# 面向对象编程

## 封装、继承、多态

class Person(object):
 def \_\_init\_\_(self, name, age):
 self.name = name
 self.age = age
 def print\_name(self):
 print self.name

## 组合与继承

优先使用组合（has a）而非继承(is a)

## 类变量和实例变量的区别

* 类变量由所有实例共享
* 实例变量由实例单独享有，不同实例之间不影响
* 当我们需要在一个类的不同实例之间共享变量的时候使用类变量

## classmethod/staticmethod区别

* 都可以通过Class.method()的方式使用
* classmethod第一个参数是cls，可以引用类变量
* staticmethod使用起来和普通函数一样，只不过放在类里去组织

class Person(object):
 Country = 'china'
 def \_\_init\_\_(self, name, age):
 self.name = name
 self.age = age

 def print\_name(self):
 print self.name

 @classmethod
 def print\_country(cls):
 print(cls.Country)

 @staticmethod
 def join\_name(first\_name, last\_name):
 return last\_name + first\_name

## 什么是元类

元类是创建类的类

* 元类允许我们控制类的生成，比如修改类的属性等
* 使用type来定义元类
* 元类最常见的一个使用场景就是ORM框架
* \_\_new\_\_用来生成实例，\_\_init\_\_用来初始化实例

例子：

# 元类继承自type
class LowercaseMeta(type):
 """ 修改类的书写名称为小写的元类"""
 def \_\_new\_\_(mcs, name, bases, attrs):
 lower\_attrs = {}
 for k, v in attrs.items():
 if not k.startswith('\_\_'):
 lower\_attrs[k.lower()] = v
 else:
 lower\_attr[k] =v
 return type.\_\_new\_\_(mcs, name, bases, lower\_attrs)

class LowercaseClass(metaclass=LowercaseMeta):
 BAR = True

 def Hello(self):
 print('hello')

print(dir(LowercaseClass))

LowercaseClass().hello()

参考链接:

https://www.cnblogs.com/huchong/p/8260151.html#\_label0 很详细的解释了call, init, new, 元类等知识

## 习题

### python中类方法、类实例方法、静态方法有何区别？

* 类方法: 是类对象的方法，在定义时需要在上方使用 @classmethod 进行装饰,形参为cls，表示类对象，类对象和实例对象都可调用
* 类实例方法: 是类实例化对象的方法,只有实例对象可以调用，形参为self,指代对象本身;
* 静态方法: 是一个任意函数，在其上方使用 @staticmethod 进行装饰，可以用对象直接调用，静态方法实际上跟该类没有太大关系

### python函数重载机制

函数重载主要是为了解决两个问题。 1。可变参数类型。 2。可变参数个数。 另外，一个基本的设计原则是，仅仅当两个函数除了参数类型和参数个数不同以外，其功能是完全相同的，此时才使用函数重载，如果两个函数的功能其实不同，那么不应当使用重载，而应当使用一个名字不同的函数。 好吧，那么对于情况 1 ，函数功能相同，但是参数类型不同，python 如何处理？答案是根本不需要处理，因为 python 可以接受任何类型的参数，如果函数的功能相同，那么不同的参数类型在 python 中很可能是相同的代码，没有必要做成两个不同函数。 那么对于情况 2 ，函数功能相同，但参数个数不同，python 如何处理？大家知道，答案就是缺省参数。对那些缺少的参数设定为缺省参数即可解决问题。因为你假设函数功能相同，那么那些缺少的参数终归是需要用的。 好了，鉴于情况 1 跟 情况 2 都有了解决方案，python 自然就不需要函数重载了。

## 参考资料

《python中的魔术方法》：https://juejin.im/post/5c9f9d436fb9a05e4f05780c