**day13**

1.面向对象特性三之多态性：(思想来源于生活)

**多态**：一种事物的多种形态

使用需要先存在继承的关系

生活层面：饭吃了吗？（米饭，面条...）

喝酒了吗？（啤酒，白酒，红酒...）

买啥水果呢？（香蕉，苹果，橘子...）

人喜欢往大的类型说（宏观上的）

计算机层面：父类的引用指向子类的对象

子类的对象给到父类的引用

最常出现的位置(代码)：在函数的形参

**案例演示：**

**①animal.py文件下**

**class** Animal:  
 **def** \_\_init\_\_(self,name):  
 self.name = name  
  
 **def** eat(self):  
 **pass**

**②cat.py文件下**

**from** animal **import** Animal  
  
**class** Cat(Animal):  
 **def** \_\_init\_\_(self,name):  
 Animal.\_\_init\_\_(self,name)  
  
 **def** eat(self):  
 print(self.name + **"，正在进食..."**)

**③dog.py文件下**

**from** animal **import** Animal  
  
**class** Dog(Animal):  
 **def** \_\_init\_\_(self,name):  
 super().\_\_init\_\_(name)  
  
 **def** eat(self):  
 print(self.name + **"，正在进食..."**)

**④person.py文件下**

**class** Person:  
 *#喂食猫* **def** feedCat(self,cat):  
 cat.eat()  
 *#喂食狗* **def** feedDog(self,dog):  
 dog.eat() **假设人可以喂食上百种动物，那么我们就需要定义上百个feedXXX函数来实现吗？  
 不需要，一个函数即可；将口开的大点(喂食的动物的类型往父类型去描述)；函数为：feedAnimal(self,animal)** 强类型语言：Java多态性如下\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 class Animal{...}  
 class Dog extends Animal{...}  
 class Cat extends Animal{...}  
   
 class Person  
 //喂食狗  
 //public void feedDog(Dog d){...}  
 //喂食猫  
 //public void feedCat(Cat c){...}  
 //喂食动物  
 public void feedAnimal(Animal){...}  
 }  
 //实例化子类对象  
 Dog d = new Dog()   
 Cat c = new Cat()  
 Person p = new Person()  
   
 p.feedAnimal(d) --> Animal a = d Animal a = new Dog()  
 p.feedAnimal(c) --> Animal a = c Animal a = new Cat()  
   
 p.feedAnimal(new Cat())\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **将函数名和形参名都往大类型去描述定义，根据传入子类对象的不同，  
 执行的eat操作也不一样  
 def** feedAnimal(self,animal):  
 animal.eat()

**⑤多态性.py文件下**

**from** cat **import** Cat  
**from** dog **import** Dog  
**from** person **import** Person **测试模块：***#实例化person、dog、cat对象*p = Person()  
d = Dog(**'汪汪'**)  
c = Cat(**'咪咪'**)  
*# p.feedDog(d)  
# p.feedCat(c)*p.feedAnimal(d)  
p.feedAnimal(c)

2.**\_\_slots\_\_使用**：

作用：接受一个元祖对象，将可以为对象动态添加的属性名定义其中称为其元素；

目的：约束对象动态添加的属性内容

位置：定义class的内部，函数的外部

注意：\_\_slots\_\_和\_\_dict\_\_不能同时出现

**演示\_\_slots\_\_的使用：****class** Person: *动态为对象添加属性的范围为：name、age、height、weight* \_\_slots\_\_ = (**'name'**,**'age'**,**'height'**,**'weight'**)  
 *# pass  
  
#实例化对象*p = Person()  
  
*#动态为对象添加属性*p.name = **"张三"**p.age = 18  
  
print(p.name,p.age) *#得到：张三 18  
# print(p.\_\_dict\_\_) 这行会报错*

**由于sex不属于以上元祖中的元素，所以不能动态被添加***# p.sex = '男' 这行会报错*

3.自定义异常类：

实现：定义一个类继承已有的一个异常类型(通常：Exception)

**定义MyException继承Exception（自定义异常类）****class** MyException(Exception):  *#两个类发生继承关系* **def** \_\_init\_\_(self,msg): *#MyException类实例化有构造函数* *，此构*

*造函数需要传递报错信息msg* super().\_\_init\_\_(msg) *#将msg传给父类*Exception*进行解析*

**演示使用自定义异常类型：**

**from** myexception **import** MyException **class** Person:  
 **def** \_\_init\_\_(self,name,age,money):  
 self.name = name  
 self.\_\_age = age *#(年龄私有化)*  
 self.money = money  
  
 **def** setAge(self,age):  
 **if** age < 0 **or** age > 130:  
 **raise** MyException(**"出现异常了..."**) *#自定义手动抛出异常对象*  
 **else**:  
 self.\_\_age = age  
  
 **def** getAge(self):  
 **return** self.\_\_age  
  
 **def** \_\_str\_\_(self): *#重写*  
 **return "name：%s，age：%s，money：%s"** %(self.name,self.\_\_age,self.money)  
*#实例化对象*p = Person(**'jack'**,30,50000.0)  
p.setAge(-50)  
print(p)

4.@property和@属性名.setter的使用，可以理解为语法糖

此操作可以取代之前讲的getter和setter操作：

格式：参考老郭3-test.py

注意：此操作和之前的getter&setter我们在实际开发中保留一套即可；

**演示@property和@属性名.setter的使用：  
class** Person:  
 **def** \_\_init\_\_(self,name,age,money):  
 self.name = name  
 self.\_\_age = age  
 self.money = money  
 **def** setAge(self,age):  
 **if** age < 0 **or** age > 130:  
 **raise** MyException(**"出现异常了..."**)  
 **else**:  
 self.\_\_age = age  
 **def** getAge(self):  
 **return** self.\_\_age  
  
 @property  
 **def** age(self):  
 print(111111111)  
 **return** self.\_\_age  
 @age.setter  
 **def** age(self,age):  
 print(222222222)  
 self.\_\_age = age  
 **def** \_\_str\_\_(self):  
 **return "name：%s，age：%s，money：%s"** %(self.name,self.\_\_age,self.money)  
*#实例化对象*p = Person(**'jack'**,30,50000.0)  
*# p.setAge(-50)*print(p)  
*#调用简化版getter和setter操作看效果*p.age = 35  
print(p.age)  
  
p.setAge(45) *#getter&setter，@property和@属性名.setter可以共存的*print(p.getAge())

课间练习：

实现英汉字典(操作：输入中文得到英文)

分析：

封装类(YingHan)，思考：类中有哪些成员？

1).考虑属性：一个字典对象(dic1)、一个path对象

2).在创建对象的过程中将解析字符串数据的操作实现了 --> 意味着dic1中内容已经确定了

3).定义函数find/search函数，传入英文，返回/得到中文

**day14 做银行系统，员工管理系统项目**