**day11笔记**

面向对象：(思想)

面向：看、瞅、关

对象：个体、实体、实例、结果

面向：看、瞅、关注

过程：经过、经历、从头到尾...

将生活中的一些常见的事情使用面向过程和面向对象的方式分别解决：

模拟场景一：

使用面向过程的思想解决吃饭的问题？

步骤：

1).思考今天吃什么？

2).去买菜(货比三家)

3).摘菜

4).洗菜

5).切菜

6).炒菜

7).焖饭

8).吃饭

9).洗刷

使用面向对象的思想解决吃饭的问题？

步骤：

1).思考今天吃什么？

2).去饭店

①.调用服务员的点菜功能

②.将菜品告知后台大厨

③.大厨调用服务员的上菜功能

3).开始吃饭

4).结账走人(多种支付方式)

模拟场景二：

使用面向过程的思想解决买电脑的问题？(组装电脑)

步骤：

1).上网查阅一些参数

2).去电脑城(路上遇到一个gg和mm)

3).回家整理一番

4).去店里购买配置

5).安装硬件设备(监工)

6).安装需要的系统软件、应用软件

7).扛电脑回家(自己LOL)

使用面向对象的思想解决买电脑的问题？(组装电脑)

步骤：

1).找一个懂电脑的老手(老郭)，将需求告知他

2).将前给到老郭，"坐"等老郭凯旋回来

3).老郭将电脑扛回你家(一起LOL)

面试环节：谈一谈你对面向对象这种思想的理解？(逗比版)

面试官你好，现在您在面试我的这个过程中就是面向对象的思想，您在这个领域叱咤风云好多年，自己绝对有能力去独立解决您公司的一个项目，但是您是人不是神，您可能独立完成这个项目需要花费3-5年的时间，所以您选到了我们哥几个，只要给我们微薄的15k，我们帮您干，您就从一个参与者变成了指挥者，我们几个一年帮您搞定，找对象嘛，我们就是可以满足你这个需求的对象，所以您找到了我们，找到我们以后，您的工作量下去了，从参与者变成了指挥者

1.名词动词提炼法：

场景一：

大象装进冰箱

名词：大象、冰箱 类型：模糊的、不清晰的

动词：装进

伪代码：

步骤一：设计类

class 大象(object):

pass

class 冰箱(object):

开开():

...

装进(大象):

...

关闭():

...

步骤二：创建对象(实例化对象)

【注意】：

1).python语言创建对象省略了new关键字（不能写new）

2).以下的dx、bx属于对象名(变量名)，记录了对象在内存中的地址

dx = 大象()

bx = 冰箱()

步骤三：通过对象调用功能完成需求

bx.开开()

bx.装进(dx)

bx.关闭()

场景二：

猫抓老鼠

名词：猫、老鼠 --> 类型

动词：抓

伪代码：

步骤一：设计类

class 猫:

看(老鼠):

老鼠.看()

抓(老鼠):

老鼠.逃()

吃(老鼠):

老鼠.叫()

class 老鼠:

看():

。。。

逃():

。。。

叫():

。。。

步骤二：创建对象(实例化对象 == 对象实例化)

tom = 猫()

jerry = 老鼠()

步骤三：对象调用属性或函数，完成需求

tom.看(jerry) --> 参数传递：老鼠 = jerry

tom.抓(jerry)

tom.吃(jerry)

场景三：火车司机刹车 名词火车司机和车 设计两个类 动词刹 是给到车类的 注意

场景四：人开门

。。。

总结：遇到**面向对象**的问题，我们可以**考虑如下三个环节**(初学者)

1).侧重于设计类，定义属性、函数、...(可能需要花费大量的时间) --> 认为是面向过程的体现

2).创建对象(简单，一行代码搞定，但是内存比较复杂)

3).通过对象调用属性或者函数完成需求

总结：在之后的开发环境中，面向对象和面向过程是同时存在的

2.将伪代码变成真实的代码

需求：

红色的车正在行驶

名词：车

动词：行驶

类的成员：

属性：描述一类事物的特征、特点；可以理解为"静"的部分；在代码中的体现，就是定义变量

函数：描述一类事物的行为、功能；可以理解为"动"的部分；在代码中的体现，就是定义函数

步骤一：设计类

class 车:

#属性

颜色 = "red"

品牌 = "BMW"

车牌 = "沪A88888"

#函数

行驶(self):

...

停止(self):

....

