# 8.文件、异常和模块

## 8.1.文件的读写

### 8.1.1.文件的打开

实际应用中，我们绝大多数的数据都是通过文件的交互完成的

首先，来看文件打开的通用格式

**with** open("文件路径", "打开模式", encoding = "操作文件的字符编码") **as** f:

"对文件进行相应的读写操作"

*#使用with 块的好处：执行完毕后，自动对文件进行close操作。*

例如：一个简单的文件读取

**with** open("E:\ipython\测试文件.txt", "r", encoding = "gbk") **as** f: *# 第一步：打开文件*

text = f.read() *# 第二步：读取文件*

print(text)

我是一个测试文件

**文件路径**

完整路径，如上例所示

程序与文件在同一文件夹，可使用相对路径

**with** open("测试文件.txt", "r", encoding = "gbk") **as** f: *# 第一步：打开文件*

text = f.read() *# 第二步：读取文件*

print(text)

我是一个测试文件

**打开模式**

"r"  只读模式，如文件不存在，报错

"w" 覆盖写模式，如文件不存在，则创建；如文件存在，则完全覆盖原文件

"x" 创建写模式，如文件不存在，则创建；如文件存在，报错

"a"  追加写模式，如文件不存在，则创建；如文件存在，则在原文件后追加内容

"b" 二进制文件模式，不能单独使用，需要配合使用如"rb"，"wb"，"ab"，该模式不需指定encoding

"t" 文本文件模式，默认值，需配合使用 如"rt"，"wt"，"at"，一般省略，简写成如"r"，"w"，"a"

"+"，与"r","w","x","a"配合使用，在原功能基础上，增加读写功能

打开模式缺省，默认为只读模式

**字符编码**

万国码 utf-8

包含全世界所有国家需要用到的字符

中文编码 gbk

专门解决中文编码问题

windows系统下，如果缺省，则默认为gbk（所在区域的编码）

为清楚起见，除了处理二进制文件，建议不要缺省encoding

### 8.1.2.文件的读取

**1、读取整个内容**——f.read()

**with** open("三国演义片头曲\_utf.txt", "r", encoding="utf-8") **as** f: *# 第一步：打开文件*

text = f.read() *# 第二步：读取文件*

print(text)

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

**with** open("三国演义片头曲\_utf.txt", encoding="utf-8") **as** f: *# "r"，可缺省，为清晰起见，最好写上*

text = f.read()

print(text)

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

解码模式不匹配

**with** open("三国演义片头曲\_utf.txt", "r", encoding="gbk") **as** f:

text = f.read()

print(text)

**UnicodeDecodeError**: 'gbk' codec can't decode byte 0x80 in position 50: illegal multibyte sequence

**with** open("三国演义片头曲\_utf.txt", "r") **as** f: *# encoding缺省，windows系统默认为"gbk"*

text = f.read()

print(text)

**UnicodeDecodeError**: 'gbk' codec can't decode byte 0x80 in position 50: illegal multibyte sequence

**2、逐行进行读取——f.readline()**

**with** open("三国演义片头曲\_gbk.txt", "r", encoding="gbk") **as** f:

**for** i **in** range(2):

text = f.readline() *# 每次只读取一行*

print(text)

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

**with** open("三国演义片头曲\_gbk.txt", "r", encoding="gbk") **as** f:

**while** **True**:

text = f.readline()

**if** **not** text:

*# print(text is "")*

**break**

**else**:

*# print(text == "\n")*

print(text, end="") *# 保留原文的换行，使print()的换行不起作用*

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

**3、读入所有行，以每行为元素形成一个列表——f.readlines()**

**with** open("三国演义片头曲\_gbk.txt", "r", encoding="gbk") **as** f:

text = f.readlines() *# 注意每行末尾有换行符*

print(text)

['临江仙·滚滚长江东逝水\n', '滚滚长江东逝水，浪花淘尽英雄。\n']

