笔记

作者：Jeff

面向对象思想

1.面向对象思想设计

基于哲学观点：一切皆对象

面向过程: 一种编程思想, 注重的是解决问题的步骤.
面向对象: 一种编程思想, 注重的是解决某个问题中出现的对象.

2.面向过程和面向对象的区别

2.1面向过程

在生活案例中：

一种看待问题的思维方式，在解决问题的时候，侧重于问题是怎样一步一步解决的，然后亲力亲为的去解决

在程序中：

代码从上往下依次执行

各个模块之间的关系尽可能的独立的，当import的时候，加载的顺序也是从上往下依次加载

每个模块中的语句结构：顺序，分支，循环

2.2面向对象

在生活案例中：

一种看待问题的思维方式，侧重于找到一个具有特殊功能的个体，然后委托这个个体帮忙完成某件事情，这个个体就被称为对象

好处：可以将复杂的问题简单化，将程序员从执行者变成了指挥者

在程序中：

根据不同的需求执行代码【代码执行顺序不一定】

程序的流程完全由需求决定【对象】

思想：如果对象存在，则直接使用；如果对象不存在，则创建对象

注意：面向对象只是一种思想，并不是一门编程语言

Python是一门面向对象的编程语言，类和对象是 面向对象的核心

```
```

```
# 理解面向对象
# 示例： 小狗吃食物（闻一闻smell、舔一舔lick、咬一咬bite）
#      分别采用面向过程和面向对象来分析
#
# 面向过程： 先闻一闻，然后再舔一舔，最后再咬一咬 （注重过程）
# 面向对象： 小狗是一个对象，它可以闻一闻食物，可以舔一舔食物，可以咬一咬食物. （不注重过程，注重对象）

# 面向过程
'''
def smell():
print('闻一闻')

def lick():
print('舔一舔')

def bite():
print('咬一咬')

smell()
lick()
bite()
'''


# 面向对象
# 类：类名一般遵守大驼峰的写法
class Dog:
name = '旺财'

def smell(a):
     print(a.name, '闻一闻')

def lick(self):
     print(self.name, '舔一舔')

def bite(self):
     print(self.name, '咬一咬')

# 对象：是由类来创建
dog = Dog()
dog.smell()
dog.lick()
dog.bite()
```

3.类和对象【掌握】

1.类和对象的概念

> 类：多个具有特殊功能的个体的集合. 例如: 人类 狗 猫
>
> 对象：在一个类中，一个具有特殊功能的个体，能够帮忙解决某件特定的事情，也被称为实例【instance】
>
> 两者之间的关系：类用于描述某一类对象的共同特征，而对象是类的具体的存在
>
> 思考问题：先有类还是先有对象?
>
> 【在程序中使用的时候，一般是先定义类，然后创建对象】
>
> 举例:
>
> | 类（抽象的） | 对象（具体的） |
> | --- | --- |
> | 人 | 李四，赵四 |
> | 快递 | 韵达，中通，圆通 |
> | Hero | 蝙蝠侠，蜘蛛侠，美国队长。。 |

2.类的定义

> 语法:
>
> class 类名( ):
>
> 类体
>
> 说明:
>
> a.Python中使用class关键字定义类
>
> b.类名只要是一个合法的标识符即可，但是要求：遵循大驼峰命名法则【首单词的首字母大写，不同单词之间首字母大写】
>
> c.通过缩进区分类体
>
> d.类体一般包含两部分内容：属性和方法(属性就是描述一些静态信息的,比如人的姓名\年龄\性别等等，方法:一般用函数表示,用来实现具体的功能)
>
> 代码演示:

```
class Dog():
  # 类属性
  name = "旺财"
  sex = "male"

  # 类方法
  def eat(self):
        print(self.name, "吃肉!")
  def say(self):
        print("今年旺不旺：旺旺")
```

4.类中的方法和变量【掌握】

1.类中的方法和变量的定义

> 类中的方法和变量是为了描述事物的行为和特征
>
> 类中定义的方法被称为成员方法
>
> 类中定义的变量被称为成员变量，也被称为属性　[os.name]
>
> 成员变量：类具有的特征
>
> 成员方法：类具有的行为
>
> 类存在的意义：拥有相同特征和行为的对象可以抽取出来一个类，类的存在是为了创建一个具体的对象
>
> 代码演示：

```
class Dog():
  # 类属性
  name = "旺财"
  sex = "male"

  # 类方法
  def eat(self):
        print(self.name, "吃肉!")
  def say(self):
        print("今年旺不旺：旺旺")
```

2.类中方法和属性的使用

2.1创建对象【实例化对象】

> 已知类，通过类创建对象
>
> 对象的创建过程被对象的实例化过程
>
> 语法：变量名 = 值
>
> 　　对象名 = 类名()
>
> 代码演示：

```
class Dog():
 # 类属性
 name = "旺财"
 sex = "male"

 # 类方法
 def eat(self):
    print(self.name, "吃肉!")
 def say(self):
    print("今年旺不旺：旺旺")
```

```
# 通过Dog类创建对象
labuladuo = Dog()

#通过对象访问方法
labuladuo.eat()
labuladuo.say()

# 通过对象访问属性
print(labuladuo.name)
print(labuladuo.sex)
```

总结：

访问变量采用：对象名.属性名

访问方法采用：对象名.方法名(参数列表)

5.构造函数和析构函数

1.构造函数【掌握】

```
# 采用上面的方式创建对象【直接给成员变量赋值】，很多的类一般倾向于创建成有初始状态的
# __init__:构造函数【作用：创建对象，给对象的成员变量赋初始值】

# 构造函数：构造器
# 调用的时机：当一个对象被创建的时候，第一个被自动调用的函数
per = Person()

# 语法：
def __init__(self,args1,args2....):
  函数体

# 说明：
# a.之前的写法中并没有显式的定义__init__函数，说明系统默认提供了一个无参的构造函数
# b.args1,args2...一般设置的形参列表和成员变量有关
```

代码演示：

```
class GirlFriend():
    '''
    类属性：
    name = "小妹"
    age = 22
    '''

    # 构造函数 参数是对象相关的属性
    def __init__(self,name,age):
```

```
        # 对象属性
        self.name = name
        self.age = age
        print("构造函数的触发时机是:当创建对象的时候自动触发")

    # 对象方法
    def say(self):
        print(self.name,"对张哥喊: 来啊!")
    def sing(self):
        print("唱歌")

# 当创建对象的时候,会自动调用__init__()
xiaomei = GirlFriend("小妹",22)
xiaomei.say()
```

2.析构函数(了解)

```
# 与构造函数正好相反，当对象被销毁的时候自动调用的函数，被称为析构函数
# __del__:

# 删除变量:    del  变量名，此时可以触发析构函数的调用

# 使用情景：清理工作，比如关闭数据库，关闭文件等
```

代码演示:

```
class GirlFriend():
 '''
   类属性:
   name = "小妹"
   age = 22
 '''

 # 构造函数 参数是对象相关的属性
 def __init__(self,name,age):
     # 对象属性
     self.name = name
     self.age = age
     print("构造函数的触发时机是:当创建对象的时候自动触发")

 # 对象方法
 def say(self):
     print(self.name,"对张哥喊: 来啊!")
 def sing(self):
     print("唱歌")

 # 析构函数:触发时机是当对象被删除时,会被自动调用,释放内存
def __del__(self):
```

```
    print("脚本运行结束,释放内存")


# 当创建对象的时候,会自动调用__init__()
xiaomei = GirlFriend("小妹",22)
xiaomei.say()
```