華南師範大學
SOUTH CHINA NORMAL UNIVERSITY

Python语言程序设计

类与对象

本章节目标

理解类的定义和创建实例的方法

掌握构造方法和析构方法

掌握类中属性和方法的定义

课程目录
Course catalogue

面向对象

程序设计方法分为面向过程程序设计和面向对象程序设计。

面向对象

面向过程程序设计就是以过程为核心，强调事件的流程、顺序，分析出解决问题所需要的步骤，然后用函数把这些步骤实现，使用的时候依次调用函数。

面向对象

面向对象程序设计是以对象为核心，强调事件的角色、主体，将软件结构建立在对象上，而不是功能上，通过对象来模拟现实世界中的事务，使计算机求解问题更加类似于人类的思维活动。

面向对象

举个栗子：将大象装进冰箱，面向过程做法。

面向过程，就是分析问题的步骤，按照步骤解决问题。

面向对象

将大象装进冰箱，面向对象做法

走进去

打开冰箱门

关上冰箱门

- 定义一个大象类(型)
- 定义一个冰箱类(型)
- 创建一个大象类的实例对象
- 创建一个冰箱类的实例对象
- 冰箱对象->开门
- 大象对象->走进去
- 冰箱对象->关门

面向对象

举个栗子：用户输入半径，求面积

```
r=float(input('请输入半径'))
area=3.14*r*r
print(area)
```

面向过程，就是分析问题的步骤，按照步骤解决问题。

面向对象

举个栗子：用户输入半径，求面积

写一个圆类（类型）
- 根据用户输入的半径，创建一个圆；
- 写若干方法
- 根据半径，求圆的周长；
- 根据半径，求圆的面积；
- 根据半径，求圆的直径；
- 修改圆的半径
- 输出圆的信息

面向对象，先写类型，创建对象，解决问题。

面向对象

举个栗子：用户输入半径，求面积

```
class Circle:
    def __init__(self,r):
        self.radius=r
    def get_area(self):
        return 3.14*self.radius**2
    def print_info(self):
        print("半径:"+str(self.radius)+"面积:"+str(self.get_area()))

c1=Circle(eval(input("请输入半径")))
c1.print_info()
```

面向对象，先写类型，创建对象，解决问题。

面向对象

面向对象程序设计以对象为程序的基本单元，提高了软件的重用性、灵活性和扩展性，提高了软件开发的效率，所以面向对象程序设计是目前软件开发领域的主流技术。

面向对象

面向对象程序设计具有三大基本特征：封装性、继承性和多态性

封装是面向对象编程的核心思想，将对象的属性和行为封装起来，封装起来的载体是类，类通常对用户隐藏其实现的细节。

采用封装的思想保证了类内部数据结构的完整性，应用该类的用户不能轻易直接操纵该类的数据，而只能执行该类允许公开的数据。这样可以避免外部对内部数据的影响，提高程序的可维护性。

面向对象

继承是在现有类的基础上通过添加属性或方法对现有类进行扩展，派生出新类的现象称为类的继承机制，也称为继承性。

继承的过程，就是从一般到特殊的过程。子类无须重新定义在父类中已经声明的属性和行为，可拥有其父类的属性和行为。

子类既具有继承下来的属性和行为，又具有自己新定义的属性和行为。在软件开发中，类的继承性使软件具有开放性、可扩充性，实现了代码重用，有效地缩短了程序的开发周期。

面向对象

多态性，是指在父类中定义的属性和方法被子类继承之后，可以具有不同的数据类型或表现出不同的行为，这使得同一个属性或方法在父类及其各个子类中具有不同的含义。多态性增强了软件的灵活性和重用性。

