存操作

```
import pickle

i = 2021
x = 3.14
s = "天地玄黄，宇宙洪荒"
lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
tu = (3, 5, 7, 11)
col = {-1, 0,1}
dict= {'a':'apple', 'b':'banana', 'g':'grape'}
data = [i, x, s, lst, tu, col, dict]

with open('pickle_sample.dat', 'wb') as fp:
    try:
        for item in data:
            pickle.dump(data, fp)
    except:
        print('写文件异常！')
```

Pickle解封操作

```
import pickle

with open('pickle_sample.dat','rb') as fp:
    while True:
        try:
            item = pickle.load(fp)
            print(item)
        except EOFError:
            break
```

```
2021
3.14
天地玄黄，宇宙洪荒
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
(3, 5, 7, 11)
{0, 1, -1}
{'a': 'apple', 'b': 'banana', 'g': 'grape'}
```

内容

- 文件的基本操作
- 二进制文件的序列化操作
- 文件的路径
- 文件级操作及目录操作
- 综合案例

绝对路径和相对路径

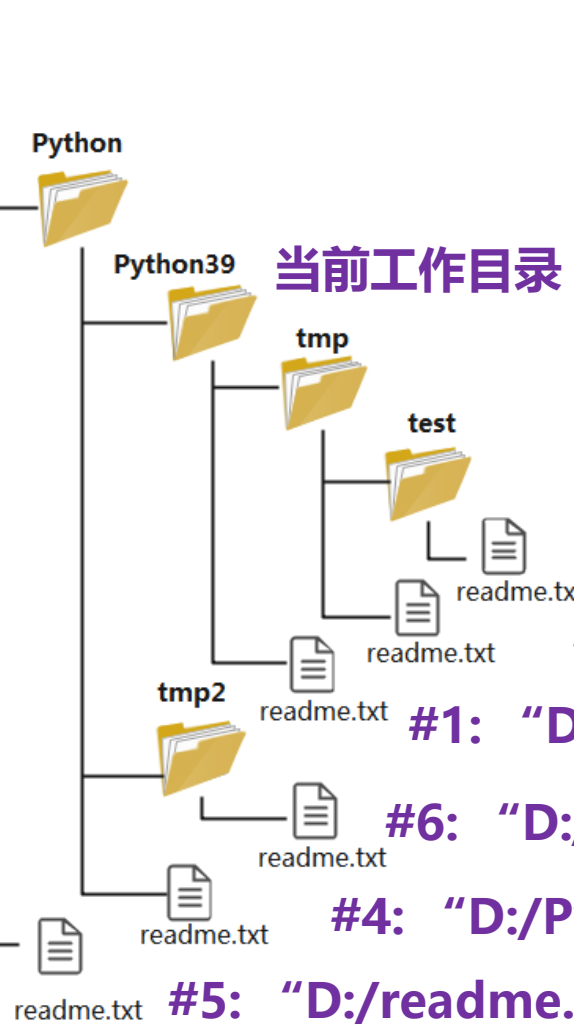
明确一个文件所在的路径，有两种表示方式

- 绝对路径：根文件夹开始一直到文件所在位置的路径。Window 系统中以盘符（C：、D：、E：）作为根文件夹，而 macOS或者 Linux 系统中以 / 作为根文件夹
- 相对路径：文件相对于当前目录所在的位置

从路径中是否包含根文件夹即可判断是绝对路径还是相对路径

open()中文件名的绝对路径和相对路径形式

```
open(“readme.txt”, “r”) #1
open(“tmp/readme.text”, “r”) #2
open(“tmp/test/readme.text”, “r”) #3
open(“../readme.txt”, “r”) #4
open(“../../readme.txt”, “r”) #5
open(“../tmp2/readme.txt”, “r”) #6
open(“D:/Python/readme.txt”, “r”)#4
```



Python程序的当前文件夹

- 一般情况下，.py文件所在的文件夹就是程序运行时的当前文件夹
- 当以命令行方式运行文件时，cmd窗口的当前文件夹就是程序运行时的当前文件夹，不管程序存储在哪里

可以通过os模块的getcwd方法获取当前文件夹

```
>>> import os
>>> os.getcwd()
'C:\\Users\\wangjing\\AppData\\Local\\Programs\\Python\\Python39'
>>> print(os.getcwd())
C:\Users\wangjing\AppData\Local\Programs\Python\Python39
```

