Chap10 Object-Oriented Programming

College of Computer Science
Nankai University

Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Introduction

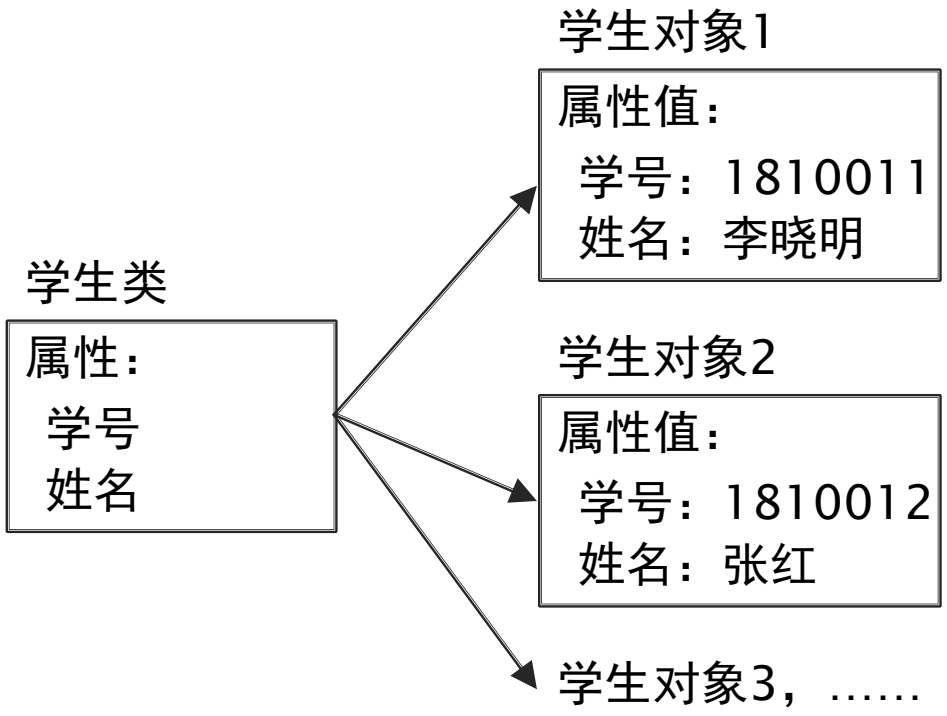
- 面向对象是当前流行的程序设计方法，其以人类习惯的思维方式，用对象来理解和分析问题空间，使开发软件的方法与过程尽可能接近人类认识世界、解决问题的思维方法与过程。
- 面向对象方法的基本观点是一切系统都是由对象构成的，每个对象都可以接收并处理其他对象发送的消息，它们的相互作用、相互影响，实现了整个系统的运转。

Introduction

- 类和对象是面向对象程序设计的两个重要概念。
- 类和对象的关系即数据类型与变量的关系，根据一个类可以创建多个对象，而每个对象只能是某一个类的对象。
- 类规定了可以用于存储什么数据，而对象用于实际存储数据，每个对象可存储不同的数据。

Introduction

与C/C++等语言不同，
Python中提供的基本
数据类型也是类，
如int、float等



Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Class and Object

- 在一个类中，除了可以包含前面所说的属性，还可以包含各种方法。
- 属性对应一个类可以用来保存哪些数据，而方法对应一个类可以支持哪些操作（即数据处理）。
- 类的定义形式多样
  - 既可以直接创建新的类，也可以基于一个或多个已有的类创建新的类；
  - 既可以创建一个空的类，然后再动态添加属性和方法，也可以在创建类的同时设置属性和方法。

Class and Object

- 类的定义

class 类名:
    语句1
    语句2
    ……
    语句N

- 定义一个空类

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

Class and Object

- 创建实例对象

```
class Student:
    pass

if __name__=='__main__':
    stu=Student()
    print(stu)
```

Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Class Property

- 定义类时指定类属性

```
class Student:
    name='Unknown'
```

- 对类属性的访问，既可以直接通过类名访问，也可以通过该类的对象访问，访问方式为

类名或对象名.属性名

Class Property

```
class Student:
    name='Unknown'
if __name__=='__main__':
    print('第4行输出：',Student.name)
    stu1=Student()
    stu2=Student()
    print('第7行输出：stu1 %s,stu2 %s‘ % (
stu1.name, stu2.name))
```

Class Property

- 类属性的访问

```
Student.name='未知'
print('第9行输出：',Student.name)
print('第10行输出：stu1 %s,stu2 %s‘ % (
stu1.name, stu2.name))
```

第9行输出： 未知

第10行输出：stu1 未知,stu2 未知

Class Property

- 类属性的访问

```
stu1.name='李晓明'
stu2.name='马红'
print('第13行输出：',Student.name)
print('第14行输出：stu1 %s,stu2 %s‘ % (
stu1.name, stu2.name))
```

第13行输出：未知

第14行输出：stu1 李晓明,stu2 马红

Class Property

- 类属性的访问

```
Student.name='学生'
print('第16行输出：',Student.name)
print('第17行输出：stu1 %s,stu2 %s'%(stu1.name,stu2.name))
```

第16行输出：学生

第17行输出：stu1 李晓明,stu2 马红

Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Class Method

- 类中的方法实际上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也需要使用def 关键字。
- 类中的方法分为两类：普通方法和内置方法。
  - 普通方法需要通过类的实例对象根据方法名调用；
  - 内置方法是在特定情况下由系统自动执行。

Class Method

- 在定义类的普通方法时，要求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，但也可以改为其他名字）。
- 当使用一个实例对象调用类的普通方法时，其语法格式为：

实例对象名.方法名(实参列表)

- 在通过类的实例对象调用类中的普通方法时，并不需要传入self参数的值，self会自动对应调用该方法时所使用的对象。

Class Method

```
class Student: #定义Student类
    name='Unknown'
    def SetName(self, newname):
        self.name=newname
    def PrintName(self):
        print('姓名：%s'%self.name)
if __name__=='__main__':
    stu1=Student()
    stu2=Student()
    stu1.SetName('李晓明')
    stu2.SetName('马红')
    stu1.PrintName()
    stu2.PrintName()
```

Quiz

- 练习一
  - 设计一个Circle类，包括一个名为radius的类属性
  - 设置radius值的方法，默认值为5
  - 一个名为getPerimeter方法返回周长
  - 一个名为getArea方法返回面积。
- 编写测试程序

Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Private Property

- 私有属性，是指在类内可以直接访问、而在类外无法直接访问的属性。
- Python中规定，在定义类时，如果一个类属性名是以__（两个下划线）开头，则该类属性为私有属性。

Private Property

```
class Student:
    name='未知'
    __id='未知'
    def SetInfo(self,newname,newid):
        self.name=newname
        self.__id=newid
    def PrintInfo(self):
        print('姓名：%s，身份证号：%s‘ %
(self.name, self.__id))
```

Private Property

```
if __name__=='__main__':
    stu=Student() #定义Student类对象stu
    stu.SetInfo('李晓明','120XXXXXXXXXXXXXXX')
    stu.PrintInfo()
    #print('身份证号：%s'%stu.__id) #取消前面的注释，则程序会报错
```

Private Property

- 实际上，Python中并不存在无法访问的私有属性。如果我们在类中定义了一个私有属性，则在类外访问该私有属性时需要在私有属性名前加上“_类名”。

Outline

- Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Constructor

- 构造方法是Python类中的内置方法之一，它的方法名为__init__，在创建一个类对象时会自动执行，负责完成新创建对象的初始化工作。

Constructor

```
class Student:
    def __init__(self):
        print('构造方法被调用！')
        self.name='未知'
    def PrintInfo(self):
        print('姓名：%s'%self.name)
if __name__=='__main__':
    stu=Student()
    stu.PrintInfo()
```

构造方法被调用！

姓名：未知

Constructor

```
class Student:
    def __init__(self,name='未知'):
        print('构造方法被调用！')
        self.name=name
    def PrintInfo(self):
        print('姓名：%s'%self.name)
```

