**OOP编程**

[零、OOP ：Python面向对象 2](#_Toc531879505)

[一、 面向对象概述（ObjectOriented，OO） 2](#_Toc531879506)

[1、 OOP思想 2](#_Toc531879507)

[2、 几个名词 2](#_Toc531879508)

[3、类和对象的概念 2](#_Toc531879509)

[二、类的基本实现 3](#_Toc531879510)

[1、类的命名 3](#_Toc531879511)

[2、如何声明一个类 3](#_Toc531879512)

[3、实例化类 4](#_Toc531879513)

[4、访问对象成员 4](#_Toc531879514)

[5、可以通过默认内置变量检查类和对象的所有成员 4](#_Toc531879515)

[三、Anaconda基本使用 4](#_Toc531879516)

[四、类跟对象的成员分析 5](#_Toc531879517)

[五、关于self 5](#_Toc531879518)

[六、面向对象的三个特性 9](#_Toc531879519)

[1、封装 9](#_Toc531879520)

# 零、OOP ：Python面向对象

1、Python的面向对象

2、面向对象编程

2.1 基础

2.2 共有私有

2.3 继承

2.4 组合，Minxi

3、魔法函数

3.1 魔法函数概述

3.2 构造类魔法函数

3.3 运算类魔法函数

# 一、 面向对象概述（ObjectOriented，OO）

## 1、 OOP思想

接触到任何一个任务，首先想到的是任务是这个世界的构成，是由模型构成的

## 2、 几个名词

2.1 OO: 面向对象

2.2 OOA: 面向对象的分析

2.3 OOD: 面向对象的设计

2.4 OOI: xxx的实现

2.5 OOP: xxx的编程

2.6 OOA -> OOD --> OOI：面向对象的实现过程

## 3、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：具象的东西，单个个体

类跟对象的关系

一个是具象，代表一类事物的某一个个体（对象）

一个是抽象，代表的是一大类事物（类）

类中应该具有两个内容

表明事物的特征，叫做属相（变量）

表明事物的功能或者动作，称为成员方法（函数）

# 二、类的基本实现

## 1、类的命名

（1） 遵守变量命名规范

（2） 大驼峰（由多个单词构成，收个单词字母大写，每个单词首字母大写）

（3） 计量避开跟系统命名相似的命名

## 2、如何声明一个类

（1） 必须使用class关键字

（2） 类由属性和方法构成，其他不允许出现

（3） 成员属性定义可以直接变量赋值，如果没有值，需要使用None

代码1：

*'''  
定义一个学生类，用来形容学生  
'''  
# 表明定义一个空的类***class** Student():  
 *# 一个空类，pass代表直接跳过  
 # 此处pass必须有* **pass** *# 定义一个对象*mingyue = Student()  
  
*# 再定义一个类：用来描述挺Python的学生***class** PythonStudent():  
 *# 用None给不确定的值赋值* name = **None** age = 18  
 course = **"Python"** *# 做作业  
 # 需要注意  
 # 1. def doHomework 的缩进层级  
 # 2. 系统默认有一个self参数* **def** doHomework(self):  
 print(**"i 在 做作业"**)  
  
 *# 推荐在函数末尾使用return语句* **return None***# 实例化一个叫yueyue的学生，是一个具体的人*yueyue = PythonStudent()  
print(yueyue.name)  
print(yueyue.age)  
*# yueyue.course = "Python"  
# 注意成员函数的调用没有传入参数*yueyue.doHomework()

## 3、实例化类

变量 = 类名() #实例化了一个对象

## 4、访问对象成员

使用点操作符obj.成员属性名称 obj.成员方法

## 5、可以通过默认内置变量检查类和对象的所有成员

（1） 对象所有成员检测

\_\_dict\_\_：dict前后各有两个下划线

obj.\_\_dict\_\_

（2） 类所有成员

classname.\_\_dict\_\_

# 三、Anaconda基本使用

（1） Anaconda主要是一个虚拟环境器

（2） 还是一个安装包管理器

（3） conda list ：显示anaconda安装的包

（4） conda env list ：显示anaconda的虚拟环境表

（5） conda create -n xxx python=3.6 ：创建Python版本为3.6.5的虚拟环境，名称为xxx

# 四、类跟对象的成员分析

（1） 类和对象可以存储成员，成员可以归类所有，也可以归对象所有

（2） 类存储成员使用的是与类关联的一个对象

（3） 对象存储成员 是存储在当前对象中

（4） 对象访问一个成员是，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，一定访问此成员

（5） 创建对象的时候，类中成员不会放入对象当中，而是得到一个空对象，没有成员

（6） 通过对象对类中成员重新复制或者通过对象添加成员的时候，对应成员会保存在对象中，而不会修改原有类成员变量

# 五、关于self

（1） 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当中。

（2） self并不是关键字，只是一个用于接收对象的普通参数，理论上可以用任何一个普通变量名代替。

（3） 方法中有self形参的方法称为非绑定类的方法，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。

