在Python中，类和实例是两种不同的对象类型，这点和C++/Java不同。

例：程序class\_detail/class\_detail1.py

class Dog:  
 def \_\_init\_\_(self, name, age):  
  
 self.name = name  
 self.\_\_age = age  
  
  
my\_dog = Dog('哈士奇', 2)  
print(type(Dog))  
print(type(my\_dog))

print(id(Dog))  
print(id(my\_dog))

输出为：

<class 'type'>

<class '\_\_main\_\_.Dog'>

2119825548616

2119861324544

class语句与def语句类似，class语句会产生一个新的类对象，每次类调用（Dog()）会产生一个新的实例对象。

类中的\_\_init\_\_方法称为构造函数。

类对象的特性

1. class语句创建类对象并将其赋值给变量名（class头部的变量名，即类名）。
2. class语句内的赋值语句（不在def之内）会创建类的属性。

实例对象：

1. 每次类调用都会产生新的实例对象
2. 每个实例对象继承类的属性并获得自己的命名空间。
3. 在方法内对self属性做赋值运算会产生每个实例自己的属性。在类方法内，第1个参数self会引用正处理的实例对象。对self的属性做赋值运算，会创建或修改实例内的数据，而不是类的数据。

例：程序class\_detail/class\_detail2.py

class FirstClass:  
 count = 0 # 类的属性，类似C++中的静态变量  
  
 def set\_data(self, value):  
 self.data = value # 每个实例对象都有data属性  
  
 def display(self):  
 print(self.data)  
  
  
x = FirstClass()  
FirstClass.count += 1  
y = FirstClass()  
FirstClass.count += 1  
  
x.set\_data("King Arthur")  
y.set\_data(3.14159)  
  
x.display()  
y.display()  
print(FirstClass.count)

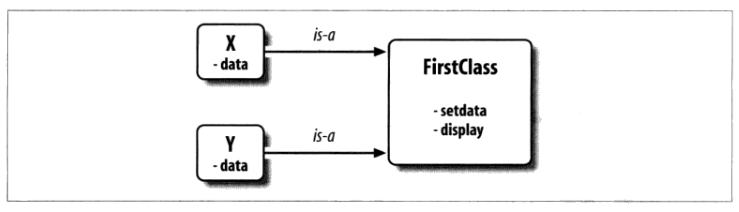
输出为：

King Arthur

3.14159

2

类和实例是类树中通过继承搜索相连的命名空间，data属性在实例内找到，setdata和display在它们之上的类中找到。



类的继承：

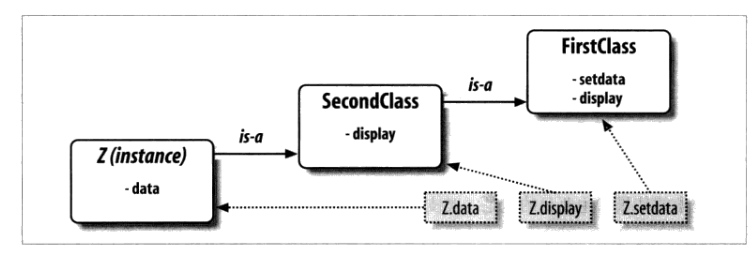
1. 超类（C++中的基类/父类）列在类开头的括号中，可以继承多个
2. 类从其超类中继承属性。
3. 实例会继承所有可读取类的属性。
4. 子类可以重新定义超类的变量名，子类就可取代并定制所继承的行为

例：程序class\_detail/class\_detail3.py

import class\_detail2  
  
  
# 类是模块内的属性  
class SecondClass(class\_detail2.FirstClass):  
 def display(self): # 重写了超类的方法  
 print('Current value = {0}'.format(self.data))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 z = SecondClass()  
 z.set\_data(42)  
 z.display()

输出为：

Current value = 42



类树中较低的子类中重新定义变量名，从而覆盖了继承的变量名并将其专有化，称为重载。

Python中，子类的方法只要名称与超类中的一致就构成重载，这与C++不同。

例：程序class\_detail/class\_detail3.py

class ThreeClass(class\_detail2.FirstClass):  
 def display(self, name): # 构成重载  
 print("value is {0} and {1}".format(self.data, name))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
   
 s = ThreeClass()  
 s.set\_data(53)  
 # s.display() # Error  
 s.display("dog")

输出为：

value is 53 and dog

Python中类也可以有私有函数和私有变量，在函数前面加两个下划线，函数就变成了私有函数。

class Student:

def \_\_init\_\_(self):

self.\_\_name = "Jam" ## 私有变量

self.\_\_private()

## 私有函数

def \_\_private(self):

print("No")

私有函数和私有变量几乎没有什么用处。