Python 第3天

作业评讲

1. 编写函数sum(n)，该函数可以用来计算1~n的和。

2. 编写一个student类，包含姓名、性别、年龄、家庭住址，并在info()方法中显示这些信息。根据类生成两个对象s1、s2，并分别调用对象的info()方法输出学员的信息。

3. 编程实现输入某年某月某日，输出它是这一年的第几天。

一、面向对象

- 通用模板

```
class 类名:
    # 类属性/类变量，整个类共享的，不管创建多少个对象，内存中只有一个类属性
    # 在类的内部，既可以用self来访问，也可以用类名来访问，但推荐用类名来访问
    # 在类的外部，可以直接使用类名来访问，也可以用对象名来访问，但推荐使用类名来访问
    类属性名1 = 值1
    类属性名2 = 值2

    # 构造方法会在未来，创建类的实例对象时，自动调用
    def __init__(self, 参数1, 参数2):
        # self.实例属性，在类的内部只能用self来访问，在类的外部只能用对象来访问
        self.实例属性名1 = 参数1
        self.实例属性名2 = 参数2
        self.实例属性名3 = 值3

    def 普通方法1(self, 可选的参数):
        方法的主体语句
        可以使用self来访问自己的实例属性和类属性
        也可以使用类名来访问类属性
        也可以使用self来调用其他的普通方法
        可以有return

    def 普通方法2(self, 可选的参数):
        方法的主体语句
        可以使用self来访问自己的实例属性和类属性
        也可以使用类名来访问类属性
        也可以使用self来调用其他的普通方法
        可以有return
```

- **继承**
  - 本质：为了提供代码的重用性，降低代码的冗余度。将共同的代码提取出来，定义在一个类中，让其他的类来公用这段代码。那么就实现了继承，提取出来的代码所在的类，被称为**父类**（也称为超类、基类），扩展了父类的类被称为**子类**（扩展类）
  - 模板：

```python
class A:
    pass

class B(A):
    pass

class C(B):
    pass

# B类继承了A类
# C类继承了B类
# B类继承A类的所有成员（构造方法、普通方法、属性）
# C类继承了A类的所有成员，以及B类的所有成员
```

    - 注意：
        - 一旦子类继承了父类，会继承父类的所有成员（构造方法、普通方法、属性）
        - 子类又经常会扩展一些属性，就会重写父类的构造方法（`__init__`），一旦重写，就需要将所有父类的属性，自己再定义一遍。
        - 子类的方法中，使用self来访问父类的属性或本类的属性
        - 子类的方法中，可以使用self来访问本类或父类中的方法。也可以明确的使用`super()`来访问父类的方法。
        - 在重写的子类方法中，只能用super()调用父类中被重写的方法，如果此时用self，调用的是子类方法自身。
        - 继承是有传递性的，如果B继承A，C又继承B，那么等于C也继承了A
        - 所有的类，如果没有显示的继承，都默认继承系统自带的object类
            - object类（对象），是所有类的根，所有的类都是对象类

- **使用`isinstaceof(数据，类)`来判断指定的数据是否是某个类型**
    - 子类类型的数据肯定也算是父类类型的
- **多态**
    - 理解：多个子类都重写了父类的某个方法，创建了子类的对象以后，调用该方法（执行的是子类的方法），所以表现出不同的形态，叫做多态。
    - 优点：对调用者隐藏子类实现的细节，调用者不关心具体的子类类型，只需要知道父类的方法，即可。

二、读写文本文件

- 目的：编写自动化脚本的时候，一般可以将测试数据集中存放，将测试数据和测试逻辑的用例代码分开。那么测试用例的脚本在运行的时候，必然要从文件中，读取测试数据。
- 操作文件的三部曲
    1. 打开文件
    2. 操作文件（从文件中读取内容、向文件中写入内容）
    3. 关闭文件（释放资源）
- 文件的打开文本
    - 文本文件：r（读取字符串）、w（覆盖写入字符串）、a（追加写入字符串）
        - 字符编码，防止乱码，建议文本文件都用utf-8编码

    - 二进制文件：rb（读取字节）、wb（覆盖写入字节）、ab（追加写入字节）
- 读取文件的操作
    - 示例代码：

```
with open('文件的路径', 'r', encoding="utf-8") as file:
    # 一次读取所有的数据，以str类型的数据返回
    file.read()
    # 一次读取所有的数据，以list类型的数据返回，一行一个元素
    file.readlines()
    # 一次读取一行数据，以str类型的数据返回，读取到文件的末尾，再次调用则返回空字符串''
    file.readline()
```

- 向文件中写入数据的操作：
    - 示例代码：

```
with open('文件的路径', 'w或a', encoding="utf-8") as file:
    # 不会自动加上换行符，如果需要换行，必须手动指定换行符
    file.write("写入的内容\n")
```