day 08 课堂笔记

课程之前

复习和反馈

```
对象 = 类名()  # __init__
对象.方法()
```

作业

```
sum() python 中函数，可以对列表/元组中的数字求和
```

```
num = 1

list1 = [1, 2, 3, 4, 5]

print(sum(list1))
```

今日内容

```
文件操作
-    按行读取文件 readline()
json 操作(特殊的文本文件)
-    如何定义,语法
-    读取(重点)
-    写入
异常(程序代码中的报错)
-    基本介绍
-    捕获异常(重点)
-    异常的传递，抛出异常
```

文件操作

按行读取文件 readline()

```
文件对象.readline()  # 一次读取一行的内容，返回读取到的内容

read() 和 readline() 如果读到文件末尾，返回的都是 空字符串
```

```
with open('a.txt', encoding='utf-8') as f:
    buf = f.readline()
    print(buf)  # aaaaaa
    buf1 = f.readline()
    print(buf1)  # bbbbbb
```

读取大文件

```
# with open('a.txt', encoding='utf-8') as f:
#     while True:
#         buf = f.readline()  # 文件读完了,返回 空字符串
#         if buf == "":
#             break
#         else:
#             print(buf, end='')


with open('a.txt', encoding='utf-8') as f:
    while True:
        buf = f.readline()  # 文件读完了,返回 空字符串
        if buf:   # 空字符串 是 False, 非空字符串 是 True
            print(buf, end='')
```

```
        else:
            break
```

打开文件的方式

r w a 称为是文本方式打开，适用于 文本文件，会对二进制进行编码转换
rb wb ab 称为是二进制方式打开，可以打开文本文件和二进制文件，但是 二进制文件只能使用 二进制方式打开,同时,不能传递 encoding 参数

json 文件

1，json 文件的本质也是文本文件，就可以直接使用 read 和 write 去进行操作
2，json 文件比较特殊，比较像 Python 中的字典和列表
3，json 文件使用比较频繁,按照 read 和 write 的操作,比较麻烦,专门的方法来操作 json 文件，可以直接得到 Python 中的列表和字典

介绍

1，json 文件，是一种基于文本，独立于语言的轻量级数据交换格式。

- 基于文本，文本文件,不包含 图片,视频等
- 独立于语言，不是某一种语言特有的，Python,Java，C++,....
- 轻量级，相同的数据量,json 文件的占用的文件大小相对较小
- 数据交换格式，后端服务器 和前端页面 交换数据使用的格式

2，在自动化测试中经常用来存放测试数据，文件后缀名为：.json

json 的语法

1, json 中的数据类型
- 对象 {} ----> Python 字典
- 数组 [] ----> Python 列表
- 字符串, 必须使用双引号 -----> str
- 数字类型 ---> int, float
- bool类型(true false) ----> True False
- 空值 null -----> None

