**OOP概述**

# **0. OOP-Python面向对象**

* Python的面向对象
* 面向对象编程
  + 基础
  + 公有私有
  + 继承
  + 组合，Mixin
* 魔法函数
  + 魔法函数概述
  + 构造类魔法函数
  + 运算类魔法函数

# **1. 面向对象概述（ObjectOriented，OO）**

* OOP思想
  + 接触到任意一个任务，首先想到的是任务这个世界的构成，是由模型构成的
* 几个名词
  + OO:面向对象
  + OOA：面向对象的分析
  + OOD：面向对象的设计
  + OOI：xxx的实现
  + OOP：xxx的编程
  + OOA->OOD->OOI: 面向对象的实现过程
* 类和对象的概念
  + 类：抽象名词，代表一个集合，共性的事物
  + 对象：具象的事物，单个个体
  + 类跟对象的关系
    - 一个具象，代表一类事物的某一个个体
    - 一个是抽象，代表的是一大类事物
* 类中的内容，应该具有两个内容
  + 表明事物的特征，叫做属性(变量)
  + 表明事物功能或动作， 称为成员方法(函数)

# **2. 类的基本实现**

类的命名

* + 遵守变量命名的规范
  + 大驼峰（由一个或者多个单词构成，每个单词首字母大写，单词跟单词直接相连）
  + 尽量避开跟系统命名相似的命名

你如何声明一个类

* + 必须用class关键字
  + 类由属性和方法构成，其他不允许出现
  + 成员属性定义可以直接使用变量赋值，如果没有纸，许使用None
  + 案例 01.py

实例化类

变量 = 类名() #实例化了一个对象

访问对象成员

使用点操作符

obj.成员属性名称

obj.成员方法

可以通过默认内置变量检查类和对象的所有成员

对象所有成员检查

# dict前后各有两个下划线

obj.\_\_dict\_\_

类所有的成员

# dict前后各有两个下划线

class\_name.\_\_dict\_\_

# **3. anaconda基本使用**

* anaconda主要是一个虚拟环境管理器
* 还是一个安装包管理器
* conda list： 显示anaconda安装的包
* conda env list:显示anaconda的虚拟环境列表
* conda create -n xxx python=3.6: 创建python版本为3.6的虚拟环境，名称为xxx

# **4. 类和对象的成员分析**

* 类和对象都可以存储成员，成员可以归类所有，也可以归对象所有
* 类存储成员时使用的是与类关联的一个对象
* 独享存储成员是是存储在当前对象中
* 对象访问一个成员时，如果对象中没有该成员，尝试访问类中的同名成员， 如果对象中有此成员，一定使用对象中的成员
* 创建对象的时候，类中的成员不会放入对象当中，而是得到一个空对象，没有成员
* 通过对象对类中成员重新赋值或者通过对象添加成员时，对应成员会保存在对象中，而不会修改类成员

# **5. 关于self**

* self在对象的方法中表示当前对象本身，如果通过对象调用一个方法，那么该对象会自动传入到当前方法 的第一个参数中
* self并不是关键字，只是一个用于接受对象的普通参数，理论上可以用任何一个普通变量名代替
* 方法中有self形参的方法成为非绑定类的方法，可以通过对象访问， 没有self的是绑定类的方法， 只能通过类访问
* 使用类访问绑定类的方法时， 如果类方法中需要访问当前类的成员，可以通过 \_\_class\_\_成员名来访问

# **6. 面向对象的三大特性**

* 封装
* 继承
* 多态

## **6.1 封装**

* 封装就是对对象的成员进行访问限制
* 封装的三个级别：
  + 公开，public
  + 受保护的，protected
  + 私有的，private
  + public，private，protected不是关键字
* 判别对象的位置
  + 对象内部
  + 对象外部
  + 子类中
* [python中下划线使用](http://blog.csdn.net/handsomekang/article/details/40303207)
* 私有

私有成员是最高级别的封装，只能在当前类或对象中访问

在成员前面添加两个两个下划线即可

class Person():

# name是共有的成员

name = "liuying"

# \_\_age就是私有成员

\_\_age = 18

Python的私有不是真私有，是一种成为name mangling的改名策略 可以使用对象.\_classname\_attributename访问

* 受保护的封装 protected
  + 受保护的封装是将对象成员进行一定级别的封装，然后，在类中或者子类中都 可以进行访问，但是在外部不可以
  + 封装方法： 在成员名称前天机爱一个下划线即可
* 公开的，公共的 public
  + 公共的封装实际对成员没有任何操作，任何地方都可以访问

