作业三的商品管理系统代码如下：

import json

class Good:

    # 编号pid  品名：pname,价格：price,数量：num,小计:acount,出厂日期：pdate

    def \_\_init\_\_(self,pid,pname,price,num,pdate):

        self.pid = pid

        self.pname = pname

        self.price = price

        self.num = num

        self.pdate = pdate

        self.acount = price \* num

    def \_\_str\_\_(self):

        return (

f"""编号: {self.pid}

品名: {self.pname}

价格: {self.price}

数量: {self.num}

小计: {self.acount}

出厂日期: {self.pdate}"""

        )

    def \_\_repr\_\_(self):

        return self.\_\_str\_\_()

    def \_\_eq\_\_(self,other):

        return self.pid == other.pid

    def \_\_lt\_\_(self,other):

        return self.price < other.price

    def \_\_gt\_\_(self,other):

        return self.price > other.price

    def \_\_le\_\_(self,other):

        return self.price <= other.price

    def \_\_ge\_\_(self,other):

        return self.price >= other.price

    def \_\_ne\_\_(self,other):

        return self.price != other.price

class GoodManager:

    def \_\_init\_\_(self,\*args):

        # 商品列表(list)

        self.goods = []

        if args:

            for good in args:

                self.goods.append(good)

    def add(self,good):

        self.goods.append(good)

    def remove(self,good):

        self.goods.remove(good)

    def \_\_str\_\_(self):

        return "\n".join([str(good) for good in self.goods])

    def \_\_repr\_\_(self):

        return self.\_\_str\_\_()

    def sort(self):

        self.goods.sort(key=lambda good:good.pid)

    def sort\_by\_price(self):

        self.goods.sort(key=lambda good:good.price)

    def sort\_by\_num(self):

        self.goods.sort(key=lambda good:good.num)

    def sort\_by\_pname(self):

        self.goods.sort(key=lambda good:good.pname)

    def search(self,pid):

        for good in self.goods:

            if good.pid == pid:

                return good

        return None

    def update(self,good):

        for i in range(len(self.goods)):

            if self.goods[i] == good:

                self.goods[i] = good

                break

    def get(self):

        return self.goods

    def \_\_iter\_\_(self):

        return iter(self.goods)

    def \_\_next\_\_(self):

        return next(self.goods)

    def \_\_getitem\_\_(self,index):

        return self.goods[index]

    def \_\_len\_\_(self):

        return len(self.goods)

    def format\_to\_dict(self):

        return {good.pid:good.\_\_dict\_\_ for good in self.goods}

    def format\_to\_list(self):

        return [good.\_\_dict\_\_ for good in self.goods]

    def format\_to\_json(self):

        return json.dumps(self.format\_to\_list())

    def from\_dict(self,dict):

        self.goods = [Good(\*\*good) for good in dict.values()]

    def from\_list(self,list):

        self.goods = [Good(\*\*good) for good in list]

    def from\_json(self,json\_str):

        self.goods = [Good(\*\*good) for good in json.loads(json\_str)]

    def save(self,filename):

        with open(filename,"w") as f:

            f.write(self.format\_to\_json())

    def load(self,filename):

        with open(filename,"r") as f:

            self.from\_json(f.read())

    def \_\_add\_\_(self,other):

        return GoodManager(\*self.goods,\*other.goods)

    def \_\_sub\_\_(self,other):

        return GoodManager(\*[good for good in self.goods if good not in other.goods])

    def \_\_and\_\_(self,other):

        return GoodManager(\*[good for good in self.goods if good in other.goods])

    def \_\_or\_\_(self,other):

        return GoodManager(\*self.goods,\*[good for good in other.goods if good not in self.goods])

if \_\_name\_\_ == "\_\_main\_\_":

    # 商品编号：1，品名：apple，价格：5，数量：100，出厂日期：2020-1-1

    g1 = Good(1,"apple",5,100,"2020-1-1")

    g2 = Good(2,"banana",3,200,"2020-1-2")

    g3 = Good(3,"orange",4,150,"2020-1-3")

