day18

类变量（类属性）

绑定属性类的数据

类方法（@classmethod）

名词：

类：

类方法

类对象

对象（实例）

实例变量（实例属性）

实例方法

累的预置属性\_\_doc\_\_绑定文档字符串

——slots\_\_列表

让实例有固定的属性

静态方法 ：@staticmethod

继承（派生）

语法：

class B（A）:

pass

覆盖是优先使用子类的方法

显式调用被覆盖的方法

super

第十九天：

用于类的函数

issubclass(cls,class\_or\_tuple)判断一个类是否继承自其他的类，

如果此类cla是class或tuple中一个派生子类则返回True否则返回false

示例：

class A:

PASS

class B(A):

pass

class B (c):

pass

issubclass(C,B)

issubclass(B,A)

issubclass(A,C)

封装enclosure

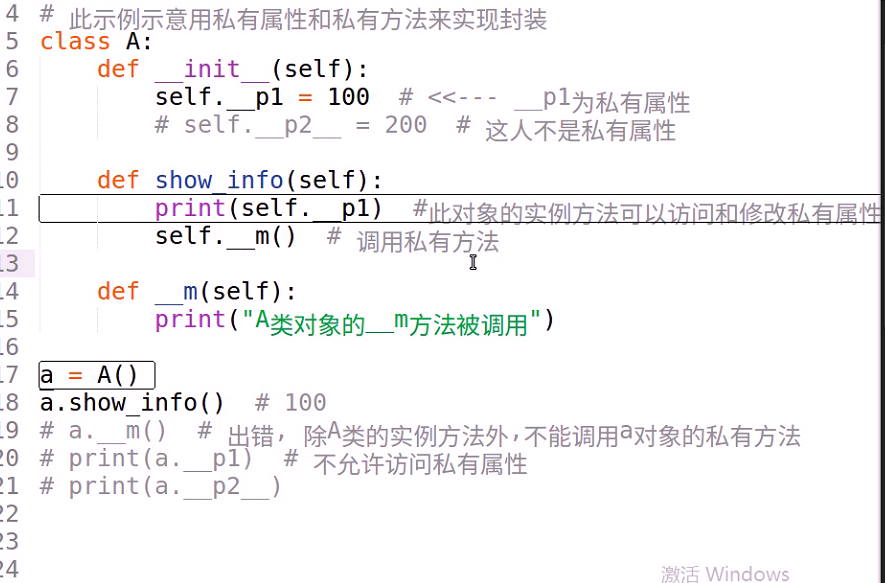
封装是指隐藏类的实现细节，让使用者不关心这些细节

封装的目的是让使用者通过尽可能少的方法（或属性）操作对象

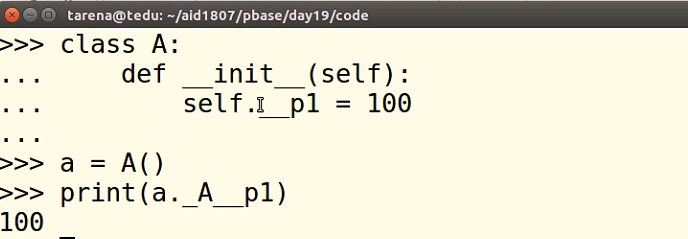
私有属性和方法

Python类中，以双下划线（‘\_\_’）开头，不以双下划线结尾的标识符为私有成员，私有成员只能使用该类的方法进行访问和修改

1. 以\_\_开头的属性为私有属性。
2. 以\_\_开头的方法为私有方法。



注：Python 的封装是假的封装（模拟的封装）



多态 polymorphic

字面意思：‘多种状态’