## **3.2 继承**

继承就是一个类可以获得另外一个类中的成员属性和成员方法

作用： 减少代码，增加代码的复用功能， 同时可以设置类与类直接的关系

继承与被继承的概念：

* + 被继承的类叫父类，也叫基类，也叫超类
  + 用于继承的类，叫子类，也叫派生类
  + 继承与被继承一定存在一个 is-a 关系

继承的语法，参见oop-2.ipynb

继承的特征

* + 所有的类都继承自object类，即所有的类都是object类的子类
  + 子类一旦继承父类，则可以使用父类中除私有成员外的所有内容
  + 子类继承父类后并没有将父类成员完全赋值到子类中，而是通过引用关系访问调用
  + 子类中可以定义独有的成员属性和方法
  + 子类中定义的成员和父类成员如果相同，则优先使用子类成员
  + 子类如果想扩充父类的方法，可以在定义新方法的同时访问父类成员来进行代码重用， 可以使用 [父类名.父类成员] 的格式来调用父类成员，也可以使用super().父类成员的 格式来调用

继承变量函数的查找顺序问题

* + 优先查找自己的变量
  + 没有则查找父类
  + 构造函数如果本类中没有定义，则自动查找调用父类构造函数
  + 如果本类有定义，则不在继续向上查找

构造函数

* + 是一类特殊的函数，在类进行实例化之前进行调用
  + 如果定义了构造函数，则实例化时使用构造函数，不查找父类构造函数
  + 如果没定义，则自动查找父类构造函数
  + 如果子类没定义，父类的构造函数带参数，则构造对象时的参数应该按父类参数构造

super

* + super不是关键字， 而是一个类
  + super的作用是获取MRO（MethodResolustionOrder）列表中的第一个类
  + super于父类直接没任何实质性关系，但通过super可以调用到父类
  + super使用两个方,参见在构造函数中调用父类的构造函数

单继承和多继承

* + 单继承：每个类只能继承一个类
  + 多继承，每个类允许继承多个类

单继承和多继承的优缺点

* + 单继承：
    - 传承有序逻辑清晰语法简单隐患少呀
    - 功能不能无限扩展，只能在当前唯一的继承链中扩展
  + 多继承：
    - 优点：类的功能扩展方便
    - 缺点：继承关系混乱

菱形继承/钻石继承问题

* + 多个子类继承自同一个父类，这些子类由被同一个类继承，于是继承关系图形成一个菱形图谱
  + [MRO](https://www.cnblogs.com/whatisfantasy/p/6046991.html)
  + 关于多继承的MRO
    - MRO就是多继承中，用于保存继承顺序的一个列表
    - python本身采用C3算法来多多继承的菱形继承进行计算的结果
    - MRO列表的计算原则：
      * 子类永远在父类前面
      * 如果多个父类，则根据继承语法中括号内类的书写顺序存放
      * 如果多个类继承了同一个父类，孙子类中只会选取继承语法括号中第一个父类的父类

构造函数

* + 在对象进行实例化的时候，系统自动调用的一个函数叫构造函数，通常此函数用来对实例对象进行初始化，顾名
  + 构造函数一定要有，如果没有，则自动向上查找，按照MRO顺序，直到找到为止

## **3.3 多态**

多态就是同一个对象在不同情况下有不同的状态出现

多态不是语法，是一种设计思想

多态性： 一种调用方式，不同的执行效果

多态： 同一事物的多种形态，动物分为人类，狗类，猪类

[多态和多态性](https://www.cnblogs.com/luchuangao/p/6739557.html)

Mixin设计模式

* + 主要采用多继承方式对类的功能进行扩展
  + [Mixin概念](https://www.zhihu.com/question/20778853)
  + [MRO and Mixin](http://blog.csdn.net/robinjwong/article/details/48375833)
  + [Mixin模式](https://www.cnblogs.com/xybaby/p/6484262.html)
  + [Mixin MRO](http://runforever.github.io/2014-07-19/2014-07-19-python-mixin%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0/)
  + [MRO](http://xiaocong.github.io/blog/2012/06/13/python-mixin-and-mro/)

我们使用多继承语法来实现Minxin

使用Mixin实现多继承的时候非常小心

* + 首先他必须表示某一单一功能，而不是某个物品
  + 职责必须单一，如果由多个功能，则写多个Mixin
  + Mixin不能依赖于子类的实现
  + 子类及时没有继承这个Mixin类， 也能照样工作，只是缺少了某个功能

