《MRP 算法设计及应用》实 验 报 告

学号：20002462　　姓名：刘子言　　班级： 计 203　　成绩：

实验名称：MRP 计算	**实验地点**：信息楼 221
所使用的工具软件及环境：Pycharm（Python3.8），MySQL	

一、实验目的：

熟悉 MRP 算法，能应用 MRP 计算出采购计划和生产计划，并应用资产负债表公式输出相应的公式

二、实验内容：

1、输入主生产计划的记录
2、MRP计算
3、输出相应子物料的采购计划和生产计划，根据需求日期的先后排序
4、输入变量名，输出相应的资产负载公式

三、实验步骤：

1、请简要叙述系统应用技术（如应用的编程语言、数据库或文件技术等）

本系统采用基于 Django 框架的全栈式开发方式：后端基于 Python 语言 Django 框架，前端运用 HTML5 语言实现页面设计；

本系统运用的数据库为 MySQL，可基于 MySQL workbench 8.0 CE 工具进行数据库表的可视化编辑；

主要的技术连接是将前后端代码及数据库接口均集成于 Django 框架之中，运用 Pycharm 软件进行系统开发。

2、请叙述数据存储的方式（如果是数据库，请列出表结构；如果是文件，请列出文件中记录的结构）

本系统采用 MySQL 数据库存储数据。

• 数据库名称：ERP_db

• 共设计 5 张原始数据的数据表，每张表的表结构如下所示：

1）BOM 表（表名：Bom）

Column	Type	Nullable	Indexes
id	bigint	no	primary
part_num	int	no	
part_name	varchar(32)	no	
part_quantity	int	no	
part_level	int	no	

2）物料表（表名：Material）

Column	Type	Nullable	Indexes
id	bigint	no	primary
m_num	int	no	
m_name	varchar(32)	no	
m_way	smallint	no	
m_wastage	decimal(3,2)	no	
work_time	int	no	

3）调配构成表（表名：Allocation）

Column	Type	Nullable	Indexes
id	bigint	no	primary
a_num	int	no	
a_code	varchar(32)	no	
materiel_time	int	no	
manufacture_time	int	no	
child_m_id	bigint	no	foreign(Material)
father_m_id	bigint	no	foreign(Material)

4）库存表（表名：Cash）

Column	Type	Nullable	Indexes
id	bigint	no	primary
operation_c	int	no	
materiel_c	int	no	
cash_m_id	bigint	no	foreign(Material)

5）资产负债表（表名：Balance）

Column	Type	Nullable	Indexes
id	bigint	no	primary
description	longtext	no	
guid_mark	varchar(16)		
guid_num	int		
asset_name	varchar(32)		

• 除了以上 5 张存储原始数据的数据表此外，对于输入的查询信息，在本系统中也设计了两张数据表分别存储（分别针对 MPS 计算和资产负债公式计算），并在新的查询开始前删除表中以往查询记录。这两张表的表结构如下：

1）MPS 记录表（表名：MPS）

Column	Type	Nullable	Indexes
id	bigint	no	primary
MPS_name	varchar(32)	no	
MPS_num	int	no	
MPS_time	data	no	

2）变量名记录表（表名：Asset）

Column	Type	Nullable	Indexes
id	bigint	no	primary
asset_name	varchar(32)	no	

注：上述所使用的数据库 ERP_db 通过 Django 框架下的 settings.py 文件中设置与系统连接；其中每张表均在 Django 框架下的 models.py 文件中定义、设计，并迁移至 MySQL 数据库中。