Python程序的当前文件夹

可以通过os模块的chdir方法改变当前文件夹

```
>>> import os
>>> os.getcwd()
'C:\\Users\\wangjing\\AppData\\Local\\Programs\\Python\\Python39'
>>> print(os.getcwd())
C:\Users\wangjing\AppData\Local\Programs\Python\Python39

>>> os.chdir('c:/Users')
>>> os.getcwd()
'c:\\Users'
```

内容

- 文件的基本操作
- 二进制文件的序列化操作
- 文件的路径
- 文件级操作及目录操作
- 综合案例

Python os模块的文件及目录操作方法

方法	说明
os.chdir(path)	将程序的当前文件夹设为path
os.getcwd()	返回程序的当前文件夹
os.listdir(path)	返回一个列表，内容是文件夹path中所有子文件夹和文件的名字
os.mkdir(path)	创建文件夹path
os.path.exists(fp)	判断文件或文件夹fp是否存在
os.path.getsize(fp)	返回文件fp的大小（单位：字节）
os.path.isfile(fp)	判断fp是否是一个文件
os.remove(fp)	删除文件fp
os.rmdir(path)	删除文件夹path。path必须是空文件夹才能删除成功
os.rename(src, dst)	将文件或文件夹src改名为dst，还可用于移动文件或文件夹

Python shutil模块的常用方法

shutil模块是Python中高级的文件、文件夹和压缩包处理模块

方法	说明
copy(src, dst)	复制文件，新文件与原文件属性相同
copy2(src, dst)	复制文件，新文件与原文件属性完全相同，包括创建时间、修改时间和最后访问时间等
copyfile(src, dst)	复制文件，不复制文件属性，如果目标文件已存在则直接覆盖
copyfileobj(fsrc, fdst)	在两个文件对象之间复制数据
copymode(src, dst)	把src的模式位（mode bit）复制到dst上
copystat(src, dst)	把src的模式位、访问时间等所有状态都复制到dst上
copytree(src, dst)	递归复制文件夹
move(src, dst)	移动或递归移动文件夹，也可以给文件和文件夹重命名
rmtree(path)	递归删除文件夹

Python文件夹操作示例

```
import os
def getTotalSize(path):
    total = 0
    lst = os.listdir(path)
    for x in lst:
        actualFileName = path + "/" + x
        if os.path.isfile(actualFileName):
            total += os.path.getsize(actualFileName)
        else:
            total += getTotalSize(actualFileName)
    return total

path = "F:/Research"
totalSize = getTotalSize(path)
print("Total size of path",path,": ",totalSize)
```

```
Total size of path F:/Research :  1287759101
```

内容

- 文件的基本操作
- 二进制文件的序列化操作
- 文件的路径
- 文件级操作及目录操作
- 综合案例

综合案例：模拟高考赋分

自2020年起，北京实行“3+3”新高考制度

综合案例：模拟高考赋分

北京高考赋分制等级表

等	A					B					C					D					E
比例	15%					40%					30%					14%					1%
级	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C1	C2	C3	C4	C5	D1	D2	D3	D4	D5	E
比例	1%	2%	3%	4%	5%	7%	8%	9%	8%	8%	7%	6%	6%	6%	5%	4%	4%	3%	2%	1%	1%
分数	100	97	94	91	88	85	82	79	76	73	70	67	64	61	58	55	52	49	46	43	40

