Mixin编程

1. 和多继承类似
2. 是一种开发模式
   将多个类中功能进行组成使用

- 利用多继承实现Mixin编程

```
class A:
    def a(self):
        print('a')

class B:
    def b(self):
        print('b')

class C(A,B):
    pass

c=C()
c.a()

c.b()
```

- 利用__bases__()

```
多继承会修改源代码，如果需要动态的给子类添加一个父类则需要使用__bases__()
class A:
    def a(self):
        print('a')

class B:
    def b(self):
        print('b')

class D:
    def d(self):
        print('d')

class C(B):
    pass

c=C()
C.__bases__+=(A,D)

c.d()
```

- 利用插件模式

```
1. 以上两种形式都使用了继承关系
2. __dict__()
        展示对象的属性，可以进行增加删除修改操作---实例属性
        class A:
            age=15

            def a(self):
                print('111111111111')

            def __init__(self):
                self.name='hehe'
                self.sex='男'


        aa=A()
        print(aa.__dict__)
        aa.__dict__['c']=100

        del aa.__dict__['name']
        print(aa.__dict__)
```

```
# 利用插件完成给其他类添加功能

class Plugin:  #是一种工具
    def __init__(self):
        self.methods=[]  #存储方法对象

    def plugin(self,owner):  #向owner对象中添加methods中的方法
        for i in self.methods:
            #i 就是每个方法对象
            owner.__dict__[i.__name__]=i

    def plugout(self,owner):
        for i in self.methods:
            del owner.__dict__[i.__name__]

class A(Plugin):   #A他是一个功能
    def __init__(self):

        super().__init__()  # 创建父类的methods列表

        self.methods.append(self.a)

    def a(self):
        print('a')

class B(Plugin):   #A他是一个功能
    def __init__(self):

        super().__init__()  # 创建父类的methods列表
```

```
        self.methods.append(self.b)

    def b(self):
        print('b')


class C:
    pass

c=C()
A().plugin(c)
B().plugin(c)

c.a()

c.b()
del c.a
```