3、请叙述系统的模块功能和具体的实现步骤，并画出所应用算法的流程图

本系统的功能模块主要分为两大部分：一是 MPS 计算，二是资产负债表公式计算。

（1）MPS 相关计算

1）输入输出设计

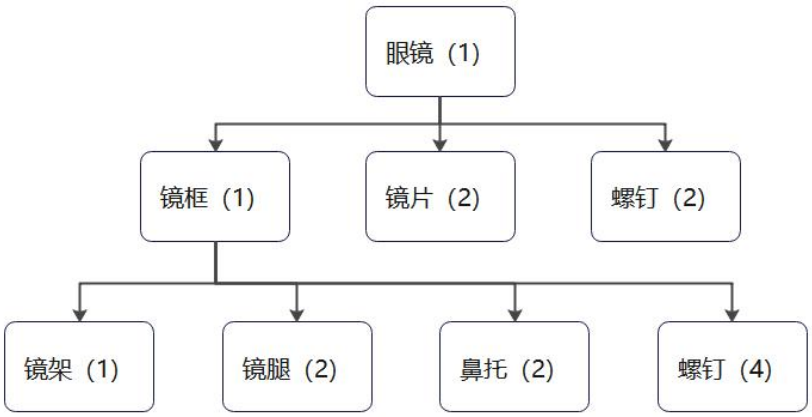
输入一条 MPS 记录，输出相应的采购计划和生产计划（没有并单处理）

输入输出均设计了相应的界面

2）基本算法思想

MPS 相关计算中，主要运用到了**深度优先遍历**算法的思想。

针对眼镜这一生产对象，画出各物料之间的层次结构（树）如下：



依据上述层次图，利用**多阶展开方式**，逐步将产品眼镜展开为各个零部件，即除了显示直接下一阶的组件项目外，还按序号展开下下阶及下下下阶，直到阶层结束为止，在实现过程中类似树形结构的深度优先遍历算法，遍历的结果如下图：

产品零件号：#20000-眼镜　层次0				
零件号	描述	装配数量	单位	层次
20100	镜框	1	个	1
.20110	镜架	1	个	2
.20120	镜腿	2	个	2
.20130	鼻托	2	个	2
.20109	螺钉	4	个	2
20300	镜片	2	个	1
20109	螺钉	2	个	1

在具体实现过程中，按照以上顺序依次将数据存入数据库 Bom 表中，从数据库中取出数据时按顺序存入列表中，以此实现深度遍历的过程。由于 Python 语言中不利于使用链表结构，因此从数据库中读取出的数据主要是以列表（或复合链表）和字典的形式存放，便于后续使用。

运用深度优先遍历的思想，这样在处理需求量的计算、日程日期的计算时都能够更快的获得上一层物料的相关数据，在处理物料库存时，也能够优先供给最底层的物料需求。比如：眼镜生产中的螺钉，处于第 2 层的螺钉会先使用库存，如果利用深度优先遍历，更利于实现这一要求；同时避免并单处理，可以分别求出螺钉在不同阶段的需求量和采购日程日期。

3）MPS 记录输入

将前端输入的 MPS 记录通过 POST 方式传递到后台，在通过数据合法性验证之后，存入数据库 MPS 表中。每次有新的记录存入之前，都会将表中以往的记录删去，避免在计算时出现错误。

4）需求量的计算

• 各物料需求量的计算公式如下：

子物料需求量 =（父物料需求数*子物料构成数）/（1-损耗率）- 工序库存 - 资材库存

• 数据处理过程

首先从数据库中提取所运用到的数据表的信息；然后再从各数据表中利用多个 for 循环获取对应属性列的数据值，顺便求出层次为 2 的物料种类的个数，便于后续计算；

然后通过 if 语句判断输入的成品名称（眼镜 or 镜框），确认接下来计算需求量的循环次数（即最终呈现在前端的列表的行数）；

最后利用 for 循环求各个子物料的需求量。关键在此 for 循环开始时，需首先判断该子物料的父物料（上一层物料）的种类，以确定计算公式中的“父物料需求数”取到正确的值，最后再分别取该子物料对应的构成数、损耗率、工序库存和资材库存，代入上述公式进行计算，将结果存入新定义的列表中。

5）日程日期计算

• 各物料日程下达和完成日期的计算公式如下：

子物料的日程完成日期 = 父物料的日程下达日期

子物料的日程下达日期 = 子物料的完成日期 - 子物料作业提前期

- 子物料配料提前期 - 子物料供应商提前期

• 数据处理过程

前期与需求量计算的过程类似：首先从数据库中提取所运用到的数据表的信息；然后再从各数据表中利用多个 for 循环获取对应属性列的数据值。然后通过 if 语句判断输入的成品名称，确认接下来计算日程日期的循环次数；这几步均可与“需求量计算”的过程放在一起进行；

最后再利用 for 循环求各个子物料的日程日期。关键在此 for 循环开始时，需首先判断该子物料的父物料（上一层物料）的种类，以确定计算公式中的“父物料的日程下达日期”取到正确的值，并将该值从日期的数据类型转换为可计算的数据类型，最后再分别取该子物料对应的作业提前期、配料提前期、供应商提前期，代入上述公式进行计算，将结果存入新定义的两个列表中。