（4） 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\_\_成员名访问。

代码1：

class A():

name = "dana"

age = 18

# 注意有一个参数self

def say(self):

self.name = 'aaaa'

self.age = 200

# 此案例说明

# 类实例的属性和其对象的属性在不对对象的实例属性赋值的前提下,指向同一个变量

# 此时，A称为类实例

print(A.name) ------dana

print(A.age) ------ 18

print("\*"\*20) --------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Id可以鉴别一个变量跟另外一个变量是否是同一个变量

print(id(A.name)) -------1234567

print(id(A.age)) -------7654321

print("\*"\*20) --------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

a = A()

print(A.\_\_dict\_\_)

----{'\_\_module\_\_': '\_\_main\_\_', 'name': 'dana', 'age': 18, 'say': <function A.say at 0x0000000004E9D0D0>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'A' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'A' objects>, '\_\_doc\_\_': None}

print(a.\_\_dict\_\_) ----{}

print(a.name) -----aaaa

print(a.age) -----200

print(id(a.name)) -------1234567

print(id(a.age)) -------7654321

print("\*"\*20) --------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

a.name = "月月"

a.age = 11

print(a.name) ------月月

print(a.age) ------11

print(id(a.name)) -------131231412

print(id(a.age)) -------1932718923

print(a.\_\_dict\_\_) ---------{'name': '月月', 'age': 11}

代码2：

class Student():

name = "dana"

age = 18

def say(self):

self.name = "danaSay"

self.age = 200

print("My name is {0}".format(self.name)) -----My name is danaSay

print("My age is {0}".format(self.age))---My age is 200

def sayAgain(s):

s.name = "danaSay"

s.age = 200

print("My name is {0}".format(s.name)) --my name is danaSay

print("My age is {0}".format(s.age)) -- My age is 200

A = Student()

print(A.say()) ---My name is dana My age is 200 None

print(A.sayAgain()) ---My name is danaSay My age is 200 None

代码3：

class Teacher():

name = "dana"

age = 19

def say(self):

self.name = "yaona"

self.age = 17

print("My name is {0}".format(self.name))

# 调用类的成员变量需要使用\_\_class\_\_

print("My age is {0}".format(\_\_class\_\_.age))

def sayAgain():

print(\_\_class\_\_.name)

print(\_\_class\_\_.age)

print("Hello, nice to meet you again")

t = Teacher()

t.say() -------My name is yaona My age is 19

# 调用绑定类的函数使用类名

Teacher.sayAgain() ----My name is dana My age is 19 Hello,Nice ...

代码4：

# 关于self的案例

class A():

name = "liuying"

age = 18

def \_\_init\_\_(self):

self.name = "aaa"

self.age = 200

def say(self):

print(self.name)

print(self.age)

class B():

name = "bbb"

age = 90

a = A()

# 此时，系统会默认把a作为第一个参数传入参数

a.say() ---aaa 200

# 此时，self被a替换

A.say(a) ---aaa 200

# 同样，可以把A作为参数传入

A.say(A) ---liuying 18

# 此时，传入的是类实例B，因为B具有name和age属性，所以不会报错

A.say(B) ----bbb 90

# 以上代码，利用了鸭子模型（不关心你是不是一个鸭子，只要你具有鸭子的形态和动作，你就是个鸭子—）

# 六、面向对象的三个特性

（1） 封装

（2） 继承

（3） 多态

## 1、封装

（1） 封装就是对对象的成员进行访问限制

（2） 封装的三个级别

【1】公开 public

【2】受保护的 protected

【3】私有的 private

【4】public 、protected、private 不是关键字

（3）判别对象的位置

【1】对象的内部

【2】对象的外部

【3】子类中

（4） 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

实现：私有成员前面添加两个下划线即可

class Person():

# name是公有成员

name = "刘颖"

# age是私有成员

\_\_age = 18

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

代码1：

# 私有变量案例

class Person():

# name是公有成员

name = "liuying"

# age就是私有成员

\_\_age = 18

print(Person.\_\_dict\_\_)

--- {'\_\_module\_\_': '\_\_main\_\_', 'name': 'liuying', '\_Person\_\_age': 18, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Person' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Person' objects>, '\_\_doc\_\_': None}

p = Person()

# name是公有变量

print(p.name) ----liuying

# age是私有变量

# 注意报错信息

print(p.\_Person\_\_age) ----18

print(p.age) --- 'Person' object has no attribute 'age'

代码2：

# name mangling技术

print(Person.\_\_dict\_\_)

print(p.\_Person\_\_age) ----18

p.\_Person\_\_age = 19

print(p.\_Person\_\_age) ----19