关于类的三大特性综述

类是用来描述具有相同属性和方法的对象的集合。其主要有三大特性，即继承、多态、封装（私有化），下文将分别进行阐述。

1 继承(Inheritance)

类的一个主要功能就是“继承”。“继承”一词从字面理解是对象A从对象B处获得某物。类似地，在面向对象编程语言里，是指一种可以使用现有类的所有属性、方法，无需重新编写,同时可在原来类的情况下，对某些方法属性进行扩展或重新定义。那么，通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。例如：子类B若要继承父类A，则在B定义括号中需写父类名称A。

```
class A:
    m=6
class B(A):
    pass
```

1.1 调用父类属性方法

```
class Father():
    def _init_(self):
        self.a='aaa'
    def action(self):
        print('调用父类的方法')
class Son(Father):
    pass

son=Son()          #子类Son继承父类Father的所有属性和方法
son.action()       #调用父类方法
son.a              #调用父类方法
```

1.2 重写父类属性方法

```
class Father():
    def _init_(self):
        self.a='aaa'

    def action(self):
        print('调用父类的方法')

class Son(Father):
    def _init_(self):
        self.a='bbb'
    def action(self):
```

```
12      print('重写父类的方法')
13
14  son=Son()          #子类Son继承父类Father的所有属性和方法
15  son.action()       #子类Son调用自身的action方法而不是父类的action方法
16  son.a
```

1.3 继承的分类

继承按照继承方式，一般分为单继承与多继承。单继承即是只从一个父类那继承，而多继承则是指继承的父类不止一个，可通过多级继承实现。按照实现方式可分为实现继承、接口继承和可视继承三类。实现继承是指使用父类的属性和方法而无需额外编码的能力；接口继承是指仅使用属性和方法的名称，但子类须提供实现的能力；可视继承是指子类使用父类的外观和实现代码的能力。

在考虑使用继承时，需注意两个类之间的关系应为“属于”关系,且在子类中不能定义和父类同名的实例变量。

2 多态(Polymorphisn)

继承让类与类之间产生类关系，提供了多态的前提。在编程语言和类型论中，多态指为不同数据类型的实体提供统一的接口。多态性是指具有不同功能的函数可以使用相同的函数名，这样就可以用一个函数名调用不同内容的函数。在面向对象方法中一般是这样表述多态性：向不同的对象发送同一条消息，不同的对象在接收时会产生不同的行为(即方法)。也就是说，每个对象可以用自己的方式去响应共同的消息(消息，就是调用函数，不同的行为就是指不同的实现，即执行不同的函数)。相比Java，Python中也支持多态，但是是有限的的支持多态性，主要是因为python中变量的使用不用声明，所以不存在父类引用指向子类对象的多态体现。实现多态，通常有两种方式:覆盖，重载。覆盖，是指子类重新定义父类的虚函数的做法。重载，是指允许存在多个同名函数，而这些函数的参数表不同。在python中没有重载的概念。

Python中多态的作用主要是让具有不同功能的函数可以使用相同的函数名，这样就可以用一个函数名调用不同内容(功能)的函数。同时，Python中多态的特点主要有以下几点： (1)只关心对象的实例方法是否同名，不关心对象所属的类型； (2)对象所属的类之间，继承关系可有可无； (3)多态的好处可以增加代码的外部调用灵活度，让代码更加通用，兼容性比较强； (4)多态是调用方法的技巧，不会影响到类的内部设计。其应用场景主要有以下两类:

a. 对象所属的类之间没有继承关系

下例中调用同一个函数fly(), 传入不同的参数（对象），可以达成不同的功能。

```
1   class Duck():                                   # 鸭子类
2       def fly(self):
3           print("鸭子沿地面飞起")
4
5   class Swan():                                   # 天鹅类
6       def fly(self):
7           print("天鹅在空中翱翔")
8
9   class Plane():                                  # 飞机类
10      def fly(self):
11          print("飞机起飞")
12
13  def fly(obj):                                   # 实现飞的功能函数
14      obj.fly()
15
16  duck = Duck()
```

```
fly(duck)

swan = Swan()
fly(swan)

plane = Plane()
fly(plane)

Output:
鸭子沿地面飞起
天鹅在空中翱翔
飞机起飞
```

b. 对象所属的类之间有继承关系。
下例中的多态性主要体现向同一个函数，传递不同参数后，可以实现不同功能。

```
class gradapa():
    def __init__(self,money):
        self.money = money
    def p(self):
        print("this is gradapa")

class father(gradapa):
    def __init__(self,money,job):
        super().__init__(money)
        self.job = job
    def p(self):
        print("this is father,重写了父类的方法")

class mother(gradapa):
    def __init__(self, money, job):
        super().__init__(money)
        self.job = job
    def p(self):
        print("this is mother,重写了父类的方法")
        return 100
#定义一个函数，函数调用类中的p()方法
def fc(obj):
    obj.p()

mother1 = mother(1000,"老师")
father1 = father(2000,"工人")
fc(mother1)
fc(father1)
print(fc(mother1))

Output:
this is mother,重写了父类的方法
this is father,重写了父类的方法
```

```
34 100
```

3 封装(Encapsulation)

封装，是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。数据封装就是将某些实例变量隐藏起来，其他程序不能直接访问。

在Python中，封装实现了对某些方法和属性的“私有化”，将其应用权限限制在某个区域内，外部无法调用。而要对某些特定对象进行私有化时，只需在相应的对象名字前加双下划线。然而，在定义一个类后，如果要对该类中已经私有化的对象进行调用，也可通过一些方法进行实现。比如：在类中专门定义一个函数（方法）用来return该私有化对象，这相当于在类中直接调用，再通过专门定义的函数对私有化对象的值进行return,见下例。

```
1 class Secret:
2     __name = 'yoga'                    # 加上双下划线进行私有化
3     def get_name(self):
4         return self.__name
5 A = Secret()
6 A.get_name()                           # 访问不到特性
```

上述方法不可在外部对私有化对象进行修改，因此可以通过使用函数property来对类中已经私有化的对象进行调用或者修改。有一些面向对象语言支持私有特性，这些特性无法从外部直接访问，需要编写getter和setter方法对这些特性进行操作。Python不需要getter和setter方法，其所有特性都是公开的。对于property函数，我们也可以通过用装饰器函数的方式来进行使用。

```
1 class foo:
2     def __init__(self):
3         self.name = 'yoda'
4         self.work = 'master'
5     def get_person(self):
6         return self.name,self.work
7     def set_person(self,value):
8         self.name,self.work = value
9     person = property(get_person,set_person)
10
11 A3 = foo()
12 A3.person
13 ('yoda', 'master')
14 A3.person = 'skylaer','programer'
15 A3.person
16 ('skylaer','programer')
```

对于面向对象的类，其封装可以隐藏实现细节，使得代码模块化；继承可以扩展已存在的代码模块（类）；它们的目的都是为了代码重用。而多态则是为了实现接口重用，为了类在继承和派生的时候，保证使用任一类的实例的某一属性时的正确调用。