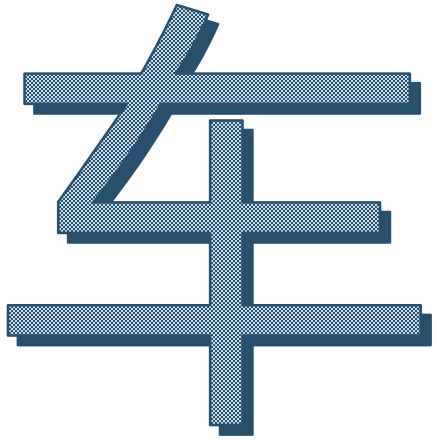
类

类是对现实生活中一类具有共同特征的事物的抽象

- 具有相同特性（数据元素）和行为（功能）的对象的抽象就是类。
- 类的具体化就是对象，也可以说类的实例是对象。
- 类具有属性，它是对象的状态的抽象，用数据结构来描述类的属性。
- 类具有操作，它是对象的行为的抽象，用实现该操作的方法来描述。

类与对象

属性（特征）
品牌
颜色
重量
车型
能源类型



功能（方法）
驾驶（跑）
能源补充
自动驾驶
听音乐
打开天窗

类与对象

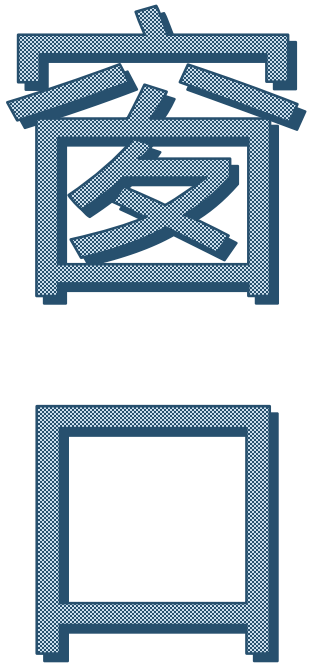
属性	说明
品牌	大众
颜色	黑色
重量	1.5吨
车型	轿车
能源类型	电动

广场上的一辆车

功能	说明
驾驶	公里/小时
能源补充	充电
功能1	自动驾驶
功能2	听音乐
功能3	打开天窗

类与对象

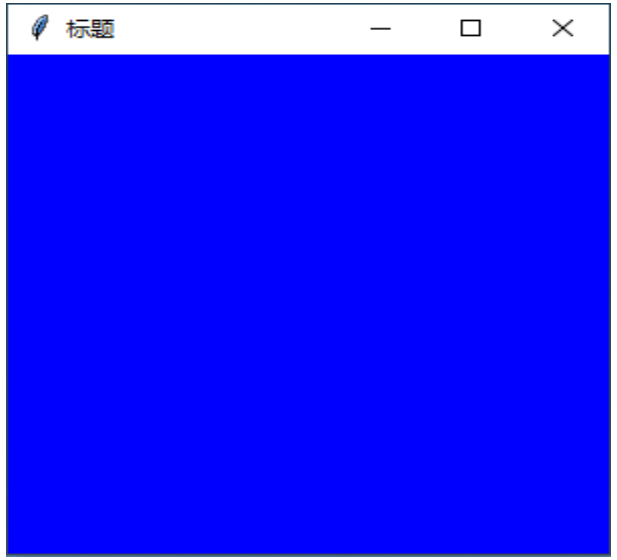
属性（特征）
宽高
背景色
字体大小
边框大小
标题
图标



操作（功能）
打开
关闭
最大化
最小化
改变大小
拖动

类与对象

属性	说明
宽高	300x300
背景色	蓝色
字体大小	20
边框大小	1



操作（功能）
可打开
可关闭
可最大化
可最小化
可改变大小
可拖动

类

类是一种用户定义的数据类型，也称类类型。

每个类包含数据说明(属性）和一组操作数据的函数（方法）。

类的定义


```
class ClassName:
    '''类的说明'''
    #属性
        [属性定义体]
    #方法
        [方法定义体]
```

类的定义以关键字class开始，后面是类名，类名通常开头字母大写，类名后面紧跟冒号“:”

类的定义

下面是一个最简单的类的定义方法，即定义一个空类

```
class Circle:
    pass
```

定义一个名字为Circle的类
pass 一个空语句，起到占位作用，
表示Circle类中没有任何属性和方法

课程目录
Course catalogue

创建类的实例对象

类定义好之后，就可将类实例化为对象。类实例化对象的语法格式如下：

```
实例对象 = 类名（ ）
```

实例化对象的操作符是：等号“=”，在类实例化对象的时候，类名后面要添加一个括号“()”

