```python
#201014_面向对象进阶
class Rectangle:  #新建一个长方形的类
    def __init__(self,length,width):  #初始化方法
        self.length=length
        self.width=width
    def perimeter(self):  #计算周长的方法
        return (self.length+self.width)*2
    def area(self):
        return self.length*self.width
    @classmethod  #装饰器,表示下面的方法是类方法
    def features(cls):
        print('两边的长相等,两边的宽也相等,长和宽的角度为90°')
    @staticmethod  #装饰器,表示下面的方法是静态方法
    def fun1(a,b):
        print(a+b)
# rec=Rectangle(8,6)
# print(rec.__dict__)  #查看实例的属性,值为字典形式
# rec.features()  #类和实例都可以直接调用类方法

#类当中的方法,分为实例方法,类方法,静态方法
# Rectangle.features()  #类方法可以直接被类调用
# Rectangle.area()  #实例方法不能直接被类调用,必须先实例化之后,由实例进行调用

#静态方法既可以由类调用,也可以由实例调用
# rec.fun1(3,6)
# Rectangle.fun1(90,180)

#用type()函数判断对象的类型
# print(type(rec.fun1))  #静态方法是function
# print(type(rec.area))  #实例方法是method
# print(type(rec.features))  #类方法是method
#inspect模块
# import inspect  #python的自检模块,可以判断一个对象是否是某种类型
# print(inspect.ismethod(rec.features))  #判断某个对象是否是方法,返回值为布尔型
# print(inspect.ismethod(rec.area))  #实例方法和类方法都是方法
# print(inspect.ismethod(rec.fun1))  #静态方法不是方法
# print(inspect.isfunction(rec.fun1))  #判断某个对象是否是函数
# print(inspect.isclass(Rectangle))  #判断某个对象是否是一个类

#继承
#完全继承
# class Square(Rectangle):
#     pass
# squ=Square(6,6)
# print(squ.perimeter())
# print(squ.area())
# squ.features()
# squ.fun1(33,66)

#重载,对需要的方法进行重写
# class Square(Rectangle):
#     def __init__(self,side):
#         self.length=side
#         self.width=side
# squ=Square(6)
```

```
# print(squ.perimeter())
# print(squ.area())

#所有方法都重写的话,其实就没有必要继承
# class Square(Rectangle):
#     def __init__(self,side):
#         self.side=side
#     def perimeter(self):
#         return self.side*4
#     def area(self):
#         return self.side**2

#父类方法的扩展,有时对于某个方法,想继承一部分,但是又不想彻底重写,只想增补一些内容
# class Square(Rectangle):
#     @classmethod
#     def features(cls):
#         # super().features()  #保留父类方法中的内容
#         Rectangle.features()  #与上一行的作用是一样的,任意选一种即可
#         print('四个边都相等')
# Square.features()
```