Day08 课堂笔记

课程笔记

复习和反馈

作业

今日内容

- 文件操作
  - 普通文件的操作
  - json 文件的操作[重点]
- 异常处理(程序代码运行时的报错)

文件介绍

计算机的 文件，就是存储在某种 长期储存设备 上的一段 数据
作用：将数据长期保存下来，在需要的时候使用

1.计算机只认识 二进制(0 1)
2.文件中存储的数据都是以二进制(0 1) 的形式去存储的

可以根据 文件中的二进制内容,能否使用记事本软件 将其转换为文字，将文件分为两种：文本文件和二进制文件

- 文本文件
  - 能够使用记事本软件打开(能够使用记事本转换为文字)
  - `txt， md, py , html, css, js , json`
- 二进制文件
  - 不能使用记事本软件打开的
  - `exe, mp3, mp4, jpg, png`

> 我们操作的基本都是文本文件

文件操作

文件操作的步骤

1. 打开文件
2. 读或者写文件
3. 关闭文件

1. 打开文件

打开文件：将文件从磁盘(硬盘) 中 读取到内存中
语法：

```
open(file, mode='r', encoding=None)
```

> 参数 file：是要打开的文件，类型是字符串，文件的路径可以是相对路径,也可以是绝对路径(从根目录开始书写的路径),建议使用相对路径(相对于当前代码文件所在的路径，./ ../)
>
> 参数 mode： 默认参数(缺省参数)，表示的是打开文件的方式
>
> > r：read 只读打开
> >
> > w：write 只写打开
> >
> > a> append 追加打开，在文件的末尾写入内容
>
> 参数 encoding：编码方式,(文字和二进制如何进行转换的)
>
> > gbk：将一个汉字转换为 2 个字节二进制
> >
> > utf-8：常用，将一个汉字转换为 3 个字节的二进制
>
> 返回值：返回的是 文件对象，后续对文件的操作,都需要这个对象

2. 读或者写文件

写文件

向文件中写入指定的内容.
前提: 文件的打开方式是 w 或者 a

```
文件对象.write('写入文件的内容')
# 返回值: 写入文件的字符数,一般不关注
# 注意 w 方式打开文件:
1. 文件不存在,会直接创建文件
2. 文件存在,会覆盖原文件(将原文件中的内容清空)
```

```
# 1, 打开文件
f = open('a.txt', 'w', encoding='utf-8')
# 2, 写文件
f.write('好好学习\n')
f.write('天天向上')
# 3, 关闭文件
f.close()
```

读文件

将文件中的内容读取出来
前提：文件的打开方式需要是 r
文件对象.read(n)
参数 n 表示读取多少个字符，一般不写,表示读取全部内容
返回值：读取到的文件内容，类型 字符串

```
# 1, 打开文件
f = open('a.txt', 'r', encoding='utf-8')
# 2, 读文件
buf = f.read()
print(buf)  # 目前只是打印读取的内容,可以做其它的事
# 3. 关闭文件
f.close()

# r 方式打开文件 ,如果文件不存在,代码会报错
```

3. 关闭文件

关闭文件：将文件占用的资源进行清理,同时会保存文件，文件关闭之后,这个文件对象就不能使用了
文件对象.close()

使用 with open 打开文件

with open() 打开文件的好处：不用自己去书写关闭文件的代码，会自动进行关闭

```
with open(file, mode, encoding='utf-8') as 变量:
    # 在缩进中去读取或者写入文件

# 缩进中的代码执行结束，出缩进之后，文件会自动关闭
```

```
with open('a.txt', 'a', encoding='utf-8') as f:
    f.write('good good study ')


# a 方式打开文件，文件不存在会创建文件，文件存在，在文件的末尾写入内容
```