创建类的实例对象


```
class Circle :
pass
```

```
c = Circle()
```

上例将类Circle实例化为对象c,也可以说创建了一个属于Circle 类型的实例对象c

创建类的实例对象


```
class Circle : #定义类
    pass

c= Circle() #创建实例对象
print(c)
```

程序执行结果：<__main__.Circle object at 0x00000216EE7DF0F0>

从输出结果中可以看到，c是Circle类的实例对象

类与对象

类和对象的关系可以看做数据类型与变量的关系

根据一个类可以创建多个对象，而对象只能是某一个类的对象。

```
class Circle:
    pass

c1= Circle()
c2= Circle()
print(c1)
print(c2)
```

```
<__main__.Circle object at 0x0000017BC3D693D0>
<__main__.Circle object at 0x0000017BC3D69C40>
```

课程目录
Course catalogue

构造方法

构造方法是Python类中的内置方法，它的方法名为__init__，在创建一个类的实例对象时会自动调用，负责完成创建实例对象的初始化工作。

```
class Circle : #定义Circle类
    def __init__(self): #定义构造方法,self对应当前实例对象
        print('构造方法被调用！')
```

__init__()方法必须包含一个self参数（也可用其它名子），并且必须是第一个参数。self参数是一个指向实例对象的引用，用于访问属性和方法，在方法调用时会传递实例对象给self。

构造方法


```
class Circle : #定义Circle类
    def __init__(self): #定义构造方法,self对应当前实例对象
        print('构造方法被调用！')


c= Circle() #创建Circle类的实例对象c，自动调用构造方法，
            #实例对象c传给构造方法的self形参
```

输出：构造方法被调用！

如果类中没有定义构造方法，系统会给它定义一个默认构造方法

构造方法

華南師範大學
SOUTH CHINA NORMAL UNIVERSITY

在构造方法中可以设置实例对象的属性，在类的外部，通过实例对象可访问属性，方法：对象名.属性

```
class Circle : #定义Circle类
    def __init__(self): #定义构造方法,self对应当前的实例对象
        print('构造方法被调用！')
        self.radius='未知' #将self对应实例对象的radius属性赋值为"未知"

c= Circle() #创建实例c，自动执行构造方法，实例c传给self
print( f'半径：{c. radius}’ ) #输出实例对象c的radius属性的值
```

输出：构造方法被调用！

半径：未知

构造方法

构造方法中，除了self，也可以设置其他参数

```
class Circle : #定义Circle类
    def __init__(self,r): #定义构造方法
        print('构造方法被调用！')
        self. radius = r #将self对应实例对象的radius属性赋为形参r的值

if __name__=='__main__' :
    my_r=float(input('请输入半径'))#输入18
    c= Circle(my_r) #创建实例对象c,自动执行构造方法, c传给self
                    #18 传给r
    print(f'半径:{c.radius ') #输出实例对象c的属性值
```

输出：构造方法被调用！

半径：18

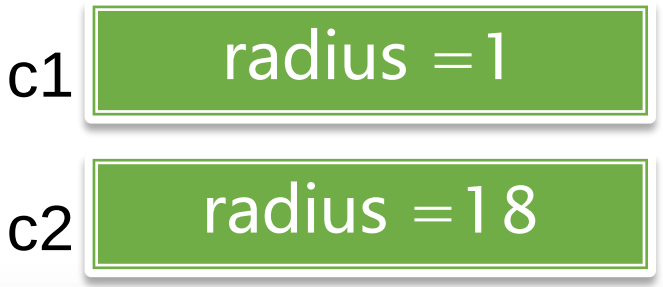
c radius=18

构造方法

构造方法也可以设置默认参数

```
class Circle : #定义Circle类
    def __init__(self,r=1): #定义构造方法
        print('构造方法被调用！')
        self. radius =r#将self对应实例对象的radius属性赋为形参r的值

if __name__=='__main__':
    c1= Circle() #创建Circle类的实例对象c1，自动执行构造方法
    c2= Circle(18) #创建Circle类的实例对象c2，自动执行构造方法
    print(f'  c1半径：{c1. radius} ') #输出半径
    print(f'  c2半径：{c2. radius} ')
```

```
构造方法被调用！
构造方法被调用！
c1半径：1
c2半径：18
```



