1.面向对象结构分析：

----面相对象整体大致分为两块区域：

--------第一部分：静态字段（静态变量）部分

--------第二部分：方法部分

--每个大区域可以分为多个小部分：

class A:

cooname = 'Jake' # 静态变量（静态字段）

\_\_cooage = 20 # 私有静态变量（私有静态字段）

def \_\_init\_\_(self, name, age): # 普通方法（构造方法）

self.name = name # 对象属性（普通字段）

self.\_\_age = age # 私有对象属性（私有普通字段）

def func1(self): # 普通方法

pass

def \_\_func(self): # 私有方法

pass

@property

def prop(self): # 属性

pass

@classmethod # 类方法

def class\_func(cls):

'''定义类方法，至少有一个cls参数'''

pass

@staticmethod # 静态方法

def stact\_func():

'''定义静态方法，无默认参数'''

pass

2.面向对象的私有与公有

对于每一个类的成员而言都有两种形式：

----共有成员，在任何地方都可以访问

----私有成员，只有在类的内部才能访问

--------私有成员和共有成员的访问限制不同：

静态字段（静态变量）

--共有静态字段：类可以访问，类内部可以访问，派生类中可以访问

--私有静态字段：仅类内部可以访问

普通字段（对象属性）

--共有普通字段：对象可以访问，类内部可以访问，派生类中可以访问

--私有普通字段：仅类内部可以访问

方法：

--共有方法：对象可以访问，类内部可以访问，派生类中可以访问

--私有方法：仅类内部可以访问

总结:

对于这些私有成员来说,他们只能在类的内部使用,不能在类的外部以及派生类中使用.

ps：非要访问私有成员的话，可以通过 对象.\_类\_\_属性名,但是绝对不允许!!!

为什么可以通过.\_类\_\_私有成员名访问呢?因为类在创建时,

如果遇到了私有成员(包括私有静态字段,私有普通字段,私有方法)

它会将其保存在内存时自动在前面加上\_类名.