**with** open("三国演义片头曲\_gbk.txt", "r", encoding="gbk") **as** f:

**for** text **in** f.readlines():

print(text) *# 不想换行则用print(text, end="")*

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

**4、文本文件读取小结**

文件比较大时，read()和readlines()占用内存过大，不建议使用

readline用起来又不太方便

可以直接对读取的对象进行迭代

**with** open("三国演义片头曲\_gbk.txt", "r", encoding="gbk") **as** f:

**for** text **in** f: *# f本身就是一个可迭代对象，每次迭代读取一行内容*

print(text)

临江仙·滚滚长江东逝水

滚滚长江东逝水，浪花淘尽英雄。

**5、二进制文件**

图片：二进制文件

**with** open("test.jpg", "rb") **as** f:

print(len(f.readlines()))

69

### 8.1.3.文件的写入

**1、向文件写入一个字符串或字节流（二进制）——f.write()**

**with** open("恋曲1980.txt", "w", encoding="utf-8") **as** f:

f.write("你曾经对我说**\n**") *# 文件不存在则立刻创建一个*

f.write("你永远爱着我**\n**") *# 如需换行，末尾加换行符\n*

f.write("爱情这东西我明白**\n**")

f.write("但永远是什么**\n**")

**with** open("恋曲1980.txt", "w", encoding="utf-8") **as** f:

f.write("姑娘你别哭泣**\n**") *# 如果文件存在，新写入内容会覆盖掉原内容，一定要注意！！！*

f.write("我俩还在一起**\n**")

f.write("今天的欢乐**\n**")

f.write("将是明天创痛的回忆**\n**")

**2、追加模式——"a"**

**with** open("恋曲1980.txt", "a", encoding="utf-8") **as** f:

f.write("姑娘你别哭泣**\n**") *# 如果文件存在，新写入内容会覆盖掉原内容，一定要注意！！！*

f.write("我俩还在一起**\n**")

f.write("今天的欢乐**\n**")

f.write("将是明天创痛的回忆**\n**")

**3、将一个元素为字符串的列表整体写入文件——f.writelines()**

ls = ["春天刮着风", "秋天下着雨", "春风秋雨多少海誓山盟随风远去"]

**with** open("恋曲1980.txt", "w", encoding="utf-8") **as** f:

f.writelines(ls)

ls = ["春天刮着风**\n**", "秋天下着雨**\n**", "春风秋雨多少海誓山盟随风远去**\n**"]

**with** open("恋曲1980.txt", "w", encoding="utf-8") **as** f:

f.writelines(ls)

### 8.1.4.既读又写

**1、"r+"**

如果文件名不存在，则报错

指针在开始的位置

要把指针移到末尾才能开始写，否则会覆盖前面内容

**with** open("浪淘沙\_北戴河.txt", "r+", encoding="gbk") **as** f:

*# for line in f:# print(line)*

*# 全部读一遍后，指针到达结尾*

f.seek(0,2) *# 或者可以将指针移到末尾f.seek(偏移字节数,位置（0：开始；1：当前位置；2：结尾）)*

text = ["萧瑟秋风今又是，**\n**", "换了人间。**\n**"]

f.writelines(text)

**2、"w+"**

若文件不存在，则创建

若文件存在，会立刻清空原内容！！！

**with** open("浪淘沙\_北戴河.txt", "w+", encoding="gbk") **as** f:

**pass**

**with** open("浪淘沙\_北戴河.txt", "w+", encoding="gbk") **as** f:

text = ["萧瑟秋风今又是，**\n**", "换了人间。**\n**"] *# 清空原内容*

f.writelines(text) *# 写入新内容，指针在最后*

f.seek(0,0) *# 指针移到开始*

print(f.read()) *# 读取内容*

**3、"a+"**

若文件不存在，则创建

若文件存在，指针在末尾，添加新内容，不会清空原内容

**with** open("浪淘沙\_北戴河.txt", "a+", encoding="gbk") **as** f:

f.seek(0,0) *# 指针移到开始*

print(f.read()) *# 读取内容*

大雨落幽燕，

白浪滔天。

秦皇岛外打鱼船。

**with** open("浪淘沙\_北戴河.txt", "a+", encoding="gbk") **as** f:

text = ["萧瑟秋风今又是，**\n**", "换了人间。**\n**"]