派生关系在有继承、派生关系的类中，调用基类对象的方法，实际能调用字类的覆盖方法的现象叫多态

状态：

静态（编译时状态）

动态（运行时状态）

说明：

1、多态调用的方法与对象相关，不写类型相关

2、Python全部对象只有“运行时状态（动态）”，没有“C++语言”里的编译时状态（静态）。

示例见：

面向对象的编程语言 的特征：

继承

封装

多态

多继承 multiple inheritance

多继承是指一个子类继承自俩个或两个以上的基类

语法：

Class 类名（基类名1，基类名2，。。。。）

说明：

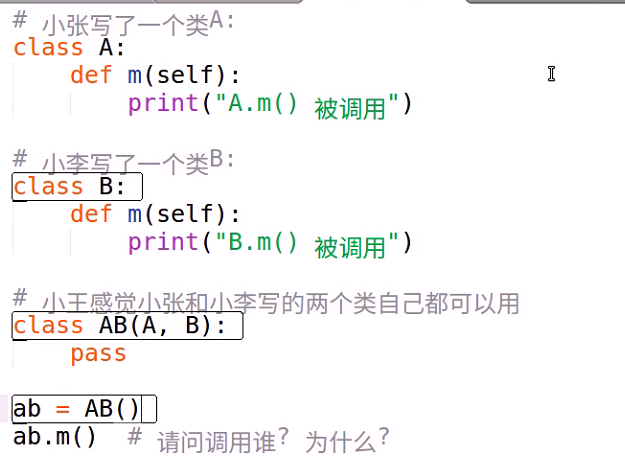
一个子类同时继承自多个父类，父类中的方法可以同时被继承下来如果两个父类中有同名的方法，而在子类中又没有覆盖此方法时，调用结果难以确定。

多继承的问题（缺陷）

标识符冲突问题

（要谨慎使用多继承）

示例：



A,最好不要用多继承

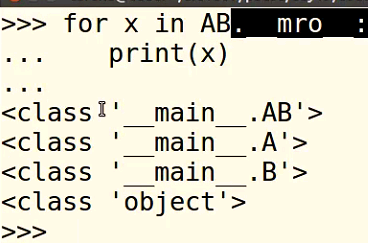
多继承的MRO（Method Resolution Order）

Python3类的\_\_mro\_\_属性

作用：用来记录类的方法查找顺序

mro很重要

记录类的查找顺序



Super（）函数就是根据\_\_mro\_\_来调用上层方法：

函数重写 overwrite

什么是函数重写

在自定义的类内添加相应的方法，让自定义的类创建的实例能够使用内数函数进行操作

对象转字符串函数

Repr（x）赶回一个能表示Python对象的表达式字符串，通常

Eval（repr(obj)）==obj

Str(x) 通过给定的对象返回一个字符串（这个字符串通常供人阅读）

示例：

S=”I’mTeacher”

Print(str(s)) #普通人可以阅读的字符串I’mTeacher

Print(repr(s)) #返回一个表示python3的字符串”I’mTeacher”

对象转字符函数的重写方法

Repr() 函数的重写方法：

der \_\_repr\_\_(self):

return 字符串

str（）函数的重写方法：

der \_\_str\_\_(self):

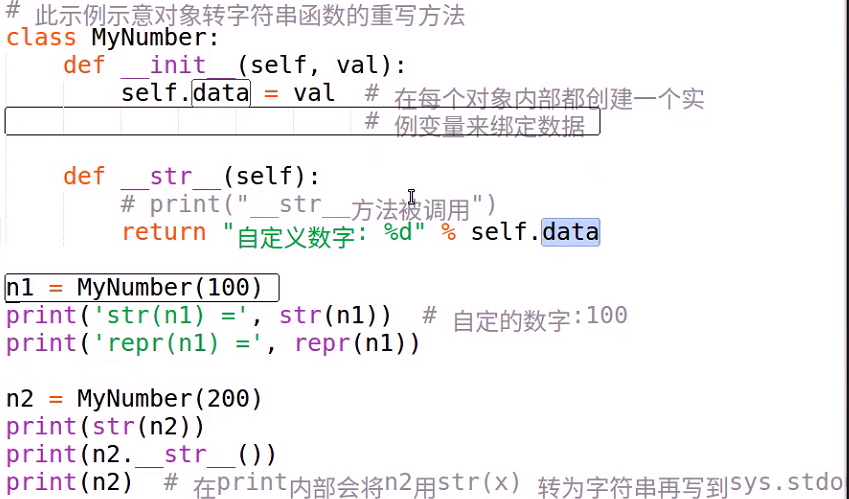
return 字符串

str （obj）函数调用方法说明：

1. str（obj）函数先查找obj.\_\_str\_\_(方法)，调用此方法并返回结果
2. 如果obj.\_\_str\_\_()方法不存在，则调用obj.\_\_repr\_\_方法并返回结果