构造方法

类规定了可以用于存储什么数据，而每个对象用于实际存储数据，每个对象可存储不同的数据。

构造方法

華南師範大學
SOUTH CHINA NORMAL UNIVERSITY

```
class Circle : #定义Circle类
        def __init__(self,r_____):
            self.radius=r

            _____________


c1= Circle(1_____) #创建Circle类对象c1
c2= Circle(18_____) #创建Circle类对象c2
```

c1
radius =1
color =red

c2
radius =18
color =blue

析构方法

华南师范大学
SOUTH CHINA NORMAL UNIVERSITY

析构方法是类的另一个内置方法，它的方法名为__del__，在销毁一个类的实例对象时会自动执行，常用于完成待销毁对象的资源清理工作，如关闭文件等。

```
class Circle : #定义Circle类
    def __init__(self,r): #定义构造方法
        self.radius=r #实例对象属性赋值
        print(f'半径为{self.radius}的对象被创建!')
    def __del__(self): #定义析构方法
        print(f半径为{self. radius}的对象被销毁!')
```

析构方法

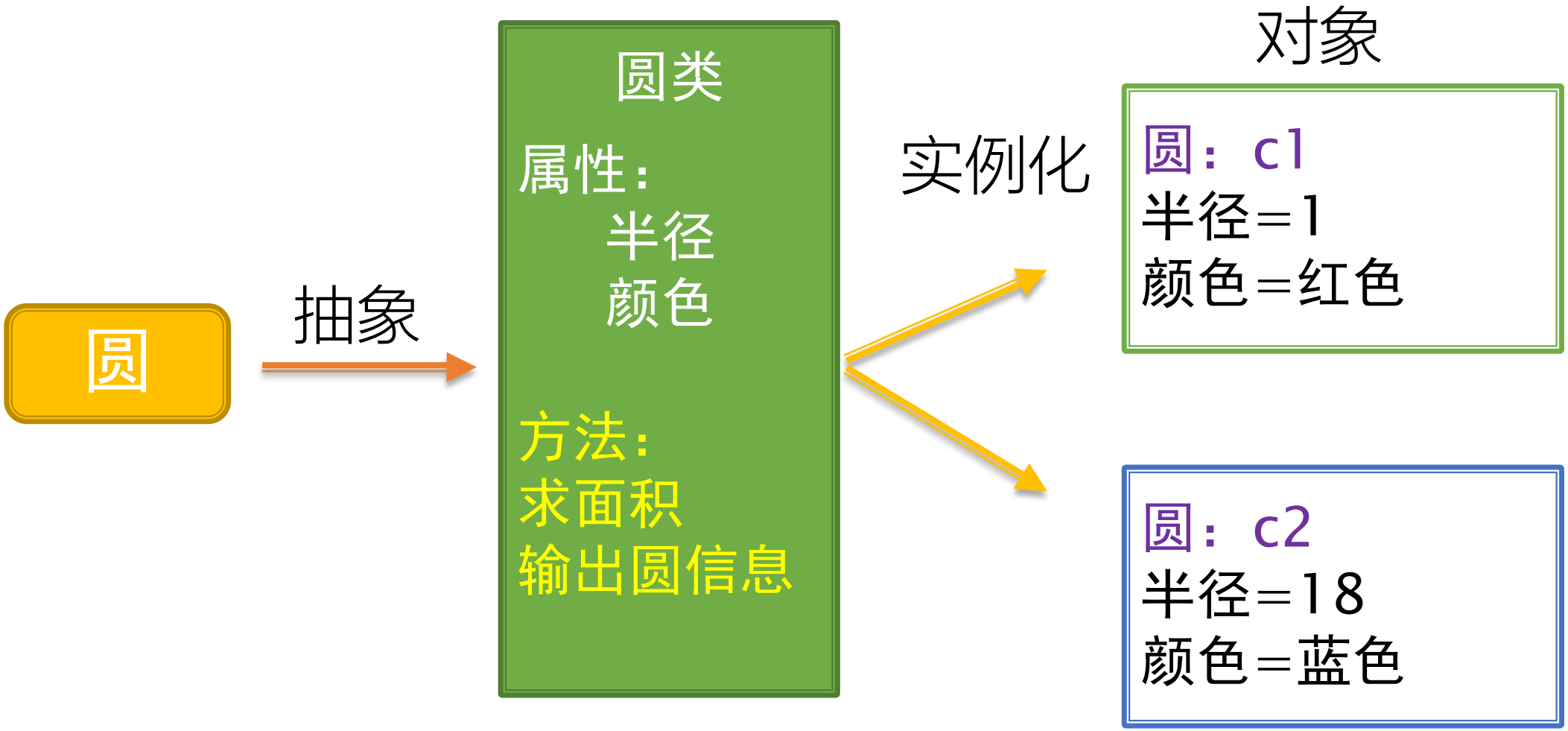

```
class Circle : #定义Circle类
    def __init__(self,r): #定义构造方法
        self. radius =r #实例对象属性赋值
        print(f'半径为{self. radius}的对象被创建!')
    def __del__(self): #定义析构方法
        print(f'半径为{self. radius}的对象被销毁!')


if __name__=='__main__':
    c1= Circle(1) #创建实例对象c1
    c2= Circle(18) #创建实例对象c2
    del c2 #使用del删除c2对象自动调用析构方法
    del c1 #使用del删除c1对象自动调用析构方法
```