f.writelines(text) *# 指针在最后,追加新内容，*

f.seek(0,0) *# 指针移到开始*

print(f.read()) *# 读取内容*

大雨落幽燕，

白浪滔天。

秦皇岛外打鱼船。

萧瑟秋风今又是，

换了人间。

### 8.1.5.数据的存储与读取

通用的数据格式，可以在不同语言中加载和存储

本节简单了解两种数据存储结构csv和json

**1、csv格式**

由逗号将数据分开的字符序列，可以由excel打开

**读取**

**with** open("成绩.csv", "r", encoding="gbk") **as** f:

ls = []

**for** line **in** f: *# 逐行读取*

ls.append(line.strip("**\n**").split(",")) *# 去掉每行的换行符，然后用“,”进行分割*

**for** res **in** ls:

print(res)

['编号', '数学成绩', '语文成绩']

['1', '100', '98']

['2', '96', '99']

['3', '97', '95']

**写入**

ls = [['编号', '数学成绩', '语文成绩'], ['1', '100', '98'], ['2', '96', '99'], ['3', '97', '95']]

**with** open("score.csv", "w", encoding="gbk") **as** f: *# encoding="utf-8"中文出现乱码*

**for** row **in** ls: *# 逐行写入*

f.write(",".join(row)+"**\n**") *# 用逗号组合成字符串形式，末尾加换行符*

也可以借助csv模块完成上述操作

**2、json格式**

常被用来存储字典类型

**写入——dump()**

**import** **json**

scores = {"Petter":{"math":96 , "physics": 98},

"Paul":{"math":92 , "physics": 99},

"Mary":{"math":98 , "physics": 97}}

**with** open("score.json", "w", encoding="utf-8") **as** f: *# 写入整个对象*

*# indent 表示字符串换行+缩进 ensure\_ascii=False 显示中文*

json.dump(scores, f, indent=4, ensure\_ascii=**False**)

**读取——load()**

**with** open("score.json", "r", encoding="utf-8") **as** f:

scores = json.load(f) *# 加载整个对象*

**for** k,v **in** scores.items():

print(k,v)

Petter {'math': 96, 'physics': 98}

Paul {'math': 92, 'physics': 99}

Mary {'math': 98, 'physics': 97}

## 8.2.异常处理

### 8.2.1.常见的异常

**1、除0运算——ZeroDivisionError**

1/0

**ZeroDivisionError**: division by zero

**2、找不到可读文件——FileNotFoundError**

**with** open("nobody.csv") **as** f:

**pass**

**FileNotFoundError**: [Errno 2] No such file or directory: 'nobody.csv'

**3、值错误——ValueError**

传入一个调用者不期望的值，即使这个值的类型是正确的

s = "1.3"

n = int(s)

**ValueError**: invalid literal for int() with base 10: '1.3'

**4、索引错误——IndexError**

下标超出序列边界

ls = [1, 2, 3]

ls[5]

**IndexError**: list index out of range

**5、类型错误——TypeError**

传入对象类型与要求不符

1 + "3"

**TypeError**: unsupported operand type(s) for +: 'int' and 'str'

**6、其他常见的异常类型**

NameError 使用一个未被定义的变量  
 KeyError 试图访问字典里不存在的键

...

print(a)

**NameError**: name 'a' is not defined

d = {}

d["1"]

