```
# 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): #求周长的方法
#         return (self.length+self.width)*2
#     def area(self): #求面积的方法
#         # self.__fun1()
#         return self.length*self.width
#     def __fun1(self): #私有方法
#         print('长方形')
#     @classmethod #装饰器，声明下面的方法是类方法，而不是实例方法
#     def features(cls):
#         print('两边的长相等，两边的宽也相等，长和宽始终为90°')
#
#     @staticmethod #声明下面的方法是静态方法
#     def sumdata():
#         print('这是一个静态方法')
#实例方法，只有类的实例才可以使用，类不能直接使用，必须实例化之后，由实例来调用
# rec=Rectangle(6,3)
# print(rec.perimeter())
# print(rec.area())
# rec.features()
#如果直接调用rec.__fun1()，则报错，因为是私有方法。但是__fun1()可以在类的内部被正常调用

#类方法，既可以由实例来调用，也可以由类来调用，也就是说不需要实例化的情况下就可以调用它
# Rectangle.features()
# rec=Rectangle(6,3)
# rec.features()

#静态方法，实际上就是函数，只不过它是存放在了类的里面，但它和类本身没有太多的联系,类和实例都可以调用它
# Rectangle.sumdata()
# rec=Rectangle(5,4)
# rec.sumdata()
import inspect #python的自检模块，它可以判断一个对象是否是某种类型
# print(inspect.ismethod(rec.features)) #inspect.ismethod 判断某个对象的类型是否是方法
# print(type(rec.features)) #此处也可以用type进行判断
# print(inspect.ismethod(rec.sumdata))
# print(type(rec.sumdata))
# print(inspect.isfunction(rec.sumdata)) #判断某个对象的类型是否是函数

#继承
#写一个正方形的类，这个类继承长方形的类
# class Square(Rectangle):
#     pass
# squ=Square(6,6)
# print(squ.perimeter())
# print(squ.area())

#方法的重写，与扩展
# class Square(Rectangle):
#     def __init__(self,side):#子类可以重新写新的初始化方法
#         self.length=side
```

```
#         self.width=side
#     def features(cls): #方法的扩展
#         super().features() #加上这句话，可以继承父类的方法，同时后面也可以扩展子类自己的
代码
#         # super(Square,cls).features() #这种方式也可以
#         print('长和宽也相等')
# squ=Square(6)
# print(squ.perimeter())
# print(squ.area())
# squ.features()
# squ.__fun1()

#私有，如果不想让子类继承某个方法或者属性，可以在方法或属性的前面加__

#Python当中，有一个类，叫object,所有的类都是它的子类,无论有没有主动继承object,实际上都继承了
# class Cls1: #隐式继承
#     '''
#     注释
#     '''

# class Cls2(object): #显式继承
#     pass

# print(dir(Cls1))
# print(dir(Cls2))
# print(Cls1.__doc__) #显示类的注释
# print(Cls1.__name__) #显示类的名字
# print(Cls1.__module__) #显示类属于哪个模块
# print(Square.__bases__) #显示类的父类的名字
# print(Square.__dict__) #显示类的属性

#isinstance 判断某一个实例是否属于某一个类
# print(isinstance(squ,Square))
# a=100.11
# print(isinstance(a,int))

#多继承 python中，可以同时继承多个父类，每个父类之间用英文的逗号隔开。
#如果多个父类中有同名的方法，则按继承的顺序，优先使用第一个继承的类
# class Class2:
#     def money(self):
#         print('这里有200万')
# class Class1:
#     def money(self):
#         print('这里有100万')
# class Class3(Class2,Class1):
#     pass
# cls3=Class3()
# cls3.money()
#多态
# class Animal:
#     def say(self):
#         print('Animal')
# class Dog(Animal):
#     def say(self):
#         print('汪汪汪')
# class Cat(Animal):
#     def say(self):
#         print('喵喵喵')
```

```
#
# def animal_say(animal):
#     animal.say()
#
# dog=Dog()
# cat=Cat()
# animal_say(dog)
# animal_say(cat)

# class Fanguan:
#     pass
# class Yuxiangrousi(Fanguan):
#     def caidan(self):
#         print('鱼香肉丝')
# class Gongbaojiding(Fanguan):
#     def caidan(self):
#         print('宫保鸡丁')
# class Qingjiaotudousi(Fanguan):
#     def caidan(self):
#         print('青椒土豆丝')
#
# def fuwuyuan(obj):
#     obj.caidan()
#
# guke1=Yuxiangrousi()
# guke2=Gongbaojiding()
# guke3=Qingjiaotudousi()
#
# fuwuyuan(guke1)
# fuwuyuan(guke2)
# fuwuyuan(guke3)

#上节课思考题
# print(len([i for i in range (1,10000) if '3' in str(i)]))

#思考题
#写一个函数，可以求某数的阶乘
```