模拟高考赋分：题目要求

输入：Excel表格，存储准考证号及各科原始成绩

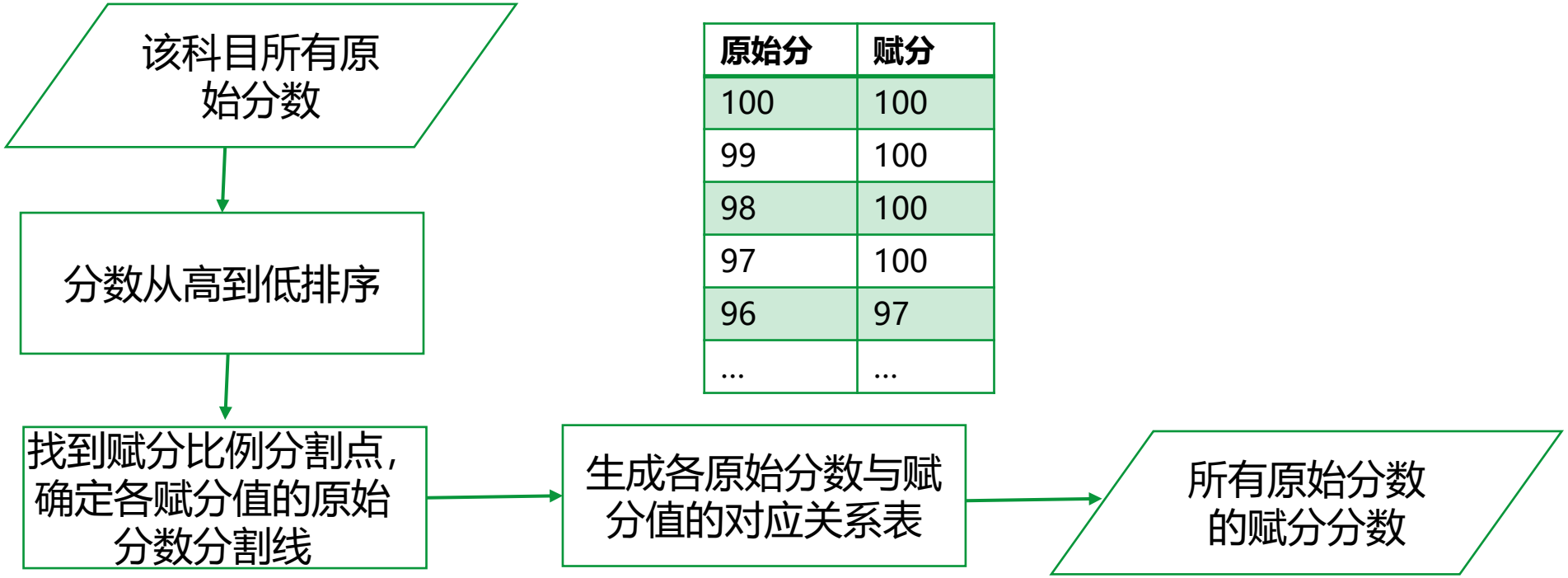
输出：赋分科目的赋分成绩，写入Excel表格

模拟高考赋分：选择合适的第三方库

- Python操作excel的常用第三方库包括：openpyxl、xlrd/xlwt、xlwings、xlxswriter等
- openpyxl几乎可以实现所有的 Excel 功能，而且接口清晰，简单易用，文档丰富，学习成本相对较低
- Openpyxl用workbook-sheet-cell的模式对.xlsx文件进行读写改操作，并可调整表格样式

模拟高考赋分：核心实现流程分析

本题目中，核心流程是对每一个赋分科目，实现从原始分到赋分的转换



原始分	赋分
100	100
99	100
98	100
97	100
96	97
…	…

比例	赋分值	原始分数分割线
1%	100	97
2%（1~3%）	97	95
3%（6~9%）	94	91
…	…	….

等级	占比	排名	分数
A1	1%	A1 ⩾ 1%	100
A2	2%	1% < A2 ⩽ 3%	
A3	3%	3% < A3 ⩽ 6%	
A4	4%	6% < A4 ⩽ 10%	
A5	5%	10% < A5 ⩽ 15%	88
B1	7%	15% < B1 ⩽ 22%	85
B2	8%	22% < B2 ⩽ 30%	82
B3	9%	30% < B3 ⩽ 39%	79
B4	8%	39% < B4 ⩽ 47%	76
B5	8%	47% < B5 ⩽ 55%	73
C1	7%	55% < C1 ⩽ 62%	70
C2	6%	62% < C2 ⩽ 68%	67
C3	6%	68% < C3 ⩽ 74%	64
C4	6%	74% < C4 ⩽ 80%	61
C5	5%	81% < C5 ⩽ 85%	58
D1	4%	86% < D1 ⩽ 89%	55
D2	4%	89% < D2 ⩽ 93%	52
D3	3%	93% < D3 ⩽ 96%	49
D4	2%	96% < D4 ⩽ 98%	46
D5	1%	98% < D5 ⩽ 99%	43
E	1%	99% < E ⩽ 100%	40

```
dict1 = {100: 1, 97: 3, 94: 6, 91: 10, 88: 15, 85: 22, 82: 30,
         79: 39, 76: 47, 73: 55, 70: 62, 67: 68, 64: 74, 61: 80,
         58: 85, 55: 89, 52: 93, 49: 96, 46: 98, 43: 99, 40: 100}
```

```
# 根据输入的成绩列表返回赋分分割线字典{赋分：分割分数}
def transDegree(scores):
    dict2 = {}
    scs = sorted(scores, reverse=True)
    l = len(scs)
    for k in dict1.keys():
        pos = round(l * dict1[k] / 100)
        if pos < l:
            dict2[k] = scs[pos]
        else:
            dict2[k] = 0
    return dict2
```