**KeyError**: '1'

当异常发生的时候，如果不预先设定处理方法，程序就会中断

### 8.2.2.异常的处理

提高程序的稳定性和可靠性

**1、try\_except**

如果try内代码块顺利执行，except不被触发

如果try内代码块发生错误，触发except,执行except内代码块

**单分支**

x = 10

y = 0

**try**:

z = x/y

**except** **ZeroDivisionError**: *# 一般来说会预判到出现什么错误*

*# z = x/(y+1e-7)*

*# print(z)*

print("0不可以被除！")

0不可以被除！

x = 10

y = 0

**try**:

z = x/y

**except** **NameError**: *# 一般来说会预判到出现什么错误*

*# z = x/(y+1e-7)*

*# print(z)*

print("0不可以被除！")

**ZeroDivisionError**: division by zero

**多分支**

ls = []

d = {"name": "大杰仔"}

**try**:

y = m

*# ls[3]*

*# d["age"]*

**except** **NameError**:

print("变量名不存在")

**except** **IndexError**:

print("索引超出界限")

**except** **KeyError**:

print("键不存在")

变量名不存在

**万能异常 Exception （所有异常的父类）**

ls = []

d = {"name": "大杰仔"}

**try**:

*# y = m*

ls[3]

*# d["age"]*

**except** **Exception**:

print("出错啦")

出错啦

**捕获异常的值 as**

ls = []

d = {"name": "大杰仔"}

*# y = x*

**try**:

y = m

*# ls[3]*

*# d["age"]*

**except** **Exception** **as** e: *# 虽不能获得错误具体类型，但可以获得错误的值*

print(e)

name 'm' is not defined

**2、try\_except\_else**

如果try 模块执行，则else模块也执行，如果进入except，则else不执行

可以将else 看做try成功的额外奖赏

**try**:

**with** open("浪淘沙\_北戴河.txt") **as** f:

text = f.read()

**except** **FileNotFoundError**:

print("找不到该文件，ta是不是用了美颜？")

**else**:

**for** s **in** ["**\n**", "，", "。", "？"]: *# 去掉换行符和标点符号*

text = text.replace(s, "")

print("毛主席的名作《浪淘沙\_北戴河》共由**{}**个字组成。".format(len(text)))

毛主席的名作《浪淘沙\_北戴河》共由65个字组成。

3、try\_except\_finally

不论是否有异常，finally最后都执行

ls = []

d = {"name": "大杰仔"}

*# y = x*

**try**:

y = m

*# ls[3]*

*# d["age"]*

**except** **Exception** **as** e: *# 虽不能获得错误具体类型，但可以获得错误的值*

print(e)

**finally**:

print("不论触不触发异常，都将执行")

name 'm' is not defined

不论触不触发异常，都将执行

## 8.3.模块简介

### 8.3.1.广义模块分类

已经被封装好

无需自己再“造轮子”

声明导入后，拿来即用

常用的模块分类：

1、Python 内置

时间库time\  随机库random\  容器数据类型collection\  迭代器函数itertools

2、第三方库

数据分析numpy、pandas\  数据可视化matplotlib\  机器学习scikit-learn\  深度学习Tensorflow

3、自定义文件

单独py文件

包——多个py文件，文件夹内多个py文件，再加一个\_\_init\_\_.py文件（内容可为空）

### 8.3.2.模块的导入

**1、导入整个模块——import 模块名**

调用方式：模块名.函数名或类名

**import** **time**

start = time.time() *# 调用time模块中的time()*

time.sleep(3) *# 调用time模块中的sleep() 休息3秒钟*

end = time.time()

print("程序运行用时：**{:.2f}**秒".format(end-start))

程序运行用时：3.00秒

**import** **numpy**

numpy.random.randint(1,10)

9

**2、从模块中导入类或函数——from 模块 import 类名或函数名**

调用方式：函数名或类名

**from** **numpy import** random

random.randint(10,20)

14

**from** **numpy.random import** randint *# 注意这种用法*

randint(10,20)

17

一次导入多个

**from** **numpy** **import** arange,sum

a = arange(10)

b = sum(a)

print(a)

print(b)

[0 1 2 3 4 5 6 7 8 9]

