面向对象特性

- 封装

1. 对象都有明确的边界，把属性保护在边界之内，内部的变化对外部的变化没有影响
2. 封装的粒度：
    粒度过大：导致对象过于复杂，不利于各司其职
    粒度过小：造成对象过于简单，让过程过于复杂

- 继承

1. 类和类之间的关系
    动物类---马类，猫类，鸟，鱼类
    父类-----子类
2. 继承：一定要满足 子类 is a 父类
3. 父类是子类的共性的抽象
4. 父类---子类 ： 一般---特殊
5. 语法：
    class 子类名（父类名）：
    父类：基类，超类
    object：是所有类的父类---根类

- 继承的语法规则

1. 父类拥有的成员，子类可以继承
2. 子类不能继承父类的私有成员
3. Python中的继承是多继承
4. 继承具有可扩展性
    父类扩展了子类
5. 如果子类没有创建任何初始化方法，则调用父类的初始化方法
    调用父类的初始化方法：
        1. 父类类名.__init__(self)
        2. super().__init__()
6. 子类的修改，不会影响其他子类

```
class Father:
    __money=1000000
    def football(self):
        print('football play good')
    def __init__(self):
        print('this is your father')

class AfterFather:#干爹
    money=100000
    def __init__(self):
        print('this is your afterfather')

class Children(Father,AfterFather):
    def __init__(self):  # self=c
        print('this is your son')
        # Father.__init__(self)
        # AfterFather.__init__(self)
```

```
        super().__init__()

c=Children()  # c
print(c.money)
c.football()
```

- 方法覆盖

1. 父类中拥有的方法（非私有），子类必定拥有，子类也实现了同名的方法，此时创建对象后，调用该方法，只会调用子类的方法（遮蔽）
2. 子类的特殊实现，遮蔽了父类的一般实现

```
class Animal:
    def eat(self):
        print('animal can eat')

class Cat(Animal):
    def eat(self):
        print('cat can eat fish')

class Dog(Animal):
    def eat(self):
        print('dog can eat bone')

dog=Dog()
cat=Cat()
```

```
dog.eat()
cat.eat()
```

- 接口

```
1. 接口就是标准
2. 父类中的方法和属性就是一种标准（接口）
3. 作用：指导，规定所有继承于该父类的子类，应该有的属性和方
4. 接口回调：
        某个标准还尚未完成时，已经可以调用该标准（需要一个预先定好的接口名）
```

```
def fun(n):  # 先使用了hehe 属性
    print(n.hehe) # n 应该有hehe属性---标准（接口）



class A:
    hehe='alksjdlaksj'

a=A()
fun(a)
```

- 多态