第一次上机：

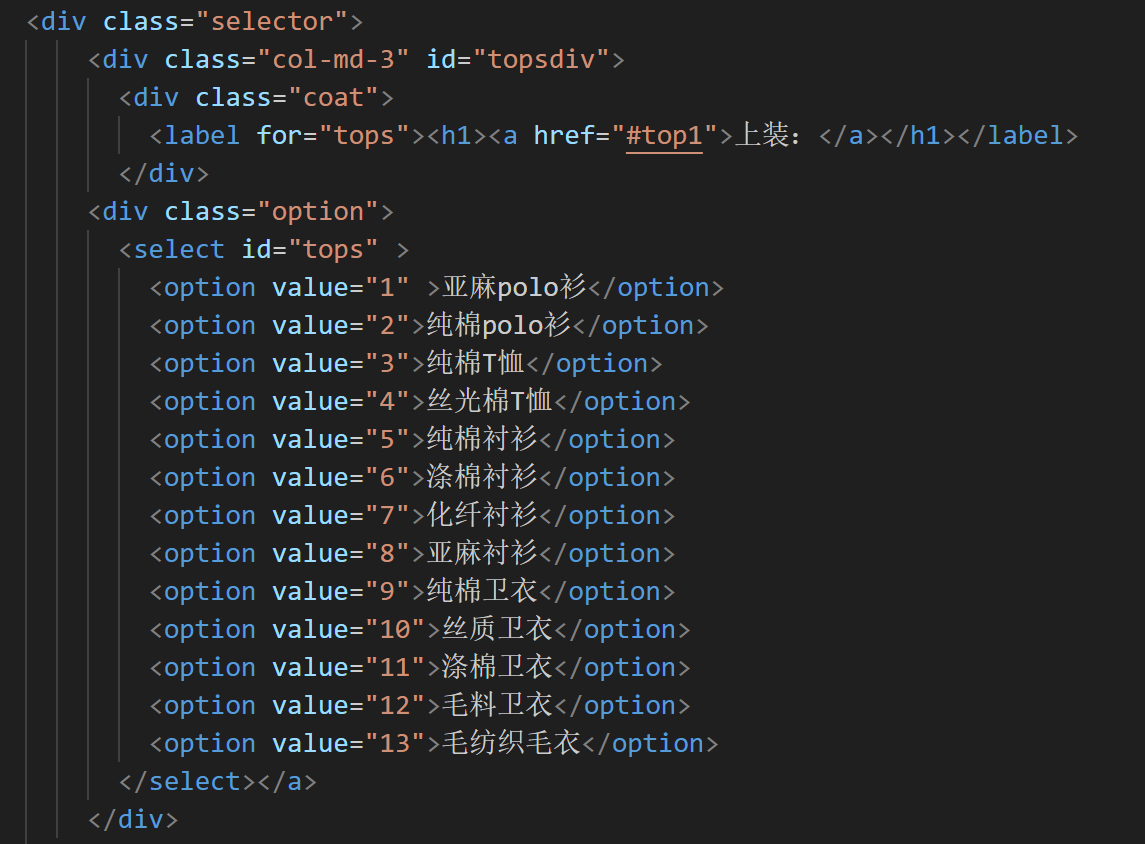
1.坏味道的代码



什么坏味道：神秘命名（Mysterious Name）

重构方法：变量改名

重构后的代码：



来源：节能减排大赛网页制作的时候合伙人写的逆天代码，不知道为什么有这么多row

2.坏味道的代码



什么坏味道：重复代码（Duplicated Code）

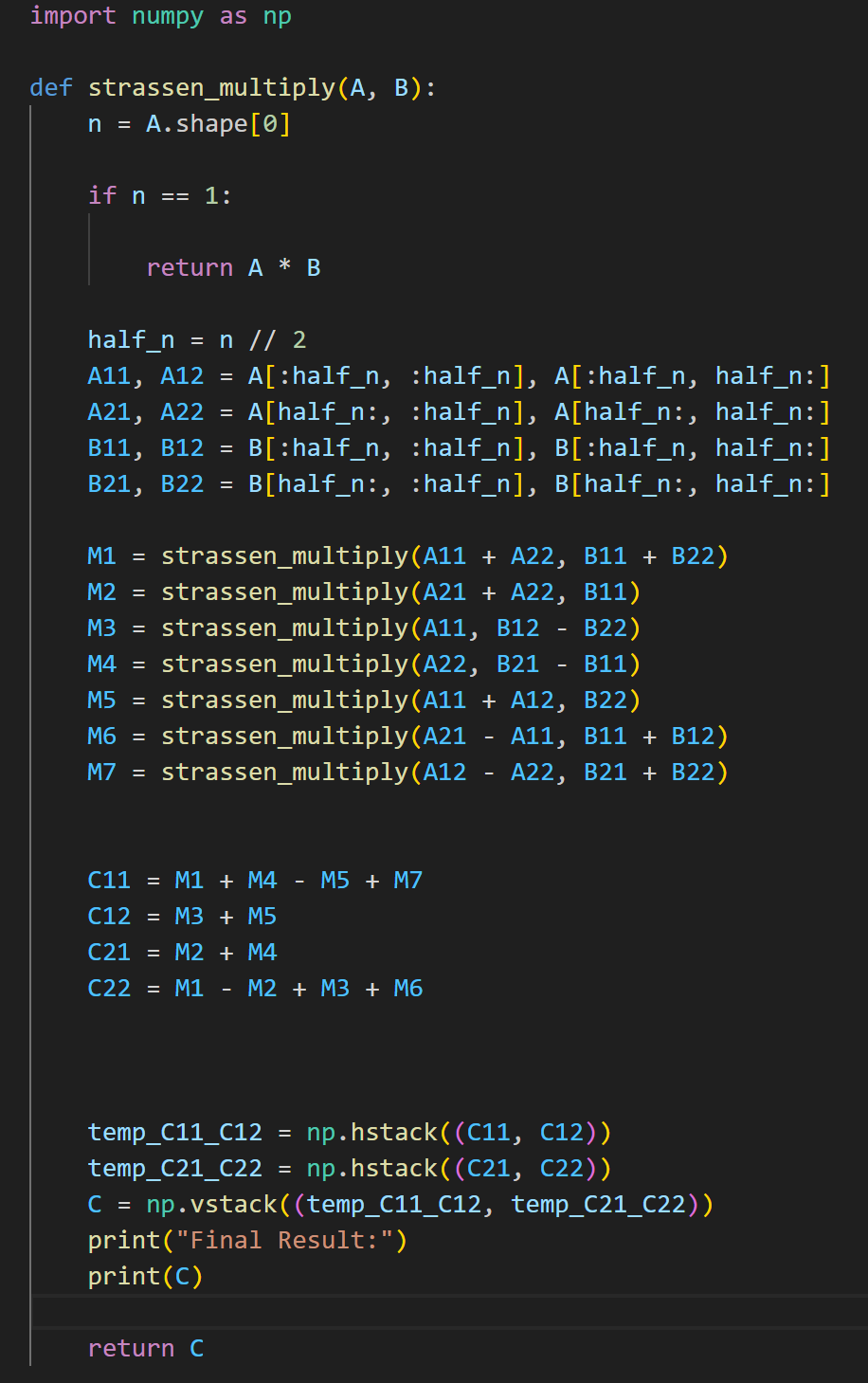
重构方法：提炼函数

重构后的代码：



来源：我的个人网页的主页css

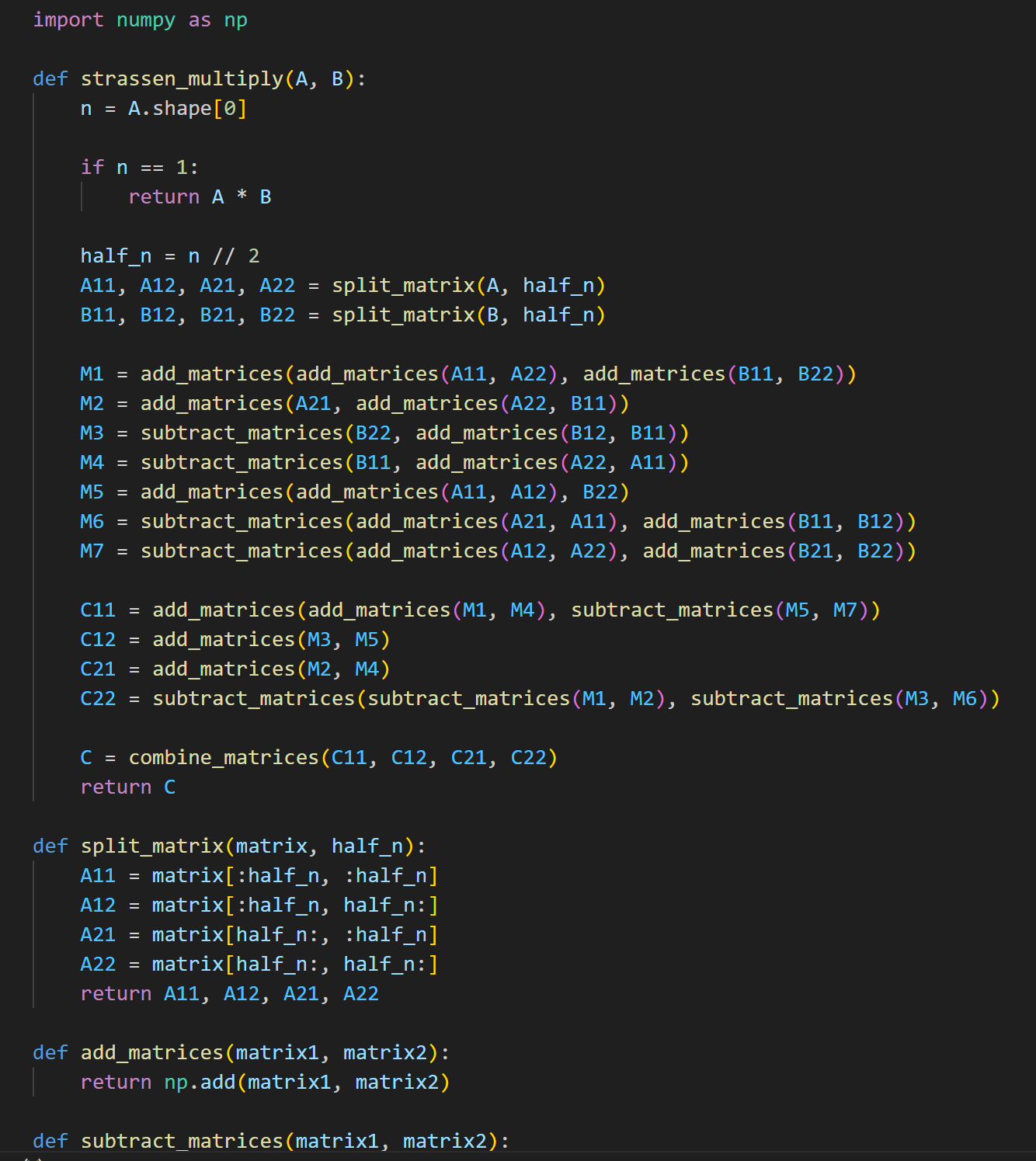
3-4.坏味道的代码



什么坏味道：冗赘的元素（Lazy Element），过长函数（Long Function）