Constructor

```
if __name__=='__main__':
    stu1=Student()
    stu2=Student('李晓明')
    stu1.PrintInfo()
    stu2.PrintInfo()
```

```
构造方法被调用！
构造方法被调用！
姓名：未知
姓名：李晓明
```

Quiz

- 设计一个Account类，它包括
  - 一个名为id的私有int属性，一个名为balance的私有float属性，一个名为annualInterestRate的私有float属性
  - 一个构造方法创建特定id(0), balance(100), annualInterestRate(0)
  - 三个私有属性的访问和修改方法
  - 一个名为withdraw的方法从账号里取钱
  - 一个名为deposit的方法向账号里存钱
- 编写测试程序

Outline

- OOP Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Destructor

- 析构方法是类的另一个内置方法，它的方法名为__del__，在销毁一个类对象时会自动执行，负责完成待销毁对象的资源清理工作，如关闭文件等。
- 类对象销毁有如下三种情况：
  - （1）局部变量的作用域结束。
  - （2）使用del删除对象。
  - （3）程序结束时，程序中的所有对象都将被销毁。

Destructor

```
class Student:
    def __init__(self,name):
        self.name=name
        print('姓名为%s的对象被创建！'%self.name)
    def __del__(self): #定义析构方法
        print('姓名为%s的对象被销毁！'%self.name)
```

Destructor

```
def func(name):
    stu=Student(name)
if __name__=='__main__':
    stu1=Student('李晓明')
    stu2=Student('马红')
    stu3=stu2
    del stu2
    func('张刚')
    del stu3
    stu4=Student('刘建')
```

```
姓名为李晓明的对象被创建！
姓名为马红的对象被创建！
姓名为张刚的对象被创建！
姓名为张刚的对象被销毁！
姓名为马红的对象被销毁！
姓名为刘建的对象被创建！
姓名为李晓明的对象被销毁！
姓名为刘建的对象被销毁！
```

Outline

- OOP Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Built-In Class Method

- 调用str函数对类对象进行处理时或者调用Python内置函数format()和print()时自动执行，__str__方法的返回值必须是字符串。

Built-In Class Method

```
class Complex:
    def __init__(self,real,image):
        self.real=real
        self.image=image
    def __str__(self):
        return str(self.real)+'+'+str(self.image)+'i'

if __name__=='__main__':
    c=Complex(3.2,5.3)
    print(c) #输出 “3.2+5.3i”
```

Built-In Class Method

- 比较运算的内置方法

内置方法	功能描述
__gt__(self, other)	进行self>other运算时自动执行
__lt__(self, other)	进行self<other运算时自动执行
__ge__(self, other)	进行self>=other运算时自动执行
__le__(self, other)	进行self<=other运算时自动执行
__eq__(self, other)	进行self==other运算时自动执行
__ne__(self, other)	进行self!=other运算时自动执行

Built-In Class Method

```
class Student:
    def __init__(self, name, age):
        self.name=name
        self.age=age
    def __le__(self, other):
        return self.age<=other.age
if __name__=='__main__':
    stu1=Student('李晓明',19)
    stu2=Student('马红',20)
    print('马红的年龄小于等于李晓明的年龄：',
stu2<=stu1)
```

