面向对象编程

车万翔

哈尔滨工业大学

面向对象编程（Object-Oriented Programming）

- **传统面向过程的编程**
  - 随着软件项目规模的扩大，程序的关联性和依赖性呈指数级增加
- **解决方案**
  - 数据隐藏：隐藏程序的细节，仅需关注对象的功能和相关参数
- **对象**
  - 数据以及对数据操作方法的封装

内建类

- **Python中的所有数据类型都是内建类**
  - 如：int、float、str、llist、dict、set 等
  - s = 'hello world!'
  - type(s) ➔ <class 'str'>
  - s.replace('a', 'b')

  - a = 10
  - type(a) ➔ ?

用户自定义类

- **类（Class）**
  - 创建所有对象的模板，定义了所有对象（类的实例）共同具有的属性和方法
  - 程序中使用的是类所定义的对象，而不是类本身
- **类定义**

```
class ClassName:
    类方法列表
```

- **创建对象**

```
objectName = ClassName(arg1, arg2, . . .)
```

类定义示例

initializer method

instance variables

mutator methods

accessor methods

other methods

state-representation method

```
class Rectangle:
    def __init__(self, width=1, height=1):
        self._width = width
        self._height = height

    def setWidth(self, width):
        self._width = width

    def setHeight(self, height):
        self._height = height

    def getWidth(self):
        return self._width

    def getHeight(self):
        return self._height

    def area(self):
        return self._width * self._height

    def perimeter(self):
        return 2 * (self._width + self._height)

    def __str__(self):
        return ("Width: " + str(self._width)
         + "\nHeight: " + str(self._height))
```

对象创建与使用示例


```
import rectangle

# Create a rectangle of width 4 and height 5
r = rectangle.Rectangle(4, 5)
print(r)
print()
# Create a rectangle with the default values for width and height
r = rectangle.Rectangle()
print(r)
print()
# Create a rectangle of width 4 and default height 1
r = rectangle.Rectangle(4)
print(r)
```

对象列表

- 列表中的元素可以是任意类型，包括用户自定义类

```
import rectangle

lst = []
for width, height in zip(range(1, 10), range(5, 15)):
    rec = rectangle.Rectangle(width, height)
    lst.append(rec)

for rec in lst:
    print('Area: ', rec.area())
```

继承（Inheritance）

- 基于现有的类（超类、父类或基类）创建新的类（子类或派生类）
- 子类可以继承、增加或覆盖父类的属性或方法
- 子类定义

```
class SubClass(SuperClass):
    子类的方法列表
```

- 使用 super() 函数调用父类

继承类创建示例


```
from rectangle import Rectangle


class Cube(Rectangle):
    def __init__(self, width=1, length=1, height=1):
        super().__init__(width, length)
        self._height = height

    def setHeight(self, height):
        self._height = height

    def getHeight(self):
        return self._height

    def area(self):
        return 2 * (self._width * self._length + self._width * self._height + self._length * self._height)

    def perimeter(self):
        return 2 * super().perimeter() + 4 * self._height

    def volume(self):
        return super().area() * self._height

    def __str__(self):
        return super().__str__() + '\nHeight: ' + str(self._height)
```

多态（Polymorphism）

- 不同类中方法相同名相同，功能不同

```
import cube

c = cube.Cube(2, 3, 4)
print(c.area())

c = cube.Rectangle(2, 3)
print(c.area())
```