45

**3、导入模块中所有的类和函数——from 模块 import \***

调用方式：函数名或类名

**from** **random** **import** \*

print(randint(1,100)) *# 产生一个[1,100]之间的随机整数*

print(random()) *# 产生一个[0,1)之间的随机小数*

36

0.6582485822110181

### 8.3.3.模块的查找路径

模块搜索查找顺序：

**1、内存中已经加载的模块**

**import** **fun1**

fun1.f1()

导入fun1成功

*# 删除硬盘上的fun1 文件*

**import** **fun1**

fun1.f1()

导入fun1成功

*# 修改硬盘上的fun1 文件*

**import** **fun1**

fun1.f1()*# 居然没变，说明是优先从内存中读取的*

导入fun1成功

**2、内置模块**

*# Python 启动时，解释器会默认加载一些 modules 存放在sys.modules中*

*# sys.modules 变量包含一个由当前载入到解释器的模块组成的字典, 模块名作为键, 它们的位置作为值*

**import** **sys**

print(len(sys.modules))

print("math" **in** sys.modules)

print("numpy" **in** sys.modules)

**for** k,v **in** list(sys.modules.items())[:20]:

print(k, ":", v)

738

True

False

sys : <module 'sys' (built-in)>

...

**3、sys.path路径中包含的模块**

**import** **sys**

sys.path

['E:\\ipython',

'C:\\Users\\ibm\\Anaconda3\\python37.zip',

...

'C:\\Users\\ibm\\.ipython']

sys.path的第一个路径是当前执行文件所在的文件夹

若需将不在该文件夹内的模块导入，需要将模块的路径添加到sys.path

**import** **sys**

sys.path.append("C:**\\**Users**\\**ibm**\\**Desktop") *# 注意是双斜杠*

**import** **fun3**

fun3.f3()

导入fun3成功

## 8.4.作业练习

文件的读写：

1、读取文件“笑傲江湖.txt”，统计文本中所有字符的出现频次，将结果按如下格式保存到文件“笑傲江湖\_字符统计.txt”。

令：3904，狐： 3904，冲：4728

异常处理：

2、通过input输入数字的时候，我们通常使用eval(input(“请输入一个数字”))来实现。

考虑该代码可能出现的错误;

应该如何应对？

模块：

3、成功实现一个模块的导入及调用

构造一个筛选n以内所有素数的函数，将其保存在screening\_prime.py文件中。

新建一个py文件，导入screening\_prime模块中筛选素数的函数，并成功调用，获得1000以内的所有素数。

答案：

1.

**def** count\_word(s):

*'''统计字符串中字符的频次'''*

d = {}

**for** i **in** s:

d[i] = d.get(i, 0) + 1

**return** d

**with** open("笑傲江湖.txt", "r", encoding="utf8") **as** f:

s = f.read()

dic = count\_word(s)

**with** open("笑傲江湖\_字符统计.txt", "w", encoding="utf8") **as** f:

**for** k, v **in** dic.items():

f.writelines("**{}**：**{}\n**".format(k, v))

2.

**while** **True**:

**try**:

num = eval(input("请输入一个数字："))

**except** **Exception**:

print("你输入的不是一个数字！")

**else**:

print("你输入的数字是：**{}**".format(num))

**break**

你输入的不是一个数字！

你输入的不是一个数字！

你输入的数字是：123

3.

screening\_prime.py文件内容：

# -\*- coding: utf-8 -\*-

from math import sqrt

def is\_prime(num):

'''判断一个数是否为素数'''

if num == 1:

return False

for i in range(2, int(sqrt(num)) + 1):

if num % i == 0:

return False

return True

导入screening\_prime模块中筛选素数的函数，获得1000以内的所有素数：

**from** **screening\_prime** **import** is\_prime

**for** i **in** range(1, 1000):

**if** is\_prime(i):

print(i, end=" ")

2 3 5 7 ... 977 983 991 997