第二十四次课

目标

私有属性和私有方法

继承中的私有属性和私有方法

类属性、类方法、静态方法

一 私有属性和私有方法

1.1 应用场景

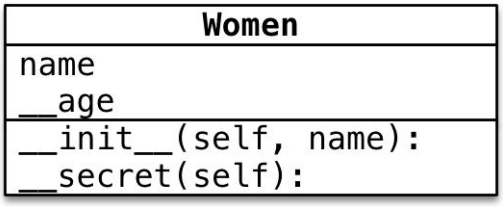
1. 在实际开发中，**对象** 的 **某些属性**或**方法** 可能**只希望** 在对象的**内部**被使用，而 **不希望在外部被访问到**

2. **私有属性** 就是 **对象** 不希望公开的 **属性**

3. **私有方法** 就是 **对象** 不希望公开的 **方法**

1.2 定义方式

在 **定义属性或方法时**，在 **属性名**或者**方法名**前 **增加 两个下划**线，定义的就是 **私有** 属性或方法




```python
class Women:

    def __init__(self, name):

        self.name = name
        # 不要问女生的年龄
        self.__age = 18

    def __secret(self):
        print("我的年龄是 %d" % self.__age)


xiaofang = Women("小芳")
# 私有属性，外部不能直接访问
# print(xiaofang.__age)

# 私有方法，外部不能直接调用
# xiaofang.__secret()
```

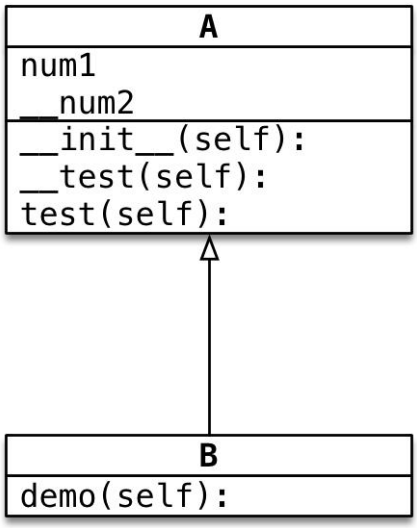
二 继承中的私有属性和私有方法

1. **子类对象 不能** 在自己的方法内部，**直接 访问 父类的 私有属性** 或 **私有方**法

2. **子类对象** 可以通过 **父类** 的 **公有方法 间接 访问**到 **私有属性** 或 **私有方法**

私有属性、方法 是对象的隐私，不对外公开，**外界** 以及 **子类** 都**不能直接访问**

私有属性、方法 通常用于做一些内部的事情



B 的对象**不能**直接**访问 __num2** 属性

B 的对象**不能**在 demo 方法内访问 **__num2** 属性

B 的对象**可以**在 demo 方法内，**调用父类的 test 方法**

父类的 test 方法内部，**能够访问** __num2 **属性**和 __test **方法**

三 类属性、类方法、静态方法

3.1 类属性

类属性 相当与**全局变量**，实例对象**共有的属性，实例对象的属性**为实例对象**自己私有**。

类属性就是**类对象**（Tool）所拥有的属性，它被所有类对象的**实例对象(实例方法)**所共有，**在内存中只存在一个副本**，这个和 C++中类的静态成员变量有点类似。对于公有的类属性，在类外可以通过**类对象**和**实例对象**访问。

类属性举例：

```
class People(object):
    name = 'Jack'  #类属性(公有)
    __age = 12     #类属性(私有)

p = People()    #创建实例对象

print(p.name)           #通过实例对象，打印类属性：name
print(People.name)      #通过类对象，打印类属性：name
print(p.__age)            #错误，不能在类外通过实例对象访问私有的类属性
print(People.__age)       #错误，不能在类外通过类对象访问私有的类属性

#结果如下：
# Jack
# Jack
# AttributeError: 'People' object has no attribute '__age'
# AttributeError: type object 'People' has no attribute '__age'
```

