发现了个新版本。

Python面向对象

- 面向对象
  - OOP
  - 面向对象的程序设计把计算机程序视为一组对象的集合
  - 在Python中，所有数据类型都可以视为对象，当然也可以自定义对象
  - 万物皆为对象
- 概念
  - 类 Class
    - 从一组对象中提取相似的部分就是类，类所有对象都具有的特征和技能的结合体
    - class 类名(父类):
    - __init__
      - 初始化，自动调用
      - __init__方法的第一个参数永远是self，self就指向创建的实例本身
  - 实例 Instance
    - 类的实现
- 特点
  - 数据封装
    - 隐藏数据
      - 实例通过少量接口函数访问
    - 组装
      - "封装"就是将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体（即类）
    - 访问限制
      - 变量以__xx开头，变成了私有属性（private），只能内部访问
        - 类名__变量名，可以访问
        - 在子类定义的__x不会覆盖在父类定义的__x
      - 外部代码不能随意修改对象内部的状态
      - __xxx__是特殊变量，特殊变量是可以直接访问的，不是private变量
  - 继承
    - 子类可以继承父类
    - 类可以继承一个或多个父类
      - 广度优先寻找
    - 子类获得了父类的全部功能，公有属性和公有方法
    - 接口
      - 接口类中定义了一些接口名（就是函数名）且并未实现接口的功能，子类继承接口类，并且实现接口中的功能
    - 抽象类
      - 抽象类是一个特殊的类，它的特殊之处在于只能被继承，不能被实例化
      - metaclass=abc.ABCMeta
    - 子类调用父类方法
      - 父类名.父类方法()
      - super().__self__()
    - Mixin
      - 重继承这种设计通常称之为MixIn
      - MixIn的目的就是给一个类增加多个功能
  - 多态
    - 一类事物有多种形态
    - 子类重写父类的方法
    - 在不同的类实例中展现不同的效果
- 属性
  - 创建一个类就会创建一个类的名称空间，用来存储类的属性
  - 数据属性
    - 类的数据属性是共享给所有对象的
  - 函数属性
    - 类的函数属性是绑定到所有对象的
  - dir(类名) 查看属性，是一个列表
  - 特殊类属性
    - 类名.__name__ 类的名字
    - 类名.__dict__ 查看属性，是一个字典
    - 类名.__doc__ 类的文档字符串
    - 类名.__base__ 类的第一个父类
    - 类名.__bases__ 类所有父类构成的元组
    - 类名.__module__ 类定义所在的模块
    - 类名.__class__ 实例对应的类
  - __slots__
    - 限制该class实例能添加的属性
    - __slots__=('name','age') # 用tuple定义允许绑定的属性名称
    - 仅对当前类实例起作用，对继承的子类是不起作用
  - @property装饰器
    - 负责把一个方法变成属性调用
    - 把一个getter方法变成属性，只需要加上@property就可以
    - @property本身又创建了另一个装饰器@score.setter，负责把一个setter方法变成属性赋值
    - 常用来将属性变成 只读 或者 读写
- 方法
  - 绑定方法
    - 类方法：绑定到类的方法
      - classmethod装饰器装饰
      - 参数 cls 类本身
      - 为类量身定制
    - 实例方法：绑定到对象的方法
      - 没有被任何装饰器装饰的方法
      - 为对象量身定制
  - 非绑定方法
    - 不与类或对象绑定，类和对象都可以调用
    - 静态方法 staticmethod装饰器装饰
- 定制类
  - __str__
    - 获取对象的描述信息
    - __str__()返回用户看到的字符串，而__repr__()返回程序开发者看到的字符串
    - __repr__ = __str__
  - __iter__
    - 返回一个迭代对象
    - Python的for循环就会不断调用该迭代对象的__next__()方法拿到循环的下一个值，直到遇到StopIteration错误时退出循环
    - 斐波那契数列
  - __getitem__
    - 像list那样按照下标取出元素
    - __setitem__()
    - __delitem__()
  - __getattr__
    - 动态返回一个属性
    - def __getattr__(self,attr):
    - 返回函数也是完全可以的
  - __call__
    - 直接对实例进行调用，好比对一个函数进行调用一样
    - 可以判断一个对象是否是"可调用"对象 callable()
  - __del__
    - 析构函数
    - 执行del命令时调用
    - 当对象引用计数为0时, 自动调用
  - 枚举类
    - 枚举类型定义一个class类型，每个常量都是class的一个唯一实例
    - from enum import Enum
    - Month=Enum('Month',('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'))
    - for name,member in Month.__members__.items():
    - 精确地控制枚举类型 @unique
  - 使用元类
    - type()
      - 动态语言和静态语言最大的不同，就是函数和类的定义，不是编译时定义的，而是运行时动态创建的
      - type()函数可以查看一个类型或变量的类型
      - Hello=type('Hello',(object,),dict(hello=fn)) #创建Hello class 还可以创建出新的类型
    - metaclass 元类
      - 先定义类，然后创建实例 先定义metaclass，然后创建类
      - 先定义metaclass，就可以创建类，最后创建实例
      - class ListMetaclass(type):
        - def __new__(cls,name,bases,attrs):
          - __new__()方法接收到的参数
            - 当前准备创建的类的对象
            - 类的名字
            - 类继承的父类集合
            - 类的方法集合
        - return type.__new__(cls,name,bases,attrs)
      - class MyList(list,metaclass=ListMetaclass):

试用版 XMind:ZEN 试用版