3、如果obj.\_\_repr\_\_方法不存在，则调用object类的\_\_repr\_\_实例方法显示<\_\_main\_\_.XXX object at 0xXXXXXXX>格式的字符串

示例：

 可以自己制定函数返回来的东西，是重写的基础

内建函数重写

方法名 函数名

def \_\_abs\_\_(self) abs(obj) 函数调用

def \_\_len\_\_(self) len (obj) 函数调用

def \_\_reversed\_\_(self) reversed (obj) 函数调用

def \_\_round\_\_(self) vround (obj) 函数调用

示例：、、

数值转换函数的重写：

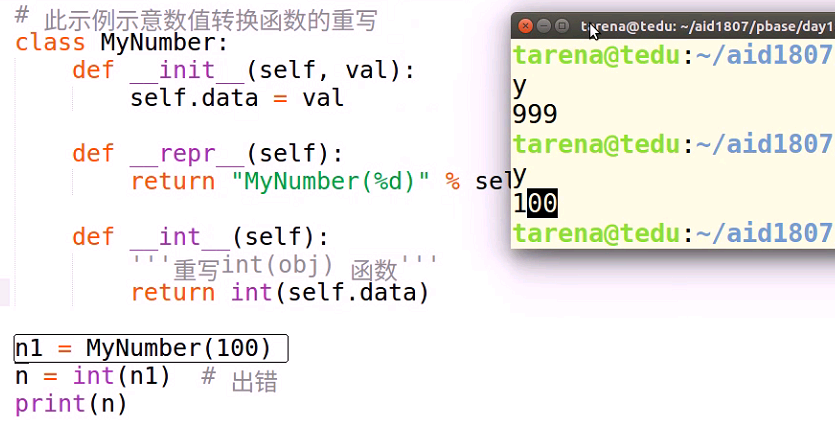
def \_\_ comples\_\_(self): complex(obj)

def \_\_int\_\_(self): int(obj)

def \_\_float\_\_(self): float(obj)

def \_\_bool\_\_(self): bool(obj)

示例见：



Self.data

布尔测试函数的重写

格式：

def \_\_bool\_\_(self):

…

作用：

用于bool（obj）函数取值

用于if语句真值表达式

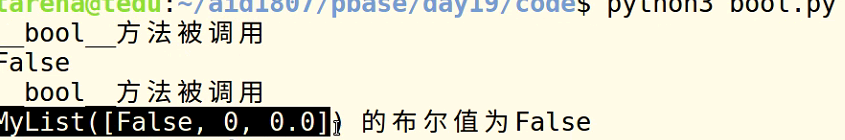
用于while语句的真值表达式

说明：

1、当自定义类内有\_\_boll\_\_(self)方法时，此方法的返回作为bool（x）的返回值

2、当不存在\_\_bool\_\_(self)方法时，返回\_\_len\_\_（self）方法的返回值是否为非零来测试布尔值

3、当不存在\_\_len\_\_(self)方法时，则直接返回True

对象的属性管理函数：

Getattr(obj,name[,default])

从一个对象用字符串name得到对象的属性，getattr(x,’y’)等同于x.y；当属性不存在时，如果给定default参数则返回default ，如果没有给出default 则触发一个attributeerror错误

Hasattr(obj,name)

用给定的name字符串返回obj是否有此属性，此种做法可以避免getattr(obj,name)时引发错误。

Setattr(obj,name,value),

给对象obj的名为name的属性设置相应的值为value，setattr(x,’y’,v)等同于 x.y=v

Delattr(obj,name)删除对象obj中的name属性 del（x,’y’）等同于del x.y

迭代器（高级）

什么是迭代器

由iter(x)返回，可以通过next（it）函数取值的对象就是迭代器

迭代器协议：

迭代器协议是指对象能能欧股使用next（）函数获取下一项数据，在没有下一个数据时触发一个stopiteration异常来终止迭代的约定。

迭代器协议的实现方法：

def \_\_naxt\_\_(self):

…

注：此方法需要实现迭代器协议

什么是可迭代对象

是指能用iter(obj)函数返回迭代器的对象（实例）

可迭代对象内部要定义\_\_iter\_\_(self)方法老返回迭代器的对象。