赋分	原始分数分割线
100	97
97	95
94	91
…	….

核心实现：生成赋分字典

赋分	原始分数分割线
100	97
97	95
94	91
…	….

原始分	赋分
100	100
99	100
98	100
97	100
96	97
…	…

```
# 根据输入的赋分分割线字典返回字典{原始分：赋分}
def transScore(degrees):
    dict4 = {}
    for i in range(100, -1, -1):
        dict4[i] = 0
        for k in degrees.keys():
            if i >= degrees[k] and dict4[i] < k:
                dict4[i] = k
    return (dict4)
```

测试数据准备：准考证号格式和考生数

准考证是参加高考的入场凭证。**准考证号**由9位数字**组成**，左边5位为考点代号，第6、7位为考场号，第8、9位为考场内座次号。

北京市公布高考考点考场安排，2020年北京设17个考区，132个考点学校，2867个考场。其中，备用考点共22个。

模拟准考证号格式如下：

AAABBCCDD

AAA：考区号，101~117

BB：考区内考点号，设各考区考点数：5~20以内随机数

CC：考场号，设各考点考场数：10~30以内随机数

DD：考场内座次号：01~20

生成准考证号

AAA：101~117，固定

BB：针对每个AAA，生成一个5~20之内的随机数，存入字典{ ‘AAA’ :BB}

AAA01~AAABB

CC：针对每个AAABB，生成一个10~30内的随机数，存入字典{ ‘AAABB’ ： CC}

DD：对于每个考点AAABB，CC最大的考场座位数为01~20之间的随机数，其它考场座位数为20个。生成字典{ ‘AAABBCC’ ： XX}，其中：XX=20或01~20之间的随机数

生成准考证号

```
#generateID.py：生成准考证号
import random
from openpyxl import load_workbook

fileName =r'C:\Users\wangj\PycharmProjects\SCoreProject\test2.xlsx'
sheetName='MySheet'
dictA, dictB, dictC = {}, {}, {}
admiNumberList = []
```

生成准考证号

```
for i in range(101, 118):
    k = str(i)
    dictA[k] = random.randrange(5, 21)
for k, v in dictA.items():
    for j in range(1, v + 1):
        if j >= 10:
            kb = k + str(j)
        else:
            kb = k + '0' + str(j)
        dictB[kb] = random.randrange(10, 31)
```

生成准考证号

```
for k, v in dictB.items():
    for j in range(1, v + 1):
        if j >= 10:
            kc = k + str(j)
        else:
            kc = k + '0' + str(j)
        if j < v:
            dictC[kc] = 20
        else:
            dictC[kc] = random.randrange(1, 21)
```

生成准考证号

```
for k, v in dictC.items():
    for j in range(1, v + 1):
        if j >= 10:
            admiNumber = k + str(j)
        else:
            admiNumber = k + '0' + str(j)
        admiNumberList.append(admiNumber)
admiNumberList.sort()
```

生成准考证号

```
wb = load_workbook(fileName)                    #打开excel文件
ws = wb[sheetName]                              #打开指定表单
ws.cell(row=1, column=1, value='准考证号')      #写单元格内容
for i in range(0, len(admiNumberList)):
    ws.cell(row=i + 2, column=1, value=admiNumberList[i])
wb.save(fileName)
                                                #保存文件内容
```