在类的内部调用实例属性或方法时要用 self.
在类的外部调用实例属性或方法时要用 对象名.

```
半径为1的对象被创建!
半径为18的对象被创建!
半径为18的对象被销毁!
半径为1的对象被销毁!
```

课程目录
Course catalogue

实例方法



实例方法


```
class Circle : #定义Circle类
        def __init__(self,r,color):
            self.radius=r
            self.color= color


c1= Circle(1,'red') #创建Circle类对象c1
c2= Circle(18, 'blue') #创建Circle类对象c2
```

c1
radius =1
color =red

c2
radius =18
color =blue

实例方法定义及调用

- 类中的实例方法就是执行某种数据处理功能的函数。
- 与普通函数定义一样，类中的方法在定义时需要使用def 关键字。

实例方法定义及调用

在定义类的实例方法时，第一个参数需要对应调用方法时所使用的实例对象（一般命名为self，也可以改为其他名字）。在类的外部，当使用一个实例对象调用类的实例方法时，其语法格式为：

实例对象名.实例方法名(实参列表)

类中实例方法定义及调用

华南师范大学 SOUTH CHINA NORMAL UNIVERSITY

c1属性	c1方法（功能、操作）	c2属性	c2方法（功能、操作）
radius：5	构造方法__init()__	radius：10	构造方法__init()__
	析构方法__del__		析构方法__del__
color: red	求圆面积get_Area	color: blue	求圆面积get_Area
	输出圆的信息 print_Circle		输出圆的信息 print_Circle

```
PI=3.14
class Circle:
    def __init__(self, r,color):
        self.radius=r
        self.color= color
    def __del__(self):
        print('半径为',self.radius,"的圆被销毁")
    def get_Area(self):
        return self.radius**2*PI
    def print_Circle(self):
        print('半径=',self.radius,'面积=',self.get_Area())
```

```
if __name__=='__main__':
    c1=Circle(5,'red')
    c1.print_Circle()
    c2=Circle(18,'blue')
    c2.print_Circle()
    del c1
    del c2
```

Demo

定义Car类,要求：包括属性名称，公里数，剩余油量,构造函数为属性赋初值,析构函数输出适当信息,实例方法carRun()用于让车跑起来，且每跑1公里消耗1升汽油，剩余油量不足时，要提示用户；实例方法carAdd实现对汽车加油，汽车油箱最大容量70升；实例方法carPrint用于输出汽车的信息；创建实例验证上述功能。

