一 添加和获取对象属性

对象属性可以在类里面添加，也可以在类外面添加

1.1 类外面添加对象属性

- 语法:
  - 对象名.属性名 = 值

```
# 洗衣服
class Washer:
    def wash(self): # 实例方法  也叫做 对象方法
        print(self)  # self 指的就是调用该方法的实例对象
        print('洗衣服====')

# 创建对象
xiaotiane = Washer()

# 验证功能    ------  对象名.实例方法名()
xiaotiane.wash()

# 添加属性
xiaotiane.height = 1000   # 高
xiaotiane.width = 500   # 宽


# 获取属性
print(xiaotiane.height)
print(xiaotiane.width)
```

1.2 类里面获取属性

```
# 洗衣服
class Washer:
    def wash(self): # 实例方法  也叫做 对象方法
        print(self)  # self 指的就是调用该方法的实例对象
        print('洗衣服====')

    def print_info(self):
        """添加属性"""
        print(self.height)
        print(self.width)


# 创建对象
xiaotiane2 = Washer()

# 验证功能    ------  对象名.实例方法名()
xiaotiane2.wash()

# 添加属性
```

```
xiaotiane2.height = 1000    # 高
xiaotiane2.width = 500    # 宽


# 获取属性
xiaotiane2.print_info()
```

二 魔法方法

在python中，--xxx--()的方法就叫魔法方法，指的是具有特殊功能的方法

2.1 --init--方法

```
# 洗衣服
class Washer:
    def __init__(self):  # 初始化魔法方法     一般就是去绑定属性的
        """添加属性"""
        print('这是初始化魔法方法')
        self.height = 1000
        self.width = 10

    def wash(self): # 实例方法   也叫做 对象方法
        print(self)  # self 指的就是调用该方法的实例对象
        print('洗衣服====')

    def print_info(self, name):  # name 属于第一个参数
        """获取属性"""
        print(self.height)
        print(self.width)
        print(name)


# 创建对象
xiaotiane2 = Washer()
xiaotiane2.print_info('xiaohuozi')
```

注意：在创建对象时候就会被调用，不需要手动调用，self参数不需要传递，python解释器会自动把当前对象的引用传过去

2.1.1带参数的init魔法方法

思考：一个类中有多个对象， 如何对多个对象设置不同的属性值

传参

```
# 洗衣服
class Washer:
    def __init__(self, height, width):  # 初始化魔法方法
        """添加属性"""
        self.height = height   # height(属性名/变量名) = height(形参)
        self.width = width

    def print_info(self):
        """获取属性"""
```

```
        print(self.height)
        print(self.width)



# 创建对象
xiaotiane = Washer(1000, 100)
xiaotiane.print_info()
# print(xiaotiane)

xiaotiane1 = Washer(10,12)
xiaotiane1.print_info()
```

2.2 str魔法方法

当print输出一个对象的时候，默认打印的是对象的内存空间地址，如果在类中定义了str魔法方法之后，那么此时在打印对象的时候就是打印的这个方法的返回值

用来返回类中的状态，必须要有return

```
# 洗衣服
class Washer:
    def __init__(self, height, width):  # 初始化魔法方法
        """添加属性"""
        self.height = height   # height(属性名/变量名) = height(形参)
        self.width = width

    def __str__(self):  # 类的使用说明或者类的状态
        return '这是洗衣机的使用说明'

    def print_info(self):
        """获取属性"""
        print(self.height)
        print(self.width)



# 创建对象
xiaotiane = Washer(1000, 100)
# xiaotiane.print_info()
print(xiaotiane)
```

2.3 del魔法方法

当删除一个对象的时候，python解释器会自动调用del魔法方法

```
# 洗衣服
class Washer:
    def __init__(self, height, width):  # 初始化魔法方法
        """添加属性"""
        self.height = height   # height(属性名/变量名) = height(形参)
        self.width = width

    def print_info(self):
```

```
        """获取属性"""
        print(self.height)
        print(self.width)



xiaotiane = Washer(10000,12)
print(id(xiaotiane.height))


xiaotiane1 = Washer(1321,520)
print(id(xiaotiane1.height))









# 洗衣服
class Washer:
    def __init__(self, height, width):  # 初始化魔法方法
        """添加属性"""
        self.height = height   # height(属性名/变量名) = height(形参)
        self.width = width

    def print_info(self):  # 实例方法
        """获取属性"""
        print(self.height)
        print(self.width)



xiaotiane = Washer(10000,12)
print(id(xiaotiane.height))
print(id(xiaotiane.print_info()))

xiaotiane1 = Washer(1321,520)
print(id(xiaotiane1.height))
print(id(xiaotiane1.print_info()))
```

注意:

- 如果一个类有多个对象，每个对象的属性是各自独立保存的，都是独立的地址
- 但是实例方法是所有对象所共有的，只占用一份内存空间，类会通过self参数来判断是哪一个对象调用了该实例方法

三 综合案例

3.1 烤牛肉串

3.1.1 需求

1.被烤的时间和牛肉串的状态

0-2分钟: 生的

2-5分钟：半生

5-10：熟了

十分钟以上：烤糊了

2.添加作料

可以根据用户的喜好添加作料

3.1.2 步骤分析

涉及到的事物： 牛肉串 抽象出来一个类

3.1.3 实现代码

3.1.3.1 定义类

- 牛肉的属性
  - 牛肉的状态
  - 被烤的时间
  - 添加的作料
- 牛肉的方法
  - 被烤
    - 用户指定烤的时间
    - 查看牛肉的状态，并且随着时间改变牛肉的状态
  - 添加作料
    - 用户想添加什么就添加什么
    - 保存用户添加的作料
- 显示牛肉的状态

```
class Beef():
    def __init__(self):
        # - 牛肉的状态
        self.cook_statr = '生的'
        # - 被烤的时间
        self.cook_time = 0
        # - 添加的作料
        self.condiments = []

    def cook(self, time):
        """烤牛肉"""
        self.cook_time += time
        if 0 <= self.cook_time <= 2:
            self.cook_statr = '生的'
        elif 2 < self.cook_time <= 5:
            self.cook_statr = '半生'
        elif 5 < self.cook_time <= 10:
```

```
            self.cook_statr = '熟了'
        elif self.cook_time > 10:
            self.cook_statr = '烤糊了'

    def add_condiments(self, condiment):
        """添加作料"""
        self.condiments.append(condiment)

    def __str__(self):
        return f'这个牛肉串被烤的总时间为{self.cook_time},状态是{self.cook_statr},添加的作料有{self.condiments}'


# 创建对象
niurou = Beef()
niurou.cook(1)
niurou.add_condiments('孜然')
print(niurou)
niurou.cook(3)
niurou.add_condiments('辣椒粉')
print(niurou)
niurou.cook(5)
print(niurou)
niurou.add_condiments('辣椒粉')
print(niurou)
```

对象名.方法名

对象名.属性名