    # 商品管理器(内部用列表存储商品对象)

    gm = GoodManager(g1,g2,g3)

    print("商品管理器：\n",gm)

    print("字典形式：",gm.format\_to\_dict())

    print("列表形式：",gm.format\_to\_list())

    print("Json：",gm.format\_to\_json())

    while True:

        print("add:添加商品")

        print("remove:删除商品")

        print("search:查找商品")

        print("update:修改商品")

        print("save:保存商品")

        print("load:加载商品")

        print("exit:退出")

        input\_str = input("输入操作：")

        if input\_str == "add":

            pid = input("输入商品编号：")

            pname = input("输入品名：")

            price = input("输入价格：")

            num = input("输入数量：")

            pdate = input("输入出厂日期：")

            gm.add(Good(pid,pname,price,num,pdate))

            print("添加成功")

        elif input\_str == "remove":

            pid = input("输入商品编号：")

            good = gm.search(pid)

            if good:

                gm.remove(good)

                print("删除成功")

            else:

                print("商品不存在")

        elif input\_str == "search":

            pid = input("输入商品编号：")

            good = gm.search(pid)

            if good:

                print(good)

            else:

                print("商品不存在")

        elif input\_str == "update":

            pid = input("输入商品编号：")

            good = gm.search(pid)

            if good:

                pname = input("输入品名：")

                price = input("输入价格：")

                num = input("输入数量：")

                pdate = input("输入出厂日期：")

                good.pname = pname

                good.price = price

                good.num = num

                good.pdate = pdate

                gm.update(good)

                print("修改成功")

            else:

                print("商品不存在")

        elif input\_str == "save":

            filename = input("输入文件名：")

            gm.save(filename)

            print("保存成功")

        elif input\_str == "load":

            filename = input("输入文件名：")

            gm.load(filename)

            print("加载成功")

        elif input\_str == "exit":

            break

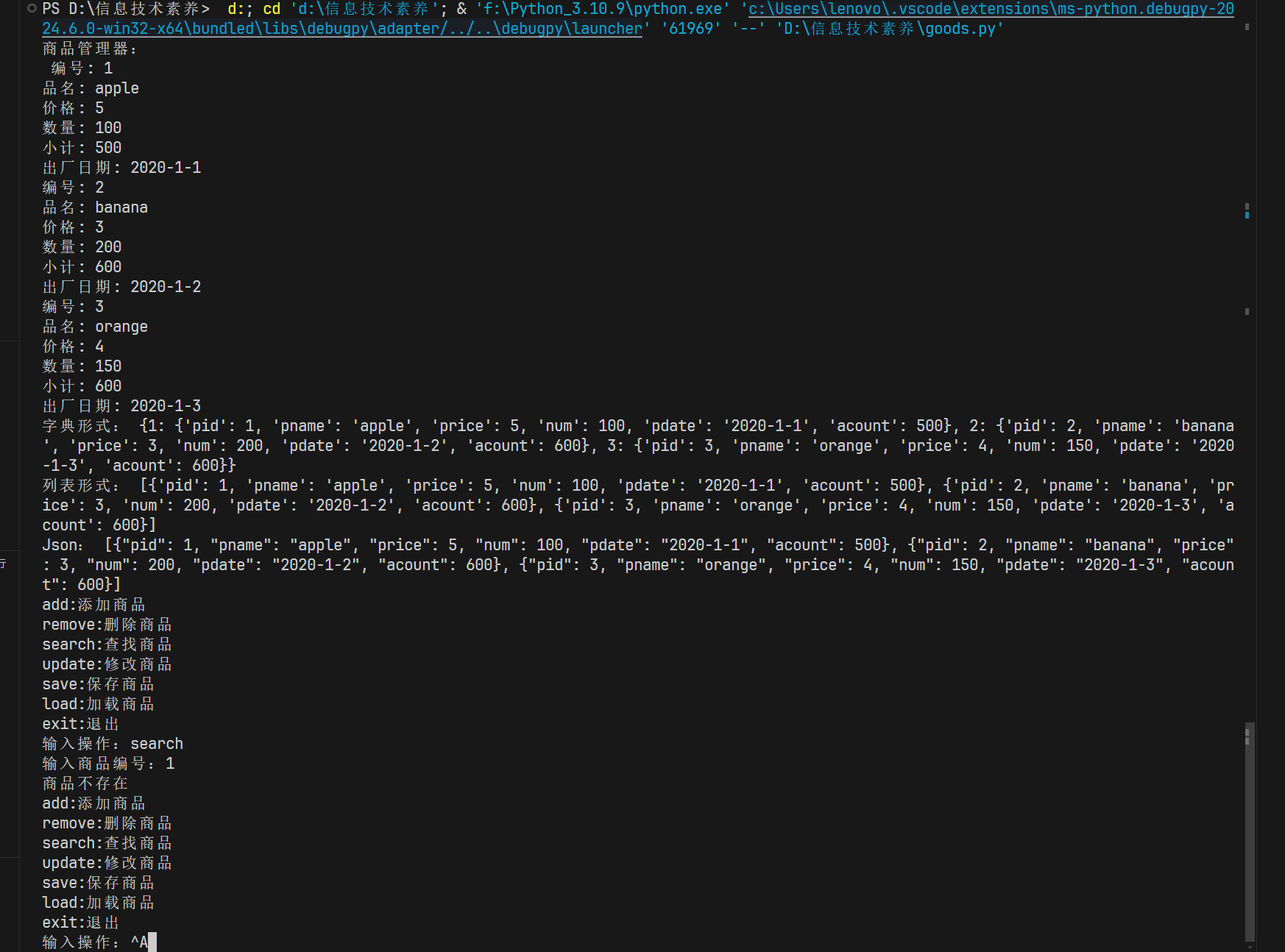
        else:

            print("无效操作")