6）关于返回前端的参数

上述计算结束之后，需要再利用一个 for 循环，将所有需要输出的列表内容逐条重新存放在一个新的复合列表中，再通过接口将该列表作为参数传递给前端进行输出。

本系统最终输出的采购和生产计划表，是按照物料日程下达日期的先后顺序排序的，更利于采购与生产计划的安排。

注：因为 Django 框架在前端 HTML 中的代码不能运用“()、[]”等符号，也缺乏对 range 函数等常用函数的兼容性，所以对多个列表的表格式输出带来了较大的不便，因此需要在后端向前端传递数据之前，将所有的单个列表整合到一个存有逐条待输出记录的复合列表中，这样更有利于前端数据的展示，提高运行效率。

（2）资产负债表公式计算

1）输入输出设计

输入一个要统计的变量名，输出该变量对应的公式

输入输出均设计了相应的界面

2）变量名的记录输入

将前端输入的要统计的变量名通过 POST 方式传递到后台，在通过数据合法性验证之后存入数据库 Asset 表中。每次有新的记录存入之前，都会将表中以往的记录删去，避免在计算时出现错误。

3）变量名对应公式的计算

• 计算实现思路

通过输入的变量名，找到与之对应的资产类的序号 id，再对整张资产负债表进行遍历，查找出所有满足“资产类汇总序号 num = id”的资产类，并将他们的变量名根据资产类方向（+）组合，得到最终的统计公式。

• 数据处理过程

首先从数据库中提取所运用到的数据表的信息（主要是 Balance 和 Asset 表）；

然后从各数据表中利用多个 for 循环获取对应属性列的数据值，包括需要查找的变量名、需要查找的变量名对应的 id、以及汇总序号等于该 id 的变量名集合；

最后通过一个 for 循环将变量名集合转换成一个字符串公式，将结果存入新定义的列表中，作为参数传入前端展示。

四、系统运行情况界面

前端主要有三个界面：首页、MPS 计算界面、资产负债表公式计算界面。

1）首页 选择

• 路由：http://127.0.0.1:8000/index/

• 页面展示如下：

点击第一个按钮，跳转到 MPS 计算界面；点击第二个按钮，跳转到资产负债表公式计算界面。

2）界面 1　MPS 计算

• 路由：输入 http://127.0.0.1:8000/index/mps/

输出 http://127.0.0.1:8000/index/plan/

• 页面初始如下：

• **输入**一条 MPS 记录（以“**眼镜**”为例）

• **点击提交**，下方输出相应的采购和生产计划表（**按日程下达日期先后排序**）

调配方式	物料号	物料名称	需求数量	日程下达日期	日程完成日期
采购	20110	镜架	100	2022-09-26	2022-10-17
采购	20130	鼻托	200	2022-09-28	2022-10-17
采购	20300	镜片	200	2022-09-28	2022-10-19
采购	20120	镜腿	170	2022-10-06	2022-10-17
采购	20109	螺钉	385	2022-10-06	2022-10-17
采购	20109	螺钉	223	2022-10-08	2022-10-19
生产	20100	镜框	100	2022-10-17	2022-10-19
生产	20000	眼镜	100	2022-10-19	2022-10-20

• **输入**一条 MPS 记录（以“**镜框**”为例）

• **点击提交**，下方输出相应的采购和生产计划表（**按日程下达日期先后排序**）

调配方式	物料号	物料名称	需求数量	日程下达日期	日程完成日期
采购	20110	镜架	100	2022-09-27	2022-10-18
采购	20130	鼻托	200	2022-09-29	2022-10-18
采购	20120	镜腿	170	2022-10-07	2022-10-18
采购	20109	螺钉	385	2022-10-07	2022-10-18
生产	20100	镜框	100	2022-10-18	2022-10-20

• 点击最下方“返回首页”按钮，跳转回首页，可重新选择想要进行的计算。

3）界面 2　资产负债表公式计算

路由：输入 http://127.0.0.1:8000/index/balance/

　　输出 http://127.0.0.1:8000/index/count/

• 页面初始如下：

• **输入**一个要统计的变量名（以 **b1** 为例）

• **点击提交**，下方输出该变量 **b1** 对应的公式

• **输入**一个要统计的变量名（以 **b3** 为例）

• **点击提交**，下方输出该变量 **b3** 对应的公式

• 点击最下方“返回首页”按钮，跳转回首页，可重新选择想要进行的计算。

五、程序设计核心代码

1）后端与 MySQL 数据库的连接 [settings.py]

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'ERP_db',    # 连接到 MySQL 数据库 ERP_db
        'USER': 'root',
        'PASSWORD': '123456',
        'HOST': '127.0.0.1',
        'PORT': 3306,
    }
}
```

2）路由设计 [urls.py]