马红的年龄小于等于李晓明的年龄： False

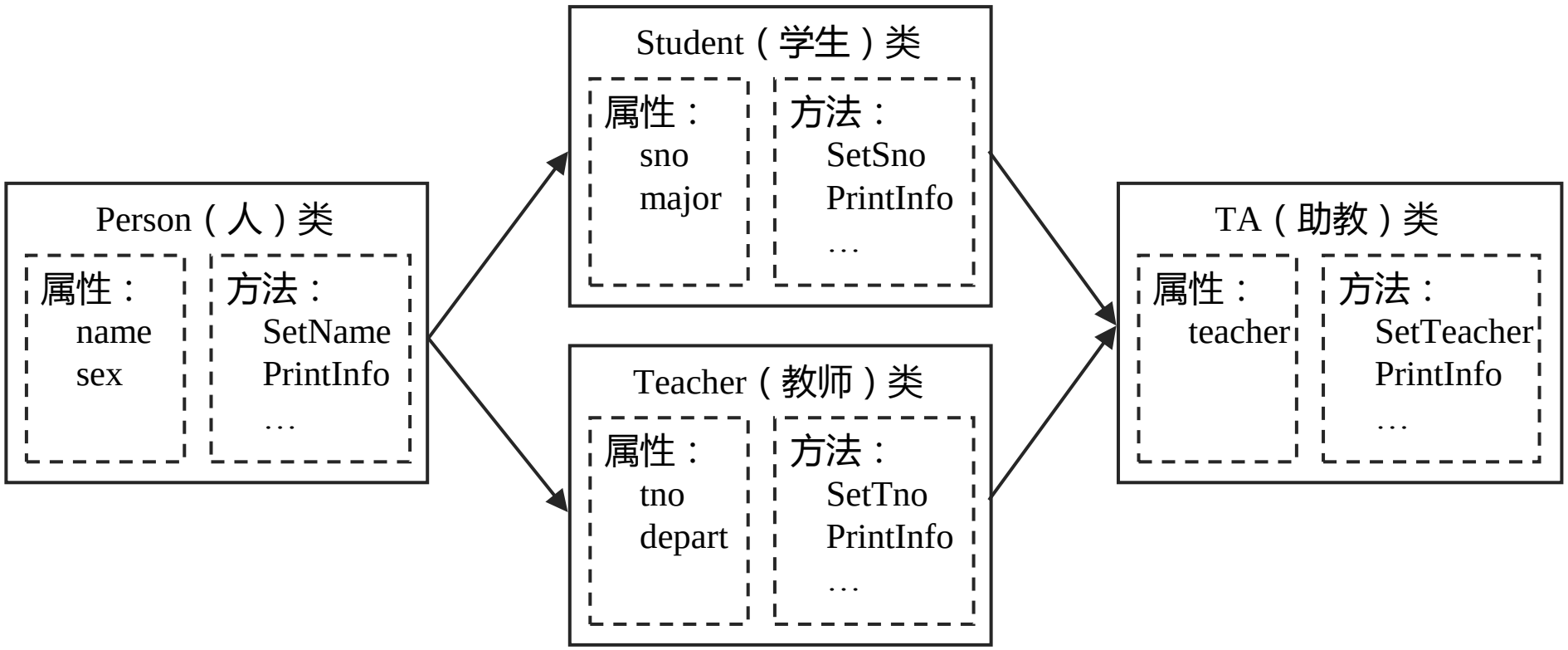
Outline

- OOP Introduction
- Class and Object(Instance)
- Class Property
- Class Method
- Private Property
- Constructor
- Destructor
- Built-In Class Method
- Inheritance

Inheritance

- 继承允许开发者基于已有的类创建新的类。
- 如果一个类C1通过继承已有类C而创建，则将C1称作子类（sub class），将C称做基类、父类或超类（base class、super class）。
- 子类会继承父类中定义的所有属性和方法，另外也能够在子类中增加新的属性和方法。
- 如果一个子类只有一个父类，则将这种继承关系称为单继承；如果一个子类有两个或更多父类，则将这种继承关系称为多重继承。

Inheritance



提示：需要结合具体的继承关系判断一个类是父类还是子类，一个类可能在一种继承关系中是子类、而在另一种继承关系中是父类。

Inheritance

```
class Person:
    def SetName(self, name):
        self.name=name

class Student(Person):
    def SetSno(self, sno):
        self.sno=sno
```

Inheritance

```
class Teacher(Person):
    def SetTno(self, tno):
        self.tno=tno

class TA(Student,Teacher):
    def SetTeacher(self, teacher):
        self.teacher=teacher
```

Inheritance

```
if __name__=='__main__':
    stu=Student()
    stu.SetSno('1810100')
    stu.SetName('李晓明')
    print('学号：%s，姓名：%s‘ % (
stu.sno,stu.name))
    t=Teacher()
    t.SetTno('998012')
    t.SetName('马红')
    print('教工号：%s，姓名：%s'%(t.tno,t.name))
```