实例属性（对象属性）举例：

```
class People(object):
    address = '山东'  # 类属性
    def __init__(self):
        self.name = 'xiaowang'  # 实例属性
        self.age = 20  # 实例属性

p = People()    #创建实例对象
p.age = 12  # 通过实例对象调用实例属性，更改实例属性值
print(p.address)  # 通过实例对象调用类属性，并打印
print(p.name)  # 通过实例对象调用实例属性，并打印
print(p.age)  # 通过实例对象调用实例属性，并打印

#结果：
# 山东
# xiaowang
# 12

print(People.address)  # 通过类对象调用类属性，并打印
print(People.name)  # 错误（程序会报错），通过类对象调用实例属性，并打印
print(People.age)  # 错误（程序会报错），通过类对象调用实例属性，并打印

#结果：
# 山东
# AttributeError: type object 'People' has no attribute 'name'
# AttributeError: type object 'People' has no attribute 'age'
```

通过实例（对象）去修改类属性举例：

```
class People(object):
    country = 'china'  # 类属性

print(People.country)   #china
p = People()
print(p.country)    #china
p.country = 'japan'
print(p.country)  # 实例属性会屏蔽掉同名的类属性：japan
print(People.country)   #china
del p.country  # 删除实例属性
print(p.country)    #实例属性被删除后，再调用同名称的属性，会调用类属性：china
```

总结：

如果需要在**类外**修改**类属性**，必须通过**类对象**去**引用**然后进行修改。如果通过**实例对象**去引用，**会产生一个同名的实例属性**，这种方式修改的是**实例属性**，**不会影响到类属性**，并且之后如果通过实例对象去引用该名称的属性，**实例属性会强制屏蔽掉类属性**，即引用的是**实例属性**，**除非删除了该实例属性**。

3.2 类方法

类属性 就是针对 **类对象** 定义的**属性**

使用 **赋值语句** 在 **class** 关键字下方可以定义 **类属性**

类属性 用于记录 与**这个类相关** 的特征

类方法 就是针对 **类对象** 定义的**方法**

在 **类方法** 内部可以直接**访问** **类属性** 或者调用其他的 **类方法**

语法格式如下：

```
@classmethod
def 类方法名(cls):
    pass
```

类方法需要用 **修饰器** **@classmethod** 来标识，告诉解释器这是一个类方法

类方法的 第一个参数 应该是 **cls**

由 **哪一个类** **调用**的方法，方法内的 **cls** 就**是** **哪一个类的引用**

这个参数和 实例方法 的第一个参数是 self 类似

提示 **使用其他名称也可以，不过习惯使用** **cls**

通过 **类名.调用** **类方法**，调用方法时，不需要传递 cls 参数

在方法内部

可以通过 **cls.** 访问**类的属性**

也可以通过 **cls.** 调用其他的**类方法**

3.3 静态方法

在开发时，如果需要在 **类** 中封装一个方法，这个方法：

既 **不需要** **访问** **实例属性** 或者调用 **实例方法**

也 **不需要** 访问 **类属性** 或者调用 **类方法**

这个时候，可以把这个方法封装成一个 **静态方法**

语法格式如下：

```
@staticmethod
def 静态方法名():
    pass
```

静态方法 需要用 修饰器 **@staticmethod** 来**标识**，告诉解释器这是一个静态方法

通过 **类名.** 调用 **静态方法**

总结：实例方法、类方法、静态方法异同点：

	实例方法（普通方法）	类方法	静态方法
实例调用a=A()	a.normalMethod(x)	a.classMethod(x)	a.staticMethod(x)
类调用A	不能调用	A.classMethod(x)	A.staticMethod(x)

1. **定义形式上：**

a. 类方法和静态方法都是通过装饰器实现的，实例方法不是;

b. 实例方法需要传入 self 参数，类方法需要传入 cls 参数，而静态方法不需要传 self 或者 cls 参数。

2. 调用方式上：

实例方法只能通过实例对象调用；类方法和静态方法可以通过类对象或者实例对象调用，如果是使用实例对象调用的类方法或静态方法，最终都会转而通过类对象调用。

3. 应用场景：

a. 实例方法使用最多，可以直接处理实例对象的逻辑；类方法不需要创建实例对象，直接处理类对象的逻辑;静态方法将与类对象相关的某些逻辑抽离出来,不仅可以用于测试，还能便于代码后期维护。

b. 实例方法和类方法，能够改变实例对象或类对象的状态，而静态方法不能