第8章

面向对象程序设计

目录 Contents

1 Python的面向对象

2 定义和使用类

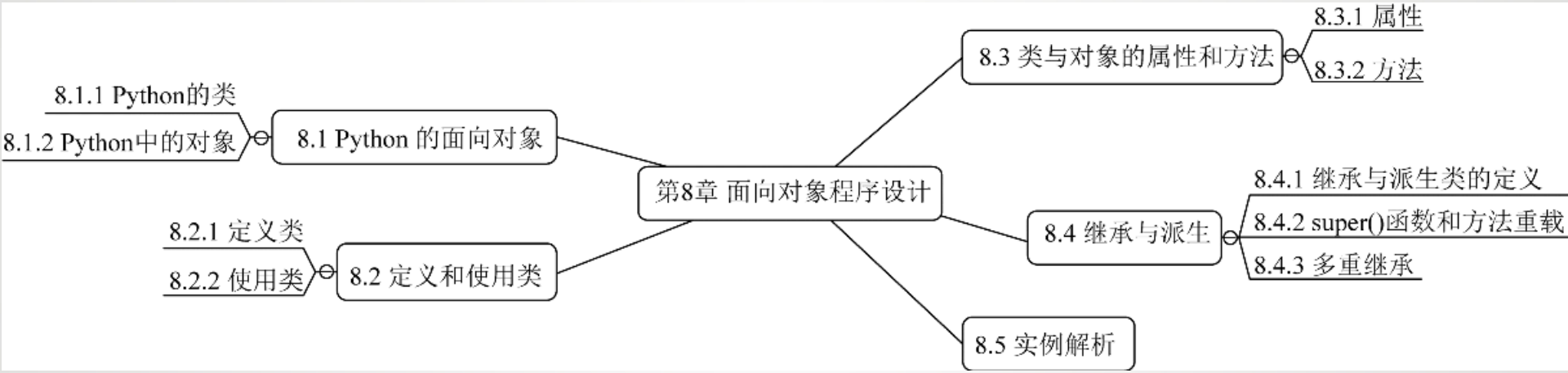
3 类与对象的属性和方法

4 继承与派生

知识点及难点

- Python的面向对象
- 定义和使用类
- 类与对象的属性和方法
- 继承与派生
- 难点是继承与派生

知识导图


第8章 面向对象程序设计
8.1 Python 的面向对象
8.1.1 Python的类
8.1.2 Python中的对象
8.2 定义和使用类
8.2.1 定义类
8.2.2 使用类
8.3 类与对象的属性和方法
8.3.1 属性
8.3.2 方法
8.4 继承与派生
8.4.1 继承与派生类的定义
8.4.2 super()函数和方法重载
8.4.3 多重继承
8.5 实例解析

8.1 Python的面向对象

01 Python的类

类也是一种数据结构

- 面向对象编程具有封装性、继承性以及多态性三大特征。
- 在现实世界中的同一类事物具有相同的特征和动作。
- 在前面章节中学习过的基础数据类型，它们也都是类

Python的类

- 例8.1 使用type()函数返回变量的对象类型。

```
i = 1
print(type(i))
s ="abc"
print(type(s))
l = [1,2,3]
print(type(l))
t = ('a', 'b', 1, 2)
print(type(t))
d = {'a': 1, 'b': 2}
print(type(d))
```

```
<class 'int'>
<class 'str'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
```

8.1 Python的面向对象

02

Python中的对象

对象是通过类定义的数据结构实例

- 对象（object）就是具有类属性和方法的具体事物，它由类来进行创建。
- 例如在Python中创建两个列表list1和list2，这两个列表就是对象，是由列表类（list）创建出来的两个实例，它们都共同拥有列表类的属性和方法。

8.2 定义和使用类

01

定义类

```
class 类名:
    <语句1>
    …
    <语句n>
```

在class后面定义类的名称，类名通常以大写字母开头。类中定义的语句主要是函数的定义，但是也允许是其他语句。类的定义与函数的定义类似，必须先定义才能使用。

定义类

- 例8.2 定义一个大学生类。

```
class UniversityStudent:
    pass
```

8.2 定义和使用类

02

使用类

实例名=类名()

实例名与变量命名类似，类名必须与定义的类名完全一致。

使用类

- 例8.3 使用例8.2的方法定义一个大学生类，再使用该类创建实例。

```
class UniversityStudent:
        pass
student1 = UniversityStudent()
student2 = UniversityStudent()
print(type(student1))
print(id(student1))
print(type(student2))
print(id(student2))
```

```
<class '__main__.UniversityStudent'>
1901064138936
<class '__main__.UniversityStudent'>
1901064999208
```

通过id()函数可以获取两个实例的内存地址（内存地址不是固定的，在不同机器上运行会有不同的结果）

8.3 类与对象的属性和方法

01 ▶ 属性

实例属性可以通过__init__()方法来绑定（init前后必须有两条下画线）。

如果把属性定义在类中（称为类属性），类的所有实例都可以访问这种属性。

- 不管是通过构造方法绑定或者是外部直接添加的属性，它们都与实例紧密相关，称为实例属性。
- 实例属性属于各个实例所有，相互独立；而类属性只有一个，创建的实例都共用这个类属性，一旦类属性改变就会影响所有的实例。
- 当实例属性和类属性重名时，实例属性优先级更高，它将屏蔽掉对类属性的访问。

属性

- 例8.4 定义一个包含学号、姓名、性别等实例属性的大学生类UniversityStudent，并使用该类创建实例。

```
class UniversityStudent:
    def __init__(self,id,name,sex):
      self.id = id
      self.name = name
      self.sex = sex
student1 = UniversityStudent("01","张三","男")
student2 = UniversityStudent("02","李思","女")
print("学号："+student1.id+" 姓名："+student1.name+" 性别："+student1.sex)
print("学号："+student2.id+" 姓名："+student2.name+" 性别："+student2.sex)
student1.id="03"
print("学号："+student1.id+" 姓名："+student1.name+" 性别："+student1.sex)
student2.age=20
print("学 号 ： "+student2.id+"  姓 名 ： "+student2.name+"  年 龄 ： "+str(student2.age))
```

```
学号：01 姓名：张三 性别：男
学号：02 姓名：李思 性别：女
学号：03 姓名：张三 性别：男
学号：02 姓名：李思 年龄：20
```

属性

- 例8.5 修改UniversityStudent类，把id变为私有属性。

```
class UniversityStudent:
    def __init__(self,id,name,sex):
        self.__id = id
        self.name = name
        self.sex = sex
student1 = UniversityStudent("01","张三","男")
print("学号: "+student1.id)
```

```
AttributeError: 'UniversityStudent' object has no attribute 'id'
```

如果要限制类中的某些属性，不让在类外部直接访问这些属性，那么可以在属性的名称前加上两条下画线__，使其变成一个私有属性，只能在类内部访问。

属性

- 例8.6 修改大学生类UniversityStudent，添加一个类属性count，用来统计一共创建了多少个实例。

```
class UniversityStudent:
    count = 0
    def __init__(self,name):
      self.name = name
      UniversityStudent.count += 1
student1 = UniversityStudent("王五")
print("学生数：",student1.count)
student2 = UniversityStudent("赵六")
print("学生数：",student2.count)
print("学生数：",UniversityStudent.count)
```

```
学生数： 1
学生数： 2
学生数： 2
```