property属性

property属性是一种用起来像是实例属性一样的特殊属性，可以对应于某个方法。

```
class Foo:
    def func(self):
        pass

    # 定义property属性
    @property
    def prop(self):
        pass

# ############### 调用 ###############
foo_obj = Foo()
foo_obj.func()  # 调用实例方法
foo_obj.prop  # 调用property属性
```

property属性的定义和调用要注意一下几点：

- 定义时，在实例方法的基础上添加 @property 装饰器；并且仅有一个self参数
- 调用时，无需括号

  ```
  方法：foo_obj.func()
  property属性：foo_obj.prop
  ```

简单的实例

> 对于京东商城中显示电脑主机的列表页面，每次请求不可能把数据库中的所有内容都显示到页面上，而是通过分页的功能局部显示，所以在向数据库中请求数据时就要显示的指定获取从第m条到第n条的所有数据 这个分页的功能包括：
>
> - 根据用户请求的当前页和总数据条数计算出 m 和 n
> - 根据m 和 n 去数据库中请求数据

```
# ############### 定义 ###############
class Pager:
    def __init__(self, current_page):
        # 用户当前请求的页码（第一页、第二页...）
        self.current_page = current_page
        # 每页默认显示10条数据
        self.per_items = 10

    @property
    def start(self):
        val = (self.current_page - 1) * self.per_items
        return val

    @property
    def end(self):
        val = self.current_page * self.per_items
        return val

# ############### 调用 ###############
p = Pager(1)
p.start  # 就是起始值，即：m
p.end  # 就是结束值，即：n
```

从上述可见

- Python的property属性的功能是：property属性内部进行一系列的逻辑计算，最终将计算结果返回。

property属性的两种方式

- 装饰器 即：在方法上应用装饰器
- 类属性 即：在类中定义值为property对象的类属性

装饰器方式

在类的实例方法上应用@property装饰器

Python中的类有经典类和新式类，新式类的属性比经典类的属性丰富。（如果类继object，那么该类是新式类）

- 经典类的实现：

```
class Goods:
  @property
  def price(self):
      return "laowang"

obj = Goods()
result = obj.price  # 自动执行 @property 修饰的 price 方法，并获取方法的返回值
print(result)
```

- 新式类的实现：

```
class Goods:
  """
  只有在python3中才有@xxx.setter  @xxx.deleter
  """
  def __init__(self):
      # 原价
      self.original_price = 100
      # 折扣
      self.discount = 0.8

  @property
  def price(self):
      new_price = self.original_price * self.discount
      return new_price

  @price.setter
  def price(self, value):
      self.original_price = value

  @price.deleter
  def price(self):
      del self.original_price
obj = Goods()
obj.price           # 获取商品价格
obj.price = 200     # 修改商品原价
del obj.price       # 删除商品原价
```


总结：

- 经典类中的属性只有一种访问方式，其对应被 @property 修饰的方法
- 新式类中的属性有三种访问方式，并分别对应了三个被@property、@方法名.setter、@方法名.deleter 修饰的方法

类属性方式

- 当使用类属性的方式创建property属性时，经典类和新式类无区别。

```
class Foo:
  def get_bar(self):
      return 'laowang'
  BAR = property(get_bar)

obj = Foo()
reuslt = obj.BAR  # 自动调用get_bar方法，并获取方法的返回值
print(reuslt)
```

property方法中有个四个参数

  - 第一个参数是方法名，调用 对象.属性 时自动触发执行方法
  - 第二个参数是方法名，调用 对象.属性 = XXX 时自动触发执行方法
  - 第三个参数是方法名，调用 del 对象.属性 时自动触发执行方法
  - 第四个参数是字符串，调用 对象.属性.**doc** ，此参数是该属性的描述信息

```
class Foo(object):
    def get_bar(self):
        print("getter...")
        return 'laowang'

    def set_bar(self, value):
        """必须两个参数"""
        print("setter...")
        return 'set value' + value

    def del_bar(self):
        print("deleter...")
        return 'laowang'

    BAR = property(get_bar, set_bar, del_bar, "description...")

obj = Foo()

obj.BAR  # 自动调用第一个参数中定义的方法：get_bar
obj.BAR = "alex"  # 自动调用第二个参数中定义的方法：set_bar方法，并将“alex”当作参数传入
desc = Foo.BAR.__doc__  # 自动获取第四个参数中设置的值：description...
print(desc)
del obj.BAR  # 自动调用第三个参数中定义的方法：del_bar方法
```


总结：

- 定义property属性共有两种方式，分别是【装饰器】和【类属性】，而【装饰器】方式针对经典类和新

式类又有所不同。

- 通过使用property属性，能够简化调用者在获取数据的流程。