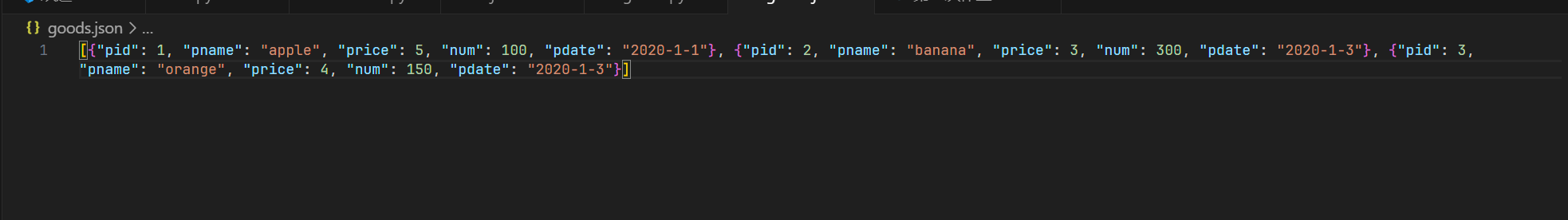
其中Good类表示商品，GoodManager类表示商品管理，GoodManager类通过List来储存Good类

Good类实现了商品的比较方法，而GoodManager类则实现了商品的添加，删除，查找，排序，转存Json等功能

运行截图如下：



其中商品信息的加载与保存是通过json格式进行的，保存的json文件如下：



format\_to\_dict（self）方法则将商品信息以商品编号为键导出为字典，方便管理

改成字典储存的代码如下：

import json

class Good:

    # 编号pid  品名：pname,价格：price,数量：num,小计:acount,出厂日期：pdate

    def \_\_init\_\_(self,pid,pname,price,num,pdate):

        self.pid = pid

        self.pname = pname

        self.price = price

        self.num = num

        self.pdate = pdate

        self.acount = price \* num

    def \_\_str\_\_(self):

        return (

f"""编号: {self.pid}

品名: {self.pname}

价格: {self.price}

数量: {self.num}

小计: {self.acount}

出厂日期: {self.pdate}"""

        )

    def \_\_repr\_\_(self):

        return self.\_\_str\_\_()

    def \_\_eq\_\_(self,other):

        return self.pid == other.pid

    def \_\_lt\_\_(self,other):

        return self.price < other.price

    def \_\_gt\_\_(self,other):

        return self.price > other.price

    def \_\_le\_\_(self,other):

        return self.price <= other.price

    def \_\_ge\_\_(self,other):

        return self.price >= other.price

    def \_\_ne\_\_(self,other):

        return self.price != other.price

class GoodManager:

    def \_\_init\_\_(self, \*args):

        # 商品字典(dict)

        self.goods = {}

        if args:

            for good in args:

                self.goods[good.pid] = good

    def add(self, good):

        self.goods[good.pid] = good

    def remove(self, good):

        if good.pid in self.goods:

            del self.goods[good.pid]

    def \_\_str\_\_(self):

        return "\n".join([str(good) for good in self.goods.values()])

    def \_\_repr\_\_(self):

        return self.\_\_str\_\_()

    def sort(self):

        # 返回排序后的商品列表

        return sorted(self.goods.values(), key=lambda good: good.pid)

    def sort\_by\_price(self):

        return sorted(self.goods.values(), key=lambda good: good.price)

    def sort\_by\_num(self):

        return sorted(self.goods.values(), key=lambda good: good.num)

    def sort\_by\_pname(self):

        return sorted(self.goods.values(), key=lambda good: good.pname)

    def search(self, pid):

        return self.goods.get(pid, None)

    def update(self, good):

        if good.pid in self.goods:

            self.goods[good.pid] = good

    def get(self):

        return list(self.goods.values())

    def \_\_iter\_\_(self):

        self.\_iter\_index = iter(self.goods.values())

        return self

    def \_\_next\_\_(self):

        return next(self.\_iter\_index)

# 示例使用

if \_\_name\_\_ == "\_\_main\_\_":

    gm = GoodManager()

    gm.add(Good(1, "Apple", 10, 100, "2021-01-01"))

    gm.add(Good(2, "Banana", 20, 150, "2021-01-02"))

    print(gm)

    for good in gm:

        print(good)

运行截图：

