变量的赋值操作：

形成两个变量，引用一个对象

浅拷贝：

源对象和引用对象会引用同一个对象

深拷贝：

使用copy.deepcopy（）函数，源对象和拷贝对象的子对象不同

【实例】直接赋值

```
class Person:
    def __init__(self):
        pass
    def say(self):
        pass
b = Person
a = b
def say() -> object:
    print("good")
a.say = say()
b.say()
```

【测试】深拷贝和浅拷贝

```
import copy

class Computer:
    def __init__(self, cpu, gpu):
        if not isinstance(cpu, Cpu) or not isinstance(gpu, Gpu):
            return
        self.cpu = cpu
        self.gpu = gpu

class Cpu:
    def __init__(self, cpu_type):
        self.type = cpu_type
    def cpuType(self):
        print(self.type)

class Gpu:
    def __init__(self, gpu_type):
        self.type = gpu_type
    def gpuType(self):
        print(self.type)

intel_i9_10000 = Cpu("intel_i7_9750")
nvidia_RTX3060 = Gpu('nvidia_RTX3060')
computer1 = Computer(intel_i9_10000, nvidia_RTX3060)

# 测试浅拷贝
computer2 = copy.copy(computer1)
print(id(computer1) == id(computer2))          # False
print(id(computer1.cpu) == id(computer2.cpu))   # True
# 结果发现，两个不同的电脑用的是同一块cpu和gpu

# 测试深拷贝
computer3 = copy.deepcopy(computer1)
print(id(computer1) == id(computer3))          # False
print(id(computer1.cpu) == id(computer3.cpu))   # False
```