生成原始分数

```
#generateScore.py
from openpyxl import load_workbook
import numpy as np
fileName =r'C:\Users\wangj\PycharmProjects\SCoreProject\test2.xlsx'
sheetName ='MySheet'

j = 0
wb=load_workbook(fileName)
ws=wb[sheetName]
num_of_student = ws.max_row-1
```

生成原始分数

```
scoreList = np.random.randint(low=0, high=150,size=num_of_student*3) #前三科
for r in range(2,num_of_student+2):
    for c in range(2,11):
        ws.cell(r,c).value=''
for r in range(2,num_of_student+2):
    for c in range(2, 5):
        ws.cell(row=r, column=c, value=scoreList[j])
        j = j + 1
scoreList = np.random.randint(low=0, high=100, size=num_of_student * 3) #选科
j = 0
for r in range(2, num_of_student + 2):
    choose=np.random.choice(6,3,replace=False)
    for c in choose:
        ws.cell(row=r,column=c+5,value=scoreList[j])
        j = j+1
wb.save(fileName)
```

生成赋分

```
#generateMarks.py
import openpyxl

dict1 = {100: 1, 97: 3, 94: 6, 91: 10, 88: 15, 85: 22, 82: 30,
         79: 39, 76: 47, 73: 55, 70: 62, 67: 68, 64: 74, 61: 80,
         58: 85, 55: 89, 52: 93, 49: 96, 46: 98, 43: 99, 40: 100}

# 根据输入的成绩列表返回赋分分割线字典{赋分：分割分数}
def transDegree(scores):
    dict2 = {}
    scs = sorted(scores, reverse=True)
    l = len(scs)
    for k in dict1.keys():
        pos = round(l * dict1[k] / 100)
        if pos < l:
            dict2[k] = scs[pos]
        else:
            dict2[k] = 0
    return dict2
```

生成赋分

```
# 根据输入的赋分分割线字典返回字典{原始分：赋分}
def transScore(degrees):
    dict4 = {}
    for i in range(100, -1, -1):
        dict4[i] = 0
        for k in degrees.keys():
            if i >= degrees[k] and dict4[i] < k:
                dict4[i] = k
    return (dict4)
```

生成赋分

```
# 对指定表格fileName的sheetName表执行赋分，scoreCols是赋分列关系字典：{原始分数列号:赋分分数列号}
def scoreByColumn(fileName, sheetName, scoreCols):
    wb = openpyxl.load_workbook(fileName)
    ws = wb[sheetName]
    rows = ws.max_row
    coldata = []
    for oriScoreCol, finScoreCol in scoreCols.items():
        for i in range(2, rows + 1):
            cellvalue = ws.cell(row=i, column=oriScoreCol).value
            if cellvalue != None and cellvalue != -1:
                coldata.append(cellvalue)  # 获得有效分数列表：非空（未选），非-1(缺考)
        degreeDict = transDegree(coldata)  # 获得等级字典 {等级：分割分数}
        transDict = transScore(degreeDict)  # 获得赋分字典{原始分：赋分}

        subName = ws.cell(row=1, column=oriScoreCol).value
        ws.cell(row=1, column=finScoreCol, value=subName + '赋分')  # 为赋分列写列名
        for r in range(2, rows + 1):
            oriScore = ws.cell(row=r, column=oriScoreCol).value
            if isinstance(oriScore,int):
                ws.cell(row=r, column=finScoreCol, value=transDict[oriScore])
    wb.save(fileName)
```

对综合案例的思考：不足

- 明确的需求：可操作、无歧义的输入输出→程序的用途/用户
- 程序的容错性：必要的错误检查、异常处理，如对重复值、异常值、空值的检查与处理
- 完善的测试集：符合输入要求，覆盖程序分支，有必要的错误值
- 友好的用户交互：错误提示，必要的中间值存储及展示，阶梯值的处理及展示
- 输出结果的可检测性

文件：总结

- 文件操作在各类软件开发中都占有重要地位
- 文件操作通过文件对象的打开、读写及关闭进行
- 文件的读写方法都会自动改变文件对象中指针的位置
- 对于一般的二进制文件，无法直接读取和理解其内容，往往需要通过专门的第三方库对其进行操作
- 通过序列化和反序列化操作，可实现内存对象（包括其数据和结构）的持久化或跨平台传输