1. **类和对象的基本概念**

**第一部分**

内容的引入



有哪些共同的属性？

有哪些共同的行为？

抽象

**一、基本概念**

**1.1、类的概念**

类是一组具有相同属性和方法的所有对象的集合或者抽象，定义一个类用关键字class，其语法如下：

class ClassName:

<statement-1>

. . .

<statement-N>

其中statement既可以包含一个成员变量，也可以包含一个成员方法。

类就是一个模板，模板里可以包含多个函数，函数里实现一些功能,对象则是根据模板创建的实例，通过实例对象可以执行类中的函数

**1.1 定义一个类，只包含成员变量**

#定义一个类,类名是stu

class stu:

#定义类的成员变量

s\_name="afeng"

s\_age=20

#通过类.成员变量名访问其属性

print(stu.s\_age)

print(stu.s\_name)

结果：

20

afeng

**1.2、对象的概念**

对象是类的一个实例，创建一个类所对应的对象很简单，只要调用该类的构造方法即可，构造方法其实就是类名加上括号即可，构造方法的主要作用就是用来创建一个对象，语法如下：

对象名=类名()

对象通过对象名.成员变量名来访问其属性

比如

#定义一个类,类名是stu

class stu:

#定义类的成员变量

s\_name="afeng"

s\_age=20

#定义类的成员方法

def getAge(self):

return self.s\_age

def getName(self,name):

self.s\_name=name

return self.s\_name

#创建一个对象

s1=stu()

#通过类.成员变量名访问其属性

print(stu.s\_age)

print(stu.getAge(s1))

#通过对象.成员变量名访问其属性

s1.s\_name="czf"

print(s1.s\_name)

print(s1.getName("we23"))

print(s1.getAge())

需要注意的是，定义class的函数之后，在调用的时候出现“'takes exactly 1 argument (2 given)' Python error”。查询[Interesting 'takes exactly 1 argument (2 given)' Pythonerror](http://stackoverflow.com/questions/4909585/interesting-takes-exactly-1-argument-2-given-python-error) 之后，原来在python中，在使用类名或者对象调用其class的方法的时候，是相当与在调用中加入本身作为第一个参数的。如下：a.method(k)就相当与：a.method(a, k)，所以，在对函数进行定义的时候，需要在函数内部参数中第一个定义为self，如：

class Person():

def method(self, k):

...

如果想作为静态函数调用，可以在定义函数上方加上“@staticmethod”来进行标识。

**第二部分**

**1.3、类的初始化方法**

很多类都倾向于将对象创建为有初始状态的。因此类可能会定义一个名为 \_\_init\_\_() 的特殊方法（构造方法），像下面这样：

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

self.data = []

类定义了 \_\_init\_\_() 方法的话，类的实例化操作会自动调用 \_\_init\_\_() 方法。

#encoding=utf-8

#定义一个类,类名是stu

class stu:

#定义类的成员变量

s\_name="czf"

s\_age=20

#定义初始化方法(构造方法) ，给成员变量赋值

def \_\_init\_\_(self,name,age):

print("调用init()")

self.s\_age=age

self.s\_name=name

#定义类的成员方法

def getAge(self):

return self.s\_age

def getName(self):

return self.s\_name

#创建一个对象

s1=stu("afeng",34)

print(s1.getAge())

print(s1.getName())

执行结果：

调用init()

34

afeng

需要注意的是，self代表类的实例，而非类，证明如下：

#encoding=utf-8

#定义一个类,类名是stu

class stu:

#定义类的成员变量

s\_name="czf"

s\_age=20

#定义初始化方法(构造方法) ，给成员变量赋值

def \_\_init\_\_(self,name,age):

print(self)

print(self.\_\_class\_\_)

self.s\_age=age

self.s\_name=name

#创建一个对象

s1=stu("afeng",34)

结果：

<\_\_main\_\_.stu instance at 0x0000000002A38C88>

\_\_main\_\_.stu

从执行结果可以很明显的看出，self 代表的是类的实例，代表当前对象的地址，self.\_\_class\_\_指向类stu。

self只是一个变量名，可以使用其他变量来取代，比如"rupeng".

**1.4、类的析构函数**

\_\_del\_\_ 析构函数，释放对象时使用,调用del也会被调用。

#encoding=utf-8

#定义一个类,类名是stu

class stu:

#定义类的成员变量(公有变量，在类外可以访问)

s\_name="czf"

#定义类的私有成员变量，只有在类的内部才可以访问

\_\_s\_age=20

#定义初始化方法(构造方法) ，给成员变量赋值

def \_\_init\_\_(self,name,age):

self.\_\_s\_age=age

self.s\_name=name

#定义一个析构函数，对象被释放时，自动调用

def \_\_del\_\_(self):

print("对象被删除")

#创建一个对象

s1=stu("afeng",34)

print(s1.s\_name)

执行结果：

afeng

对象被删除

在Python中，可以说一切皆是对象。而所有的类最根本的基类则是object

class X:  
 pass  
print(X.\_\_class\_\_)  
print(X.\_\_class\_\_.\_\_base\_\_)

<class 'type'>

<class 'object'>