python3使用class变量

有些时候可能一些公共的函数需要写成classmethod的方式，即不用把class给先实例化之后再使用，可以直接采用《类名.方法名》的方式使用。

但是这种场景如果需要需要先调一个实例化的底层变量，然后包进当前的class以classmethod的方式进行使用，此时就需要把这个实例化的底层变量以class 变量的方式存在，不然这个实例化信息无法传递到class method那。

下面这个例子，class a总是以classmethod的方式进行function的调用，其中class a里面的方法需要用到class b，而class b必须首先要实例化之后才能使用。下面说明了2种方式来达到目的。

- class b在class a实例化时进行赋值，之后class a一直以classmethod的方式使用。
- class b的实例化在class a的一个classmethod方法中完成，class a的方法总是以classmethod的方式进行使用，在使用的角度上来看，或许此种方式看起来舒服一些，

```
class b(object):
    def __init__(self, name):
        self.name = name
    def myprint(self):
        print(f'haha, b.name = {self.name}')

class a(object):
    b_inst = None
    def __init__(self, name):
        a.b_inst = b(name)    #注意此处使用的是class 变量(a.b_inst)，即类名.变量名,在此处完成了class b的实例化到a.b_inst


    @classmethod
    def set_name(cls, name):
        a.b_inst = b(name)    #也可以使用cls.method显式地设置a.b_inst这个类变量,完成class b的实例化到class的类变量a.b_inst

    @classmethod
    def myprint(cls):
        a.b_inst.myprint()


if __name__ == '__main__':
    a_inst = a('name_b0')    #此处实例化， 仅仅只是为了给a的类变量赋值
    a.myprint()        # 调用class a的classmethod， 方法里有使用到class a的类变量。

    # or
    a.set_name('name_b1') # 可以不用把a实例化
    a.myprint()        # 调用class a的classmethod， 方法里有使用到class a的类变量。
```