# 面向对象

1、定义一个类 ：class 类名：

class Cat： //命名和java一样

def eat(self): //方法 方法名和一个冒号 参数中：self，相当于this，

print(“猫在吃鱼>>>>”) //注意 不用分号

初始化对象：def \_\_init\_\_(self): 创建对象： def \_\_new \_\_(cls)两个合并相当于构造方法功能

无参数构造方法

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

print(“呵呵，创建对象了我就会调用”)

有参数构造方法 等同于添加实例属性

def \_\_init\_\_(self,name,age):

self.name=name

self.age=age

创建对象 变量=方法名(): 方法中代码尾数不用分号结尾

创建无参对象

tom=Cat() //这样就创建了对象 在内存开辟了一片内存，返回对象的引用

创建有参数对象

tom=Cat(self.name,self.age)

属性 python有 类属性、实例属性、全局属性

创建对象来添加属性 也是实例属性

tom.name=”给tom对象添加了属性name”

tom.age=”给tom对象添加了属性age”

类属性是可以说和java定义一样：s=none //定义了一个null

调用方法

tom.eat() #注意不用分号结尾

重写了toString:def \_\_str\_\_(self): 相当于重写了toString方法

def \_\_str\_\_(self):

return“%s的年龄是：%d”%(self.name,self.age)

私有方法和属性:\_\_test1(self) 方法名前面加双下划线就是私有方法，

def \_\_user(self): //私有方法

print(“sdfsdf”)

def \_\_init\_\_(self,name,age):

self.name=name

self.\_\_age=age 私有属性写法

\_\_del\_\_ 类中

①没有对象引用此类的时候 最后调用此方法，②有对象引用，程序结束（虚拟机停止）最后善后也会调用此方法

def \_\_del\_\_(self):

pring(“只要类中有这个方法，用来善后的，上面两个条件会执行”)

检查类的对象引用 个数

第一 导包 import sys 第二 sys.getrefcount(放入对象)：注意 得到结果要减去1才是真正数量

2、继承 class Dog(Animal):

重写：和java的一样没区别

子类中的方法调用父类的方法：父类的类名.父类的方法名(self)

第一种 调用父类的方法

def bark(self):

pringt(“我是本类的方法”)

Dog.bark(self) //调用父类的方法

第二种 调用用父类的方法

def bark(self):

pringt(“我是本类的方法”)

super().bark() **//调用父类的方法**

多继承

class A(B,C): 用逗号隔开

注意：一个子类继承两个父类，两个父类以上都有同名的方法那么会通过\_\_mro\_\_显示的顺序调用方法

老师建议：尽量不要有相同的方法，避免不了自己来指定，使用：Dog.bark(self)操作

类名.\_\_mro\_\_:决定这 调用一个方法的时候，搜索的顺序，如果在某个类中找到了方法，那么就停止搜索

类名.\_\_mro\_\_:打印出下面的元祖：它会有一个算法C3算法算出调用那个

(<class ‘\_\_main\_\_.C’>, <class ‘\_\_main\_\_.A’>, <class ‘\_\_main\_\_.B’>, <class ‘\_\_main\_\_.Base’>, <class ‘object’ >)

多态

老师介绍的是 函数中的形参放入对象，进入的是什么对象它就是那个对象

类属性和实例属性区别 可以说类属性就是静态属性但又有所不同

class Test(object):

a = 100 #类属性

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

self.b = b #实例属性

t = Test(100)

#通过实例化对象访问 类属性

print("t.a = %d"%t.a)

#通过类名访问 类属性

print("Test.a = %d"%Test.a)

#通过实例化对象访问 实例属性

print("t.b = %d"%t.b)

#通过类名访问 实例属性

print("Test.b = %d"Test.b) #error 无法通过（类名.属性名）的方式访问实例属性

类属性的修改

类属性是所有实例共有的属性即静态属性，类似JAVA/C++中static，但是仅仅只是类似。

我们知道在JAVA中所有类对象共享static内数据的内存，也就是说通过修改static中的值会导致其他对象中static值

同样被修改，但在Python中并不是这样

class Test(object):

a = 100

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

pass

t1 = Test()

t2 = Test()

#1.通过实例对象t1修改类属性的值

print("修改前:t1.a = %d, t2.a = %d, Test.a = %d"%(t1.a, t2.a, Test.a)) 打印 ：t1.a = 100, t2.a = 100, Test.a = 100

t1.a += 100

print("修改后:t1.a = %d, t2.a = %d, Test.a = %d"%(t1.a, t2.a, Test.a))打印 ：修改后:t1.a = 200, t2.a = 100, Test.a = 100

结论 t1的a属性修改了，但是a2的没有改

类方法 @classmethod 可以说是静态方法的折中

class A(object):

num=0

定义类方法

@classmethod

def add(cls): 这里必须要有一个参数，

cls.num=100

调用操作

a=A()

A.add() //调用类方法

a.add() //也可以用创建的对象进行调用

print(A.num)

静态方法 @staticmethod 和类方法一模一样的操作

区别 方法中可以没有参数。 应该是共享数据的还没有测试

# 异常