3.面向对象的成员

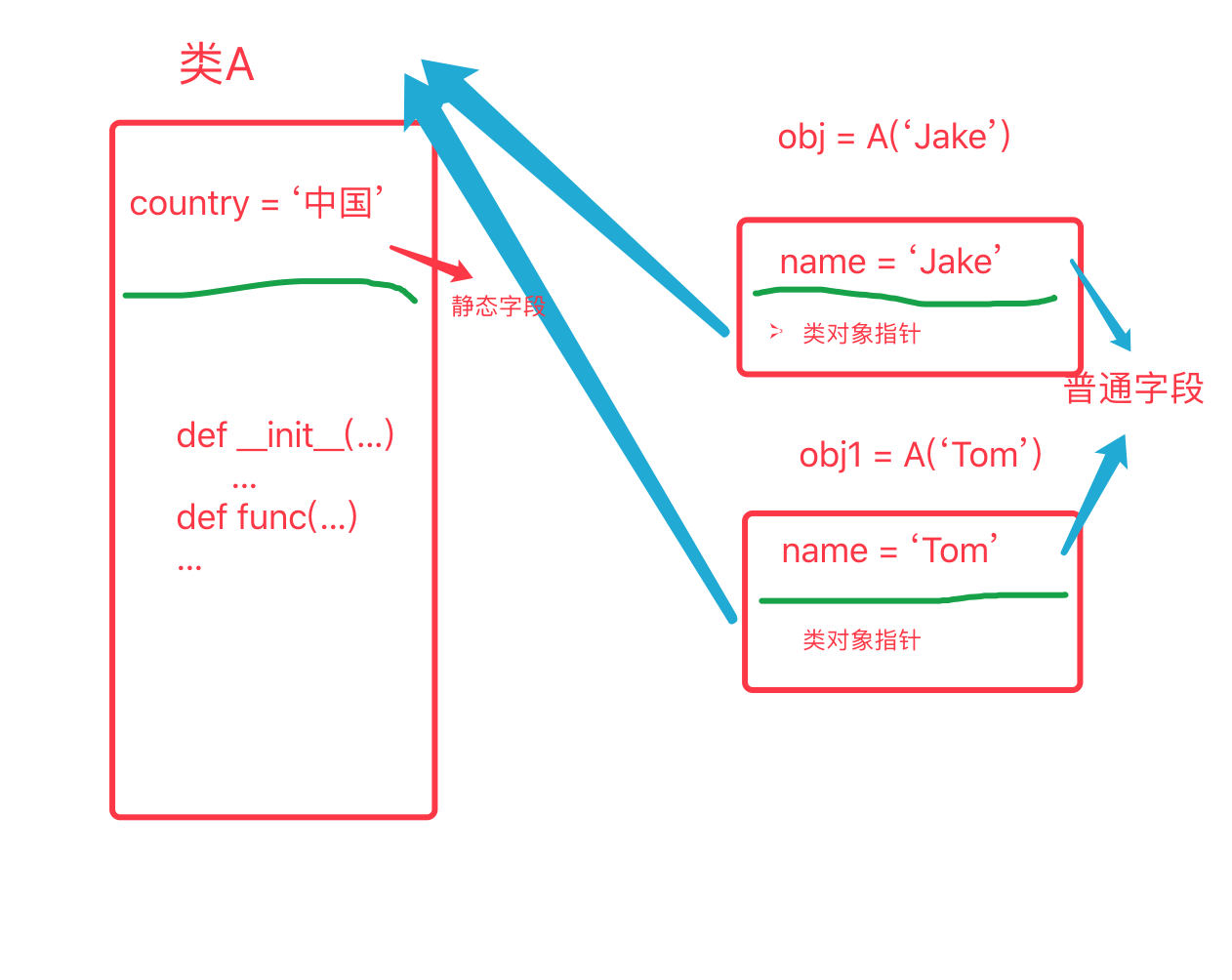
1）字段

字段包括：普通字段和静态字段，他们在定义和使用中有所区别，

而最本质的区别是内存中保存的位置不同，

--普通字段属于对象

--静态字段属于类



由上图：

静态字段在内存中只保存一份

普通字段在每个对象中都要保存一份

应用场景： 通过类创建对象时，如果每个对象都具有相同的字段，那么就使用静态字段

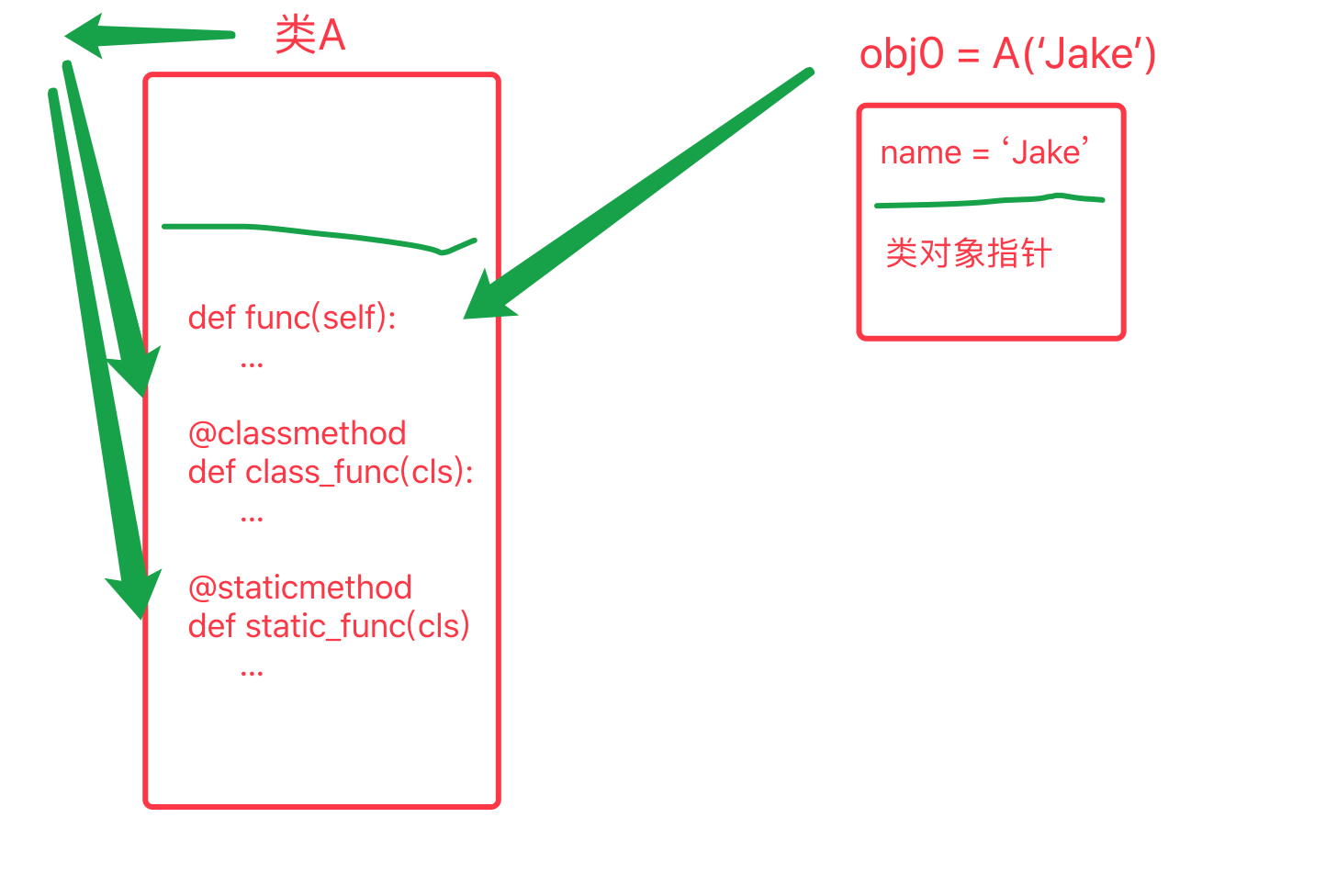
2）方法

方法包括：普通方法、静态方法和类方法，三种方法在内存中都归属于类，区别在于调用方式不同。

--普通方法：由对象调用；至少一个self参数；执行普通方法时，自动将调用该方法的对象赋值给self；

--类方法：由类调用； 至少一个cls参数；执行类方法时，自动将调用该方法的类复制给cls；

--静态方法：由类调用；无默认参数；



如上图：

相同点：对于所有的方法而言，均属于类（非对象）中，所以，在内存中也只保存一份。

不同点：方法调用者不同、调用方法时自动传入的参数不同。

4.property classmethod staticmethod

1）property #将一个方法，伪装成属性

class Bmi(object):

def \_\_init\_\_(self, name, weight, highe):

self.name = name

self.weight = weight

self.height = highe

@property

def func(self):

BMI = self.weight / (self.height \*\* 2)

return BMI

person = Bmi('Tom', 80, 1.73)

s = person.name

ss = person.func #将方法伪装成属性

print('%s的BIM指数为%s' % (s, ss))

--------------------------------------------------

class Person():

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

self.name = name

self.\_\_age = age if type(age) is int else print('重新输入')

@property #执行查询操作自动执行此操作

def age(self):

return self.\_\_age

@age.setter #执行更改操作执行此操作

def age(self, temp):

self.\_\_age = temp if type(temp) is int else print('重新输入')

@age.deleter #执行del操作自动执行此方法

def age(self):

del self.\_\_age

2）classmethod

----类方法：通过类名调用方法，类方法中第一个参数约定俗称cls，python自动将类名传给cls

class A:

def func(self): # 普通方法

print(self)

@classmethod

def func0(cls): # 类方法

print(cls)

a = A()

a.func() # <\_\_main\_\_.A object at 0x105d8a438>

A.func0() # <class '\_\_main\_\_.A'>

a1 = A()

a1.func0() # 对象调用类方法,cls 得到的是类本身.

\*\*\*\*￥类方法应用场景

1-类中一些方法不需要对象参与

2-对类中静态变量进行改变

3-继承中，父类得到子类的类空间

例子：

class A:

age = 10

@classmethod

def func(cls):

print(cls.age)

class B(A):

age = 20

B.func() # 20

3）staticmethod

由类名调用，无默认参数，主要作用是：能够使代码清晰，复用性强！