2, json 文件,是一个对象 或者是数组, 对象和数组可以相互嵌套

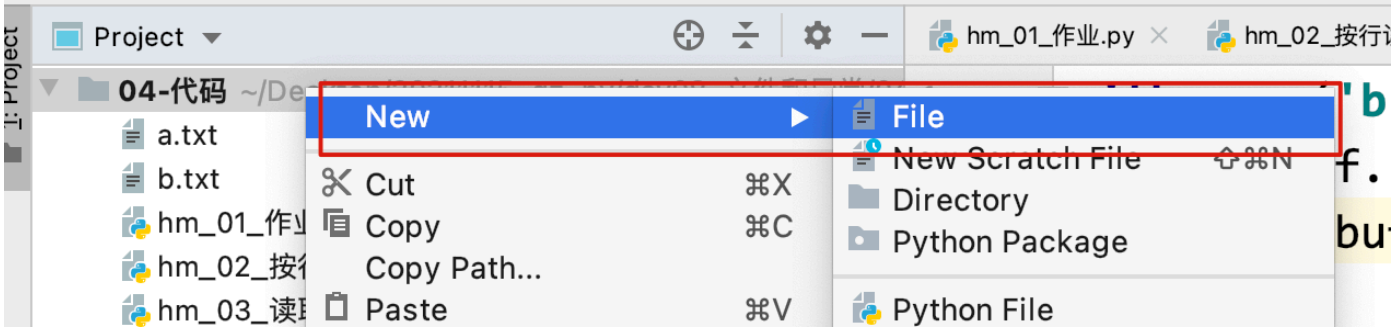
3, json 中的对象,是由键值对组成的, 键必须是 字符串类型

4, json 中的数据直接使用逗号隔开, 最后一个数据后边不能加逗号

5, json 文件的后缀是 .json

json 文件的定义

我叫小明,我今年 18 岁,性别男, 学校 空, 爱好 听歌,吃饭,打豆豆, 我的居住地址为 国家 中国, 城市 广州.



```
{
  "name": "小明",
  "age": 18,
  "isMan": true,
  "school": null,
  "like": ["听歌", "吃饭", "打豆豆"],
  "address": {
    "country": "China",
    "city": "广州"
  }
}
```

读取 json 文件

```
1, 可以直接使用 read 去读, 但是 想要取到数据很麻烦
2, 使用 专门的方法去读
-   1. 导包 import json
-   2. json.load(文件对象)  ---> 得到的是 列表 或者字典
```

```python
import json


with open('info.json', encoding='utf-8') as f:
    buf = json.load(f)
    print(type(buf))
    print(buf)
    # 姓名
    print(buf.get('name'))
    # 城市
    print(buf.get('address').get('city'))
```

练习

```
我叫小明,我今年 18 岁,性别男, 爱好 听歌, 吃饭,打豆豆,
我的居住地址为 国家中国, 城市广州.   ---> 对象

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.   ---> 对象
-----
获取 每个人的姓名, 年龄 性别, 城市
```

- Json 文件

```
[
```

```
{
  "name": "小明",
  "age": 18,
  "isMan": true,
  "like": [
    "听歌",
    "吃饭",
    "打豆豆"
  ],
  "address": {
    "country": "China",
    "city": "广州"
  }
},
{
  "name": "小红",
  "age": 17,
  "isMan": false,
  "like": [
    "听歌",
    "学习",
    "购物"
  ],
  "address": {
    "country": "China",
    "city": "北京"
  }
```

```
  }
]
```

- 代码

```
import json

with open('info2.json', encoding='utf-8') as f:
    data_list = json.load(f)  # 列表
    for data in data_list:
        # sex = "男" if data.get('isMan') else
"女"  # 扁平化
        if data.get('isMan'):
            sex = "男"
        else:
            sex = '女'
        print(f"姓名: {data.get('name')}, 年龄:
{data.get('age')}"
              f"性别: {sex}, 城市:
{data.get('address').get('city')}")

    # 条件为True 执行的代码 if 判断条件 else 条件为
False 执行的代码
    # a = 'a ' if 3 > 1 else 'b'
```

写入(了解)

```
# 将 Python 中列表或者字典 转换为 json 文件

# 导包 import json
# 使用 json.dump(Python中数据, 文件对象)
```

```
import json

info = {'name': '小明', 'age': 18}

with open('info3.json', 'w', encoding='utf-8') as f:
    # json.dump(info, f)  #
    # json.dump(info, f, ensure_ascii=False)  # 直接显示中文
    json.dump(info, f, ensure_ascii=False, indent=2)
 # 直接显示中文
```

异常

介绍

```
1, 程序在运行时, 如果Python解释器遇到到一个错误, 则会停止程序的执行, 并且提示一些错误信息, 这就是异常.
2, 程序停止执行并且提示错误信息这个动作, 通常称之为: 抛出(raise) 异常
```

```
异常类型: 错误描述信息
FileNotFoundError: [Errno 2] No such file or
directory: 'aaaa.txt'
ZeroDivisionError: division by zero
ValueError: invalid literal for int() with base 10:
'78.1'
```