优点

* + 使用Mixin可以在不对类进行任何修改的情况下，扩充功能
  + 可以方便的组织和维护不同功能组件的划分
  + 可以根据需要任意调整功能类的组合
  + 可以避免创建很多新的类，导致类的继承混乱

# **4 类相关函数**

* issubclass:检测一个类是否是另一个类的子类
* isinstance:检测一个对象是否是一个类的实例
* hasattr:检测一个对象是否由成员xxx
* getattr: get attribute
* setattr: set attribute
* delattr: delete attribute
* dir: 获取对象的成员列表

# **5. 类的成员描述符（属性）**

* 类的成员描述符是为了在类中对类的成员属性进行相关操作而创建的一种方式
  + get： 获取属性的操作
  + set：修改或者添加属性操作
  + delete： 删除属性的操作
* 如果想使用类的成员描述符，大概有三种方法
  + 使用类实现描述器
  + 使用属性修饰符
  + 使用property函数
    - property函数很简单
    - property(fget, fset, fdel, doc)
  + 案例参看notebook
* 无论哪种修饰符都是为了对成员属性进行相应的控制
  + 类的方式： 适合多个类中的多个属性共用用一个描述符
  + property：使用当前类中使用，可以控制一个类中多个属性
  + 属性修饰符： 使用于当前类中使用，控制一个类中的一个属性

# **6. 类的内置属性**

\_\_dict\_\_:以字典的方式显示类的成员组成

\_\_doc\_\_: 获取类的文档信息

\_\_name\_\_:获取类的名称，如果在模块中使用，获取模块的名称

\_\_bases\_\_: 获取某个类的所有父类，以元组的方式显示

# **7. 类的常用魔术方法**

* 魔术方法就是不需要人为调用的方法，基本是在特定的时刻自动触发
* 魔术方法的统一的特征，方法名被前后各两个下滑线包裹
* 操作类
  + \_\_init\_\_: 构造函数
  + \_\_new\_\_: 对象实例化方法，此函数较特殊，一般不需要使用
  + \_\_call\_\_: 对象当函数使用的时候触发
  + \_\_str\_\_: 当对象被当做字符串使用的时候调用
  + \_\_repr\_\_: 返回字符串，跟\_\_str\_\_具体区别请百度
* 描述符相关
  + \_\_set\_\_
  + \_\_get\_\_
  + \_\_delete\_\_
* 属性操作相关
  + \_\_getattr\_\_: 访问一个不存在的属性时触发
  + \_\_setattr\_\_: 对成员属性进行设置的时候触发
    - 参数：
      * self用来获取当前对象
      * 被设置的属性名称，以字符串形式出现
      * 需要对属性名称设置的值
    - 作用：进行属性设置的时候进行验证或者修改
    - 注意： 在该方法中不能对属性直接进行赋值操作，否则死循环
    - 参看案例
* 运算分类相关魔术方法
  + \_\_gt\_\_: 进行大于判断的时候触发的函数
    - 参数：
      * self
      * 第二个参数是第二个对象
      * 返回值可以是任意值，推荐返回布尔值
      * 案例

# **8. 类和对象的三种方法**

* 实例方法
  + 需要实例化对象才能使用的方法，使用过程中可能需要截止对象的其他对象的方法完成
* 静态方法
  + 不需要实例化，通过类直接访问
* 类方法
  + 不需要实例化
* 参看案例
* 三个方法具体区别自行百度

# **9. 所用软件**

* 画图软件是Faststone capture
* 有问题加群 158184562
* 官网地址 [www.tulingxueyuan.com](http://www.tulingxueyuan.com/)

# **10. 抽象类**

抽象方法： 没有具体实现内容的方法成为抽象方法

抽象方法的主要意义是规范了子类的行为和接口

抽象类的使用需要借助abc模块

import abc

抽象类：包含抽象方法的类叫抽象类，通常成为ABC类

抽象类的使用

* + 抽象类可以包含抽象方法，也可以包含具体方法
  + 抽象类中可以有方法也可以有属性
  + 抽象类不允许直接实例化
  + 必须继承才可以使用，且继承的子类必须实现所有继承来的抽象方法
  + 假定子类没有是现实所有继承的抽象方法，则子类也不能实例化
  + 抽象类的主要作用是设定类的标准，以便于开发的时候具有统一的规范

# **11. 自定义类**

* 类其实是一个类定义和各种方法的自由组合
* 可以定义类和函数，然后自己通过类直接赋值
* 可以借助于MethodType实现
* 借助于type实现
* 利用元类实现- MetaClass
  + 元类是类
  + 备用来创造别的类