按行读取文件内容

按行读取文件：一次读取一行内容

文件对象.readline()

```
# with open('b.txt', encoding='utf-8') as f:
#     buf = f.readline()  # 111
```

```
#     print(buf)
#     print(f.readline())  # 222


# with open('b.txt', encoding='utf-8') as f:
#     for i in f:  # 按行读取, 读到文件末尾结束
#         print(i, end='')


# read() 和 readline() 读到文件末尾, 返回一个空字符串, 即长度为 0

with open('b.txt', encoding='utf-8') as f:
    while True:
        buf = f.readline()
        if len(buf) == 0:
            break
        else:
            print(buf, end='')


# 在容器中 , 容器为空,即容器中的数据的个数为 0 ,表示 False, 其余情况都是 True
with open('b.txt', encoding='utf-8') as f:
    while True:
        buf = f.readline()
        if buf:  # if len(buf) != 0
```

```
            print(buf)
        else:
            break
```

json 文件的处理

json 文件 也是一个文本文件，就可以直接使用 read() 和 write() 方法 去操作文件，只是使用这两个方法,不方便,所以对 json 文件有自己独特的读取和写入的方法

常用在 在做测试的时候，将测试数据定义为 json 文件格式，使用 代码读取 json 文件,即读取测试数据，进行传参(参数化)

json 的介绍

json 基于文本，独立于语言的轻量级的数据交换格式
- 基于文本，是一个文本文件，不能包含图片,音视频等
- 独立于语言，不是某个语言特有的，每种编程语言都可以使用的
- 轻量级，相同的数据，和其他格式相比,占用的大小比较小
- 数据交换格式，后端程序员 给前端的数据 (json, html xml)

json 文件的语法

1. json 文件的后缀是 .json
2. json 中主要数据类型为 对象({} 类似 Python 中 字典) 和 数组([], 类似 Python 中的列表), 对象和 数组可以互相嵌套
3. 一个json 文件是一个 对象或者数组(即 json 文件的最外层要么是一个 {}, 要么是一个 数组 [])
4. json 中的对象是由键值对组成的, 每个数据之间使用 逗号隔开,但是最后一个数据后边不要写逗号
5. json 中的字符串 必须使用 双引号
6. json 中的其他数据类型

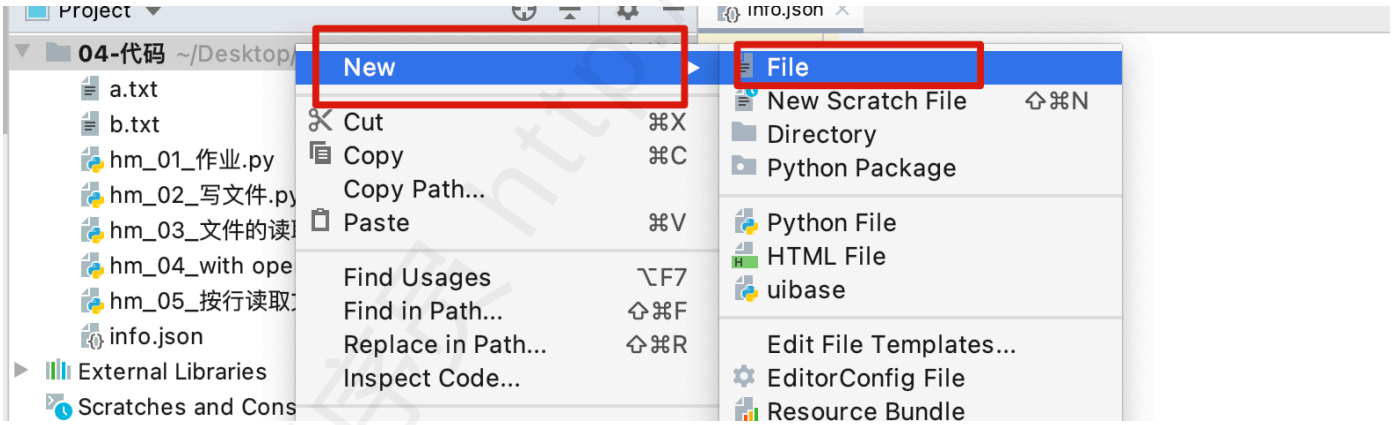
> 数字类型 ----> int float
> 字符串 string ---> str
> 布尔类型 true, false -----> True, False
> 空类型 null ----> None