步骤二：创建对象

car = che()

步骤三：对象调用属性或者函数完成需求

print(car.颜色,car.品牌,car.车牌)

car.行驶()

car.停止()

区分：类和对象？

类：具有一些列相同特征、行为的"事物"，它的表现是不具体的、不清晰的、模糊的概念

对象：从类中实例化得到，一个实实在在的个体，在内存中有体现；它的表现是具体的、清晰的、看得见摸的着的

类名也是标识符的一种：

满足规则和规范

对于其规范：我们需要遵从驼峰命名(多个单词组成，每个单词首字母大写，其余字母都小写) 例如：HelloWorld

**需求：红色的车正在行驶** *#1.设计类***class** Car(object): *#（object）可以省去*  
 *#属性* color = **"红色"** brand = **"BMW"** number = **"沪A88888"** *#函数* **def** run(self):  
 print(**'%s的%s，车牌为%s，正在飞速的行驶...'** %(self.color,self.brand,self.number))  
  
 **def** stop(self):  
 print(self.color,self.brand,self.number)  
 print(**'车停了...'**)  
*#2.创建对象*c1 = Car()  
*#3.对象调用属性*print(c1,type(c1)) *#得到<\_\_main\_\_.Car object at 0x000000000220D710> <class '\_\_main\_\_.Car'>*print(c1.color,c1.brand,c1.number)  
*#3.对象调用函数*c1.run() *#红色的BMW，车牌为沪A88888，正在飞速的行驶...*c1.stop()  *#红色 BMW 沪A88888*

*车停了...*

*#创建第二个对象*c2 = Car()  
print(c2,type(c2))  *#得到<\_\_main\_\_.Car object at 0x000000000272D7B8> <class '\_\_main\_\_.Car'>*  
print(c1 == c2) *#得到False ，比较的是地址，c1和c2的地址不一样*c2.color = **"白色"**c2.brand = **"BYD"**c2.number = **"京A66666"**print(c2.color,c2.brand,c2.number) *#白色 BYD 京A66666*  
print(c1.color,c1.brand,c1.number) *#红色 BMW 沪A88888*  
**在一个模块中可以创建对个对象，它们彼此之间是相互独立存在(堆空间有体现)，切互不干扰...**

c3 = c1  *#c1和c3的地址是一样的，共用，c1不调用了，但c3还指在堆山上*

*c1记录的地址又给到c3一份*

c1 = None

**此时思考一下：堆中有几个空间，存在垃圾空间吗？  
有1个空间，不存在垃圾空间；因为c1虽然被赋值为None了，但是c3仍然记录了堆中对象空间的地址(维护这层关系)  
【具体可以看视频理解】**c3.color = **"黑色"**c3.brand = **"AUDI"**c3.number = **"闵A55555"**print(c3.color,c3.brand,c3.number) *#黑色 AUDI 闵A55555*  
print(c1.color,c1.brand,c1.number) *#黑色 AUDI 闵A55555*

3.**给对象动态添加属性**：(python存在的，很多语言不支持)

作用：实例化某个对象成功后，"后天"给其单独追加一些必要的属性；此过程理解为：私人订制

时机以及格式：

在实例化结束后 例如：对象名 = 类名()

以对象名.追加属性名并且赋值操作 例如：对象名.属性名 = 值

【注意】：这种情况，开发用的并不是很多，但是要知道它是存在的；

*案例：设计人类，随便为其定义一些行为，假设以下代码在demo.py文件（模块）下，没有属性，只有函数***class** Person:  
 *#函数* **def** eat(self):  
 print(**"吃一个..."**)  
 **def** sleep(self):  
 print(**'睡一个...'**)  
 **def** drink(self):  
 print(**"走一个..."**)

*#此时在另一个text.py文件（模块）下导入demo.py模块下的Person类*

**from** demo **import** Person  
***#实例化对象*p1 = Person()**  
*#动态为p1对象添加3个属性：name、age、sex*p1.name = **"李雷"**p1.age = 24  
p1.sex = **"男"**p1.eat() *#吃一个...*  
p1.sleep() *#睡一个...*  
p1.drink() *#喝一个...*  
  
print(p1.name,p1.age,p1.sex) *#李雷 24 男*  
  
*#再实例化一个对象*p2 = Person()  
print(p2.name,p2.age,p2.sex) *#会报错，因为动态添加只是为某一个对象*

*添加而已*

4.构造函数：constructor

构造函数/构造方法/构造器

**格式：**

**\_\_init\_\_(self):**

执行时机：在创建对象时被执行

【注意】

一个对象只会被执行一次，何时调用由系统自行决定

作用：

1).创建对象

2).**为对象的属性赋值**

【注意事项】

1).在设计类的时候，如果不显示的定义构造函数(参数无所谓)，那么编辑器会自动提供了一个默认空参构造；一旦定义了，编辑器就不会再提供了

2).构造函数也是函数，同样没有函数重载现象

**class** Person(object):  
 *#三个属性&构造函数(三个参数，self不算在内)* **def** \_\_init\_\_(self,n,a,addr):  
 print(**'我是三个参数的构造...'**)  
 self.name = n *#self.name是属性，n是局部变量* self.age = a  
 self.address = addr