捕获异常

```
1，程序代码在执行的时候，如果遇到异常，程序就会终止,不会
继续执行
2，需求：程序遇到异常之后，不会结束,可以继续执行，
3，实现需求：就需要使用 异常捕获
```

基本语法

```
try:
    可能发生异常的代码
except:  # 可以捕获任意类型的异常
    发生了异常执行的代码
```

```python
# 1. 获取用户从键盘输入的数据
num = input('请输入数字:')
try:
    # 2. 转换数据类型为整数
    num = int(num)
    # 3. 数据转换类型正确时，输出数据内容
    print(num)
except:
    # 4. 数据转换类型错误时，提示输入正确数据
    print('请输入正确的数字')
```

捕获指定类型的异常

发生的异常可能存在多种，针对不同类型的异常，解决处理的方案不一样

```python
try:
    可能发生异常的代码
except 异常类型1:
    发了异常类型1,执行的代码
except 异常类型2:
    发了异常类型2,执行的代码
except .....:
    pass
```

```
try:
    num = int(input('请输入一个整数数字:'))
    num1 = 8 / num
    print(num1)
except ValueError:  #
    print('输入的内容非数字,请重新输入')
except ZeroDivisionError:
    print('不能输出数字 0, 请重新输入')
```

捕获未知类型的异常(使用最多)

```
try:
    可能发生异常的代码
except Exception as 变量:  # Exception 常见异常类的父类, 变量 异常对象,print()可以打印异常信息
    发生异常执行的代码
```

```
try:
    num = int(input('请输入一个整数数字:'))
    num1 = 8 / num
    print(num1)
except Exception as e:
    print(f'发生了异常, {e}')
```

异常捕获的完整结构

```
try:
    可能发生异常的代码
except 异常类型:
    发生了指定类型的异常执行的代码
except Exception as e:
    发生了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
```

案例

需求:
1. 获取用户输入的数字
2. 判断获取的数字是否整数
3. 如果不是整数, 提示输入错误
4. 如果是整数, 则进一步判断是奇数还是偶数
5. 最终提示: 程序运行结束

方案一
使用异常捕获
方案二
if 判断
字符串.isdigit() 判断数字是否是纯数字,是 纯数字,返回 True,不是返回 False

```
# 1. 获取用户输入的数字
num = input('请输入数字:')
try:
    # 2. 判断获取的数字是否整数
    num = int(num)   # 没有发生异常, 说明是整数,如果发生异常,说明不是整数
except Exception as e:
    # 3. 如果不是整数, 提示输入错误
    print("输入错误", e)
else:
    # 4. 如果是整数, 则进一步判断是奇数还是偶数
    if num % 2 == 0:
        print('偶数')
```

```
    else:
        print('奇数')
# 5. 最终提示：程序运行结束
finally:
    print('程序运行结束')
```

```
# 1. 获取用户输入的数字
num = input('请输入数字:')
# 2. 判断获取的数字是否整数
if num.isdigit():
    # 如果是 True，表示是整数
    # 类型转换
    num = int(num)
    # 4. 如果是整数，则进一步判断是奇数还是偶数
    if num % 2 == 0:
        print('偶数')
    else:
        print('奇数')
# 3. 如果不是整数，提示输入错误
else:
    print('输入错误')

# 5. 最终提示：程序运行结束
print('程序运行结束')
```

异常传递

1，异常传递 是 Python 中已经实现好的功能，不需要我们写代码实现
2，异常传递是指，在函数的嵌套调用过程中,如果发生了异常，没有进行捕获,会将这个异常传递到函数调用的地方，直到被捕获为止，如果一直没有捕获,才会报错,终止执行

抛出异常

1，在执行代码的过程中,之所以会发生异常,终止代码执行,是因为 代码执行 遇到了 raise 关键字
2，raise 关键字的作用，就是来抛出异常,让代码终止执行
3，应用场景：自己书写代码模块,让别人使用,为了让别人按照你的规定使用你的代码,你就可以在他不满足你条件的情况下,使用 raise 抛出异常