json 文件的书写

我叫小明,我今年 18 岁,性别男, 爱好 听歌, 游戏,购物,吃饭,睡觉,打豆豆,
我的居住地址为 国家中国, 城市上海

```
{
  "name": "小明",
  "age": 18,
```

```
  "isMen": true,
  "like": [
    "听歌",
    "游戏",
    "购物",
    "吃饭",
    "睡觉",
    "打豆豆"
  ],
  "address": {
    "country": "中国",
    "city": "上海"
  }
}
```



读取 json 文件

```
1. 导包 import json
2. 读打开文件
3. 读文件
json.load(文件对象)

# 返回的是 字典(文件中是对象)或者列表(文件中是数组)
```

```python
# 1, 导入 json
import json

# 2, 读打开文件
with open('info.json', encoding='utf-8') as f:
    # 3. 读取文件
    # buf = f.read()
    # print(type(buf), buf)
    result = json.load(f)
    print(type(result))  # <class 'dict'>
    # 姓名
    print(result.get('name'))
    # 年龄
    print(result.get('age'))
    # 城市
    print(result.get('address').get('city'))
```

练习

我叫小明,我今年 18 岁,性别男, 爱好 听歌, 游戏,吃饭,睡觉,打豆豆,
我的居住地址为 国家中国, 城市上海.
我叫小红,我今年 17 岁,性别女, 爱好 听歌, 学习,购物
我的居住地址为 国家 中国, 城市北京.

- Info2.json

```
[
  {
    "name": "小明",
    "age": 18,
    "isMen": true,
    "like": [
      "听歌",
      "游戏",
      "购物",
      "吃饭",
      "睡觉",
      "打豆豆"
    ],
    "address": {
      "country": "中国",
      "city": "上海"
    }
```

```
  },
  {
    "name": "小红",
    "age": 17,
    "isMen": false,
    "like": [
      "听歌",
      "购物",
      "学习"
    ],
    "address": {
      "country": "中国",
      "city": "北京"
    }
  }
]
```

- 代码文件

```
import json

with open('info2.json', encoding='utf-8') as f:
    info_list = json.load(f)
    for info in info_list:
        print(info.get('name'), info.get('age'),
info.get('address').get('city'))
```

练习 2

某网站的测试数据如下 data.json，需求,提取 json 文件中的用户名,密码和预期结果，组成如下格式：[(), (), ()]（自动化参数化需要的数据格式）

```
[
   {
      "desc": "正确的用户名密码",
      "username": "admin",
      "password": "123456",
      "expect": "登录成功"
    },
    {
      "desc": "错误的用户名",
      "username": "root",
      "password": "123456",
      "expect": "登录失败"
    },
    {
      "desc": "错误的密码",
      "username": "admin",
      "password": "123123",
      "expect": "登录失败"
    }
```

```
]
```

```
import json


def read_data():
    new_list = []
    with open('info3.json', encoding='utf-8') as f:
        data = json.load(f)  # 列表
        # print(data)
        for i in data:  # i 字典
            # print((i.get('username'),
i.get('password'), i.get('expect')))
            new_list.append((i.get('username'),
i.get('password'), i.get('expect')))

        # print(new_list)
    return new_list
```

json 的写入

文件对象.write(字符串) 不能直接将 Python 的列表 和字典作为参数传递

想要将 Python 中的数据类型存为 json 文件，需要使用 json 提供的方法，不再使用 write

步骤：

1. 导包 import json
2. 写(w) 方式打开文件
3. 写入

json.dump(Python 中的数据类型，文件对象)

```python
import json

my_list = [('admin', '123456', '登录成功'), ('root', '123456', '登录失败'), ('admin', '123123', '登录失败')]

with open('info4.json', 'w', encoding='utf-8') as f:
    # json.dump(my_list, f)
    # json.dump(my_list, f, ensure_ascii=False)  # 直接显示中文,不以 ASCII 的方式显示
    # 显示缩进
    # json.dump(my_list, f, ensure_ascii=False, indent=2)
    json.dump(my_list, f, ensure_ascii=False, indent=4)
```