**def** details(self):  
 print(**'姓名为：%s，年龄为：%d，籍贯 是：%s'** %(self.name,self.age,self.address))

*#创建对象*p1 = Person(**"薇薇姐"**,38,**"上海"**)

p1.details()  *#姓名为：薇薇姐，年龄为：38，籍贯是：上海人*  
print(p1.name,p1.age,p1.address) *#薇薇姐 38 上海*

创建第二个对象，与p1互不干扰，共同存在与堆中

p2 = Person(**"老郭"**,29,**"上海"**)  
p2.details()

**以下带有会报错：**如果一个class中没有人为的去定义构造函数，就会默认给用空参数的构造

*#定义空参构造  
# def \_\_init\_\_(self):self后面没有加参数  
# print('我是空参数的构造...')*

**如果在一个class中自己显示的去定义了构造函数(不管几个参数)，  
那么编译器就不会提供一个默认空参数的构造了；***# p3 = Person()  
# print(p3)*

**【注意】：构造函数也是函数，同样没有函数重载这一说！！  
后者将前者覆盖掉...**

5.**析构函数**：

**格式：**

**\_\_del\_\_(self)：**

执行时机：

在程序结束前，将对象回收，清出内存

【注意】

有一个对象存在就会被执行一次，何时调用由系统自行决定

作用：

回收对象，清理内存

*演示析构函数的定义和使用：***class** Dog:  
 **def** \_\_init\_\_(self,name,age,color):  
 print(**'我是构造函数...'**)  
 self.name = name *#为了完美，更好的辨别，一般将变量和属性保持一样的名字* self.age = age  
 self.color = color  
 **def** \_\_del\_\_(self):  
 print(**'我是析构函数...'**)  
 **def** func(self):  
 print(**'我是func函数...'**)  
 d = Dog(**'如花'**,6,**'gold'**)  
d1 = Dog(**'旺财'**,3,**'white'**)  
d2 = Dog(**'来福'**,5,**'black'**)  
d2.func()

*#得到 我是构造函数...*

*我是构造函数...*

*我是func函数...*

*我是构造函数...*

*我是析构函数...*

*我是析构函数...*

*我是析构函数...*

6.\_\_str\_\_(self)函数：

作用：

对象实例化之后将数据给到对象名，此时如果打印对象名，

在控制台上我们看到的是整个对象的类型以及在内存中的地址(十六进制)，

但是我们在开发过程中，对于类型和地址并不关注；我们更希望看到的是**对象的各个属性内容，**

此时我们可以自己重新定义\_\_str\_\_(self)函数的函数体(就是**函数重写**)，

此函数有返回值，return后面的内容必须是str类型

执行时机：

在打印对象名/引用名时被触发，由系统决定何时调用 *演示\_\_str\_\_(self)函数的定义和使用：***class** Person(object):  
 **def** \_\_init\_\_(self,name,age,address):  
 self.name = name  
 self.age = age  
 self.address = address  
 **def** \_\_str\_\_(self):  
 **return '姓名为：%s，年龄为：%d，籍贯是：%s人'** %(self.name,self.age,self.address)  *#return后面必须是字符串数据*  
  
p = Person(**'韩梅梅'**,20,**'上海'**)  
print(p) *#姓名为：韩梅梅，年龄为：20，籍贯是：上海人*  
  
p.name = **'李雷'**print(p) *#姓名为：李雷，年龄为：20，籍贯是：上海人*

7.魔术函数：

\_\_开头并且\_\_结尾的函数，我们称为魔术函数；

特点：

调用执行都不需要程序员关注，系统自行决定

例如：

\_\_init\_\_、\_\_del\_\_、\_\_str\_\_ 构造 析构 重写

**案例演示：***#设计Road类，有一个属性长度(length)***class** Road:  
 **def** \_\_init\_\_(self,length):  
 self.length = length

*#设计Car类，有一个属性速度(speed)，行驶功能(run)* **class** Car(object):  
 **def** \_\_init\_\_(self,speed):  
 self.speed = speed  
 *#参数位置：接受一个Road类型的对象，此过程认为将Road类型和Car类发生一种关系(关联关系)* **def** run(self,road):  
 *#得到road的长度* length = road.length  
 *#循环让汽车跑一会* i = 10  
 **while** i >= 0:  
 print(**'汽车在水产路上飞速的行驶...'**)  
 time.sleep(1)  
 i -= 1  
 print(**'汽车在水产路上一个行驶花费：%s小时'** %(length/self.sp

eed))

*#实例化Road和Car对象*r = Road(30)  
c = Car(60)  
*#调用执行函数*c.run(r)

*#结果*

*#汽车在水产路上飞速的行驶...*

*#......(第一行的内容一共十一次)*

*#......*

*#汽车在水产路上飞速的行驶...*

*#汽车在水产路上一个行驶花费：0.5小时*