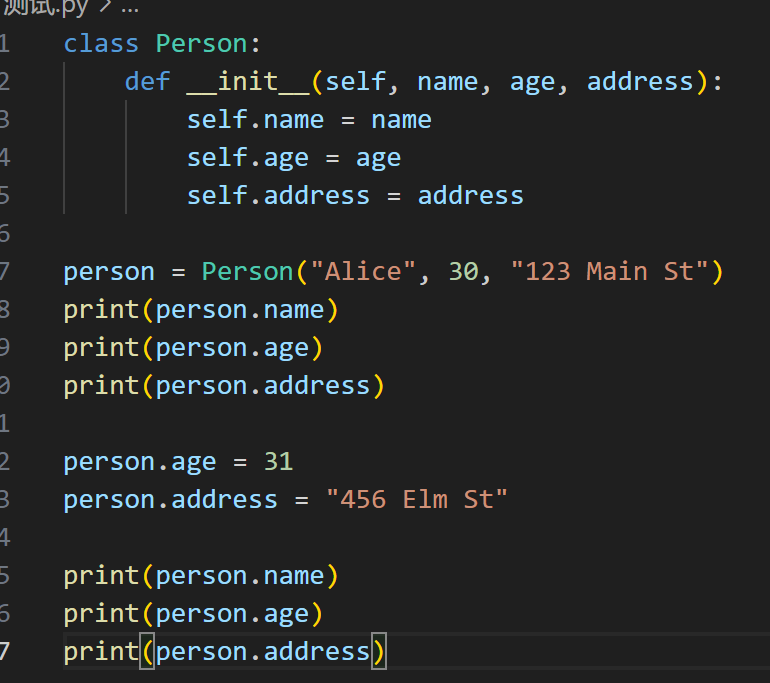
重构方法：减少不必要的变量，提炼函数

重构后的代码：



来源：算法课设

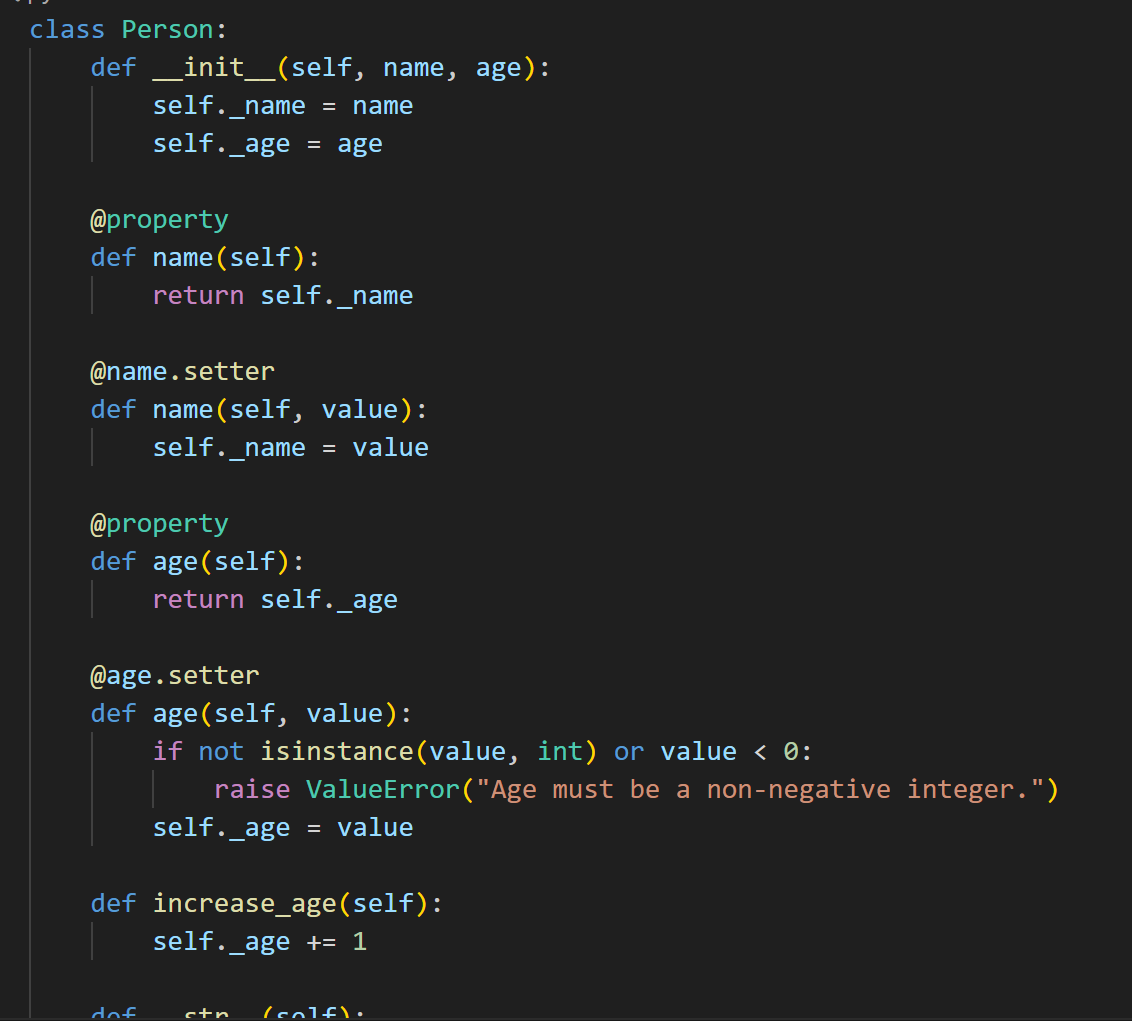
1. 坏味道的代码：



什么坏味道：纯数据类（Data Class）

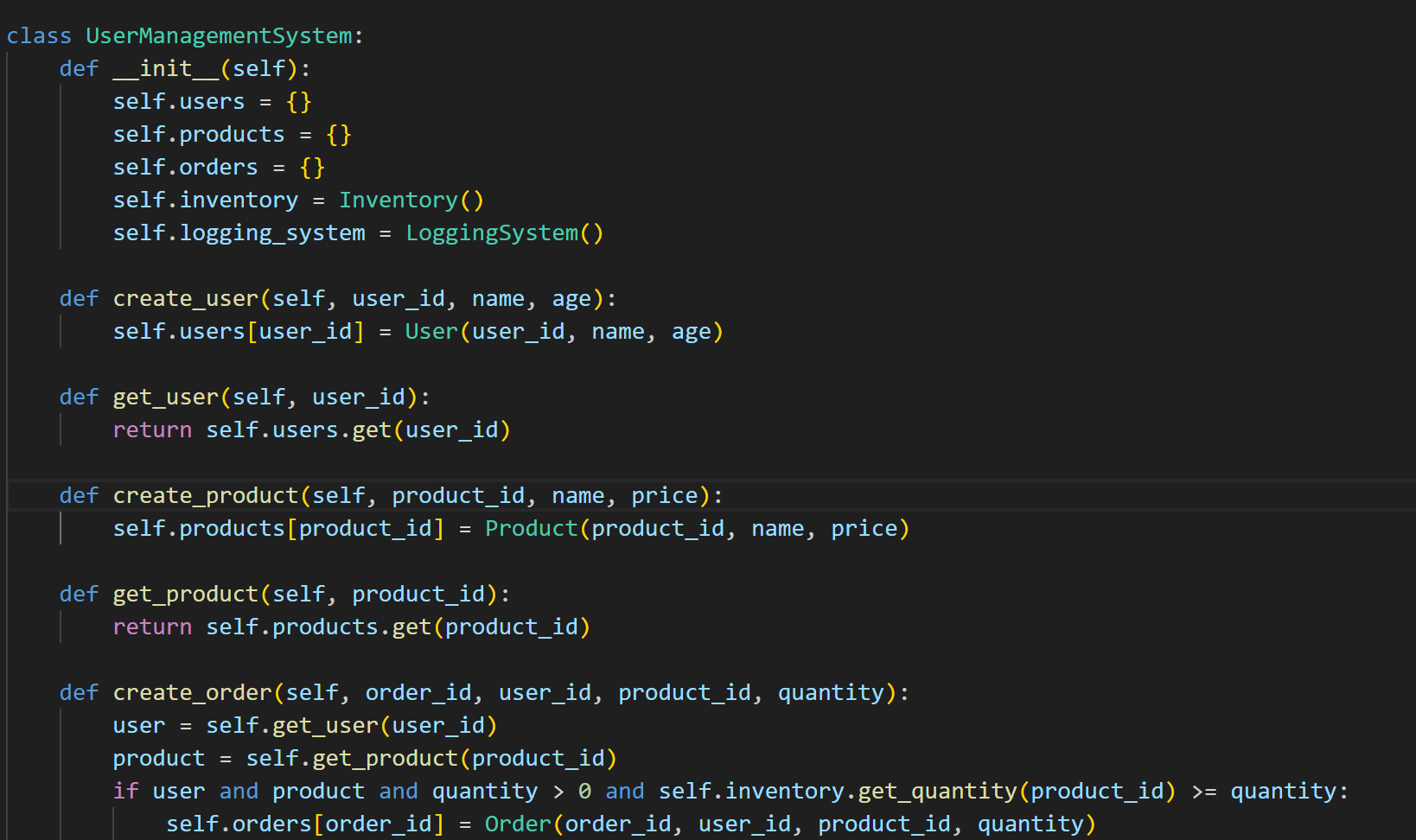
重构方法：封装记录

重构后的代码：



来源：python练习

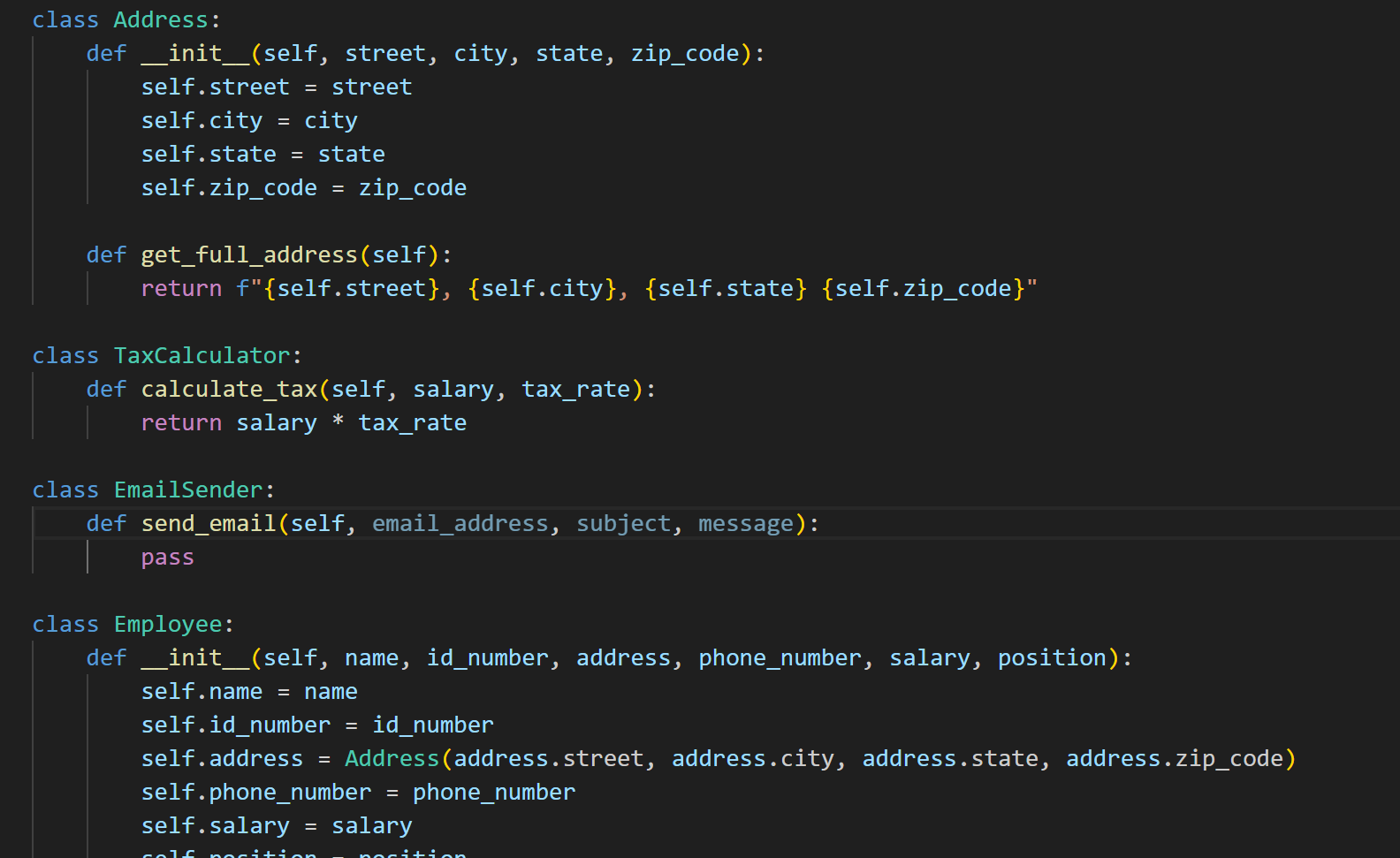
1. 坏味道的代码：



什么坏味道：过大的类（Large Class）

重构方法：提炼类

重构后的代码：



来源：python作业