异常

程序在运行时，如果 Python 解释器 遇到到一个错误，会停止程序的执行，并且提示一些错误信息，这就是异常

程序停止执行并且提示错误信息 这个动作，抛出异常(raise 关键字)

捕获异常：程序遇到异常，默认动作是终止代码程序的执行，遇见异常之后，可以使用 异常捕获，让程序代码继续运行,不会终止运行(重点)

```
Traceback (most recent call last):
  File "/Users/nl/Desktop/20210620-23-python/day08_文件和异常/04-代码/hm_10_异常的介绍.py", line 2, in <module>
    num = int(num)
ValueError: invalid literal for int() with base 10: 'abcd'
 异常类型       异常的描述信息
```

异常捕获[重点]

基本语法

```
try:
    书写可能发生异常的代码
except:  # 任何类型的异常都能捕获
    发生了异常执行的代码


try:
    书写可能发生异常的代码
except 异常类型:  # 只能捕获指定类型的异常，如果不是这个异常,还是会报错
    发生了异常执行的代码
```

```
try:
    # 1. 获取用户从键盘输入的数据
    num = input('请输入数字:')
    # 2. 转换数据类型为整数
    num = int(num)
    # 3. 输出转换之后的数据内容
    print(num)
except:
    print('请输入正确的数字')

print('后续其他的代码,可以继续执行')

try:
    # 1. 获取用户从键盘输入的数据
    num = input('请输入数字:')
```

```
    # 2. 转换数据类型为整数
    num = int(num)
    # 3. 输出转换之后的数据内容
    print(num)

except ValueError:  # 只能捕获 ValueError 类型及其子类的异常
    print('发生了异常，请输入正确的数字...')
```

捕获多个指定类型的异常

好处：可以针对不同的异常错误,进行单独的代码处理

```
try:
    书写可能发生异常的代码
except 异常类型1:  # 只能捕获指定类型的异常，如果不是这个异常,还是会报错
    发生了异常1执行的代码
except 异常类型2:
    发生了异常2执行的代码
except 异常类型...:
    发生了异常...执行的代码
```

```
try:
    # 1. 获取用户从键盘输入的数据
```

```
    num = input('请输入数字:')
    # 2. 转换数据类型为整数
    num = int(num)
    # 3. 输出转换之后的数据内容
    print(num)
    a = 10 / num    # 10 / 0
    print(f'a: {a}')

except ValueError:  # 只能捕获 ValueError 类型及其子类的异常
    print('发生了异常, 请输入正确的数字...')
except ZeroDivisionError:
    print('除数不能为 0')
```

异常捕获的完整版本

完整版本中的内容,不是说每一次都要全部书写,根据自己的需要,去选择其中的进行使用

```
try:
    可能发生异常的代码
except 异常类型1:
    发生异常类型1执行的代码
#  Exception 是常见异常类的父类，这里书写 Exception,可以捕获常见的所有一会，as 变量，这个变量是一个异常类的对象，print(变量) 可以打印异常信息
except Exception as 变量:
    发生其他类型的异常,执行的代码
else:
    没有发生异常会执行的代码
finally:
    不管有没有发生异常,都会执行的代码
```

```
try:
    可能发生异常的代码
except Exception as e:
    发生异常执行的代码
```

```
try:
    # 1. 获取用户从键盘输入的数据
    num = input('请输入数字:')
    # 2. 转换数据类型为整数
    num = int(num)
    # 3. 输出转换之后的数据内容
    print(num)
```

```
    a = 10 / num    # 10 / 0
    print(f'a: {a}')

except Exception as e:
    print(f"错误信息为: {e}")
else:
    print('没有发生异常我会执行')
finally:
    print('不管有没有发生异常,我都会执行')
# print('不管有没有发生异常,我都会执行')
```