属性	方法（功能、操作)
名称name	构造方法__init()__
公里数k	析构方法__del__
剩余油量v	让车跑起来carRun
	汽车加油carAdd
	输出汽车的信息carPrint

```
class Car:
    def __init__(self,name):
        self.name=name
        self.k=0
        self.v=0
    def __del__(self):
        print(f'汽车{self.name}被销毁')
    def carRun(self,k):
        if k>self.v:
            print(f"{self.name}当前油量不足以跑{k}公里")
        else:
            self.k+=k
            self.v-=k
            print(f“{self.name}跑了{k}公里，用了{self. k}升油")
    def carPrint(self):
        print(f'{self.name}当前公里数={self.k}，剩余油量={self.v}’)
```

```
    def carAdd(self,v):
        if self.v+v>70:
            self.v=70
            print(f"{self.name}加满了油")
        else:
            self.v+=v
            print(f“{self.name}加了{v}升汽油")
```

```
if __name__=='__ma
    c1=Car(“myCar")
    c1.carAdd(30)
    c1.carRun(30)
    c1.carPrint()
    del c1
```

Demo：写一个Student类,此类的对象有属性name, age, score, 用来保存学生信息:
1) 写一个函数input_student读入n个学生的信息,用对象来存储这些信息,并返回对象的列表;
2) 写一个函数output_student 打印这些学生信息;

```
class Student():
    def __init__(self, name, age, score):
        self.name = name
        self.age = age
        self.score = score
```

```
姓名:TOM
年龄:14
成绩:98
姓名:Lily
年龄:15
成绩:88
姓名:
姓名:TOM 年龄:14 成绩:98
姓名:Lily 年龄:15 成绩:88
```

```
def input_student():
    L = []
    while True:
        name = input("姓名:")
        if not name:
            break
        age = input("年龄:")
        score = input("成绩:")
        s =  Student(name, age, score)
        L.append(s)
    return L
def output_student(lst):
    for i in lst:
        print(f'姓名:{ i.name } 年龄:{ i.age } 成绩:{ i.score  }')

L = input_student()
output_student(L)
```

总结

- 面向对象的定义；
- 类的定义，类与对象的关系；
- 如何定义一个类，及创建类的实例对象的方法；
- 构造方法的定义，如何在构造方法中为实例对象初始化属性；
- 析构方法的定义；
- 实例属性在类的内部及外部如何使用？
- 如何定义实例方法？
- 实例方法在类的内部及外部如何调用？

下节内容预览

- 类属性及类方法的定义；
- 静态方法的定义；
- 了解继承；

作业

1. 定义Rect(矩形)类,要求：包括属性长，宽,构造函数为长、宽赋值,构造函数的默认参数值为0,析构函数输出适当信息,在类中定义一个实例方法setRect修改矩形的长、宽,定义一个实例方法get_Area返回矩形面积，定义一个实例方法print_Area用于输出矩形面积，并创建两个实例分别验证上述功能。

2. 设计一个 银行账户类:Account，该类包含三个属性：账号、用户名、余额。该类提供三个方法：存款、取款、转账。初始化时，账户余额为0，取款和转账前需判断余额是否充足，余额不足时，操作失败，输出相关提示信息。输出账户对象时，将会显示账号、用户名、余额等基本信息。

作业

2. 设计一个 银行账户类:Account，该类包含三个属性：账号、用户名、余额。该类提供三个方法：存款、取款、转账。初始化时，账户余额为0，取款和转账前需判断余额是否充足，余额不足时，操作失败，输出相关提示信息。输出账户对象时，将会显示账号、用户名、余额等基本信息。

```
a = Account("007", "张三")   # 创建账户
a.put(2000)   # 存款 2000
a.get(3000)   # 取款 3000
a.get(800)   # 取款 800
b = Account("009", "李四")   # 创建账户
a.transform(b, 500)   # 转账 500
b.transform(a, 1000)   # 转账 1000
```

```
账户创建成功， 账号为：007， 用户名为：张三， 余额为：0 元
成功存入 2000 ，账号为：007， 用户名为：张三， 余额为：2000 元
余额不足，取款失败，请调整取款金额！
成功取走 800 ，账号为：007， 用户名为：张三， 余额为：1200 元
账户创建成功， 账号为：009， 用户名为：李四， 余额为：0 元
账号 007 向账号 009 成功转了 500
账号为：007， 用户名为：张三， 余额为：700 元
账号为：009， 用户名为：李四， 余额为：500 元
余额不足，转账失败，请调整转账金额！
```