第二次上机：

def iptest(ip):

    parts = ip.split('.')

    if len(parts) != 4:

        return False

    for part in parts:

        if not part.isdigit():

            return False

        i = int(part)

        if i < 0 or i > 255:

            return False

        if len(part) > 1 and part[0] == '0':

            return False

    return True

import unittest

class TestIPValidator(unittest.TestCase):

    def test\_valid\_ipv4\_address(self):

        # 测试有效的IPv4地址

        self.assertTrue(iptest("192.168.1.1"))

        self.assertTrue(iptest("1.2.3.4"))

        self.assertTrue(iptest("0.0.0.0"))  # 允许全零地址

        self.assertTrue(iptest("255.0.0.0"))

        self.assertTrue(iptest("0.255.0.0"))

        self.assertTrue(iptest("0.0.255.0"))

        self.assertTrue(iptest("0.0.0.255"))

    def test\_invalid\_ipv4\_address(self):

        # 测试无效的IPv4地址，包含前导零

        self.assertFalse(iptest("01.2.3.4"))  # 第一部分有前导零

        self.assertFalse(iptest("1.02.3.4"))  # 第二部分有前导零

        self.assertFalse(iptest("1.2.03.4"))  # 第三部分有前导零

        self.assertFalse(iptest("1.2.3.04"))  # 第四部分有前导零

        self.assertFalse(iptest("256.0.0.0"))  # 第一部分大于255

        self.assertFalse(iptest("0.265.0.0"))  # 第二部分大于255

        self.assertFalse(iptest("0.0.355.0"))  # 第三部分大于255

        self.assertFalse(iptest("0.0.0.355"))  # 第四部分大于255

        self.assertFalse(iptest("a.b.c.d"))   # 非数字字符

        self.assertFalse(iptest("1..3.4"))    # 缺失部分

        self.assertFalse(iptest("1.2..4"))    # 缺失部分

        self.assertFalse(iptest("1.2.3."))    # 缺失最后一部分

        self.assertFalse(iptest(""))          # 空字符串

        self.assertFalse(iptest("1.2.3"))     # 缺少最后一部分

        self.assertFalse(iptest("::1"))       # IPv6地址

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