```
urlpatterns = [
    path('index/', views.index),    # 首页
    path('index/mps/', views.MPS_list),    # mps 计算
    path('index/plan/', views.Plan_list),    # mps 结果
    path('index/balance/', views.Balance_list),    # 资产负债计算
    path('index/count/', views.Count_list),    # 资产负债公式结果
]
```

3）数据表设计 [models.py]

```
# BOM 表
class BOM(models.Model):
    """BOM 表"""
    part_num = models.IntegerField(verbose_name="零件号")
    part_name = models.CharField(verbose_name="零件描述", max_length=32)
    part_quantity = models.IntegerField(verbose_name="装配个数")
    part_level = models.IntegerField(verbose_name="所属层次")

# 物料表
class Materiel(models.Model):
    """物料表"""
    m_num = models.IntegerField(verbose_name="物料号")
    m_name = models.CharField(verbose_name="物料名称", max_length=32)
    way_choices = (
        (1, "生产"),
        (2, "采购"),
    )
    m_way = models.SmallIntegerField(verbose_name="调配方式", choices=way_choices)
    m_wastage = models.DecimalField(verbose_name="损耗率", max_digits=3, decimal_places=2)
    work_time = models.IntegerField(verbose_name="作业提前期(/天)")
```

```
# 调配构成表
class Allocation(models.Model):
    """调配构成表"""
    a_num = models.IntegerField(verbose_name="调配基准编号")
    a_code = models.CharField(verbose_name="调配区代码", max_length=32)
    father_m = models.ForeignKey(verbose_name="父物料", to="Materiel", related_name="father_m", on_delete=models.CASCADE)
    child_m = models.ForeignKey(verbose_name="子物料", to="Materiel", related_name="child_m", on_delete=models.CASCADE)
    materiel_time = models.IntegerField(verbose_name="配料提前期(/天)")
    manufacture_time = models.IntegerField(verbose_name="供应商提前期(/天)")

# 库存表
class Cash(models.Model):
    """库存表"""
    cash_m = models.ForeignKey(verbose_name="库存物料", to="Materiel", on_delete=models.CASCADE)
    operation_c = models.IntegerField(verbose_name="工序库存")
    materiel_c = models.IntegerField(verbose_name="资材库存")

# MPS 记录
class MPS(models.Model):
    """每条 MPS 记录"""
    MPS_name = models.CharField(verbose_name="产品名称", max_length=32)
    MPS_num = models.IntegerField(verbose_name="数量")
    MPS_time = models.DateField(verbose_name="完工日期")

# 资产负债表
class Balance(models.Model):
    """资产负债表"""
    description = models.TextField(verbose_name="资产类说明")
    guid_mark = models.CharField(verbose_name="资产类方向", max_length=16, null=True)
    guid_num = models.IntegerField(verbose_name="资产类汇总序号", null=True)
    asset_name = models.CharField(verbose_name="变量名", max_length=32, null=True)

# balance 变量记录
class Asset(models.Model):
    """每次查询的变量记录"""
    asset_name = models.CharField(verbose_name="变量名", max_length=32)
```

4）MPS 计算 [views.py]

```
def Plan_list(request):
    """MPS 计算 显示采购计划和生产计划"""
```

```
    # 从数据库中提取所有表的数据
    bom = models.BOM.objects.all()
    materiel = models.Materiel.objects.all()
    allo = models.Allocation.objects.all()
    cash = models.Cash.objects.all()
    mps_f = models.MPS.objects.filter()

    # 在数据表中获取对应列的数据
    MPS_name = []  # 成品名称
    MPS_num = []  # 成品需求量
    MPS_time = []  # 完工日期
    for obj in mps_f:
        MPS_name = obj.MPS_name
        MPS_num = obj.MPS_num
        MPS_time = obj.MPS_time

    part_quantity = []  # 装配个数
    level_2 = 0  # 层次为 2 的物料数
    for obj in bom:
        part_quantity.append(obj.part_quantity)
        if obj.part_level == 2:
            level_2 = level_2 + 1

    m_way = []  # 调配方式
    m_num = []  # 物料号
    m_name = []  # 物料名称
    m_wastage = []  # 损耗率
    work_time = []  # 作业提前期
    for obj in materiel:
        m_way.append(obj.get_m_way_display())
        m_num.append(obj.m_num)
        m_name.append(obj.m_name)
        m_wastage.append(obj.m_wastage)
        work_time.append(obj.work_time)

    materiel_time = []  # 配料提前期(len = 7)
    manufacture_time = []  # 供应商提前期(len = 7)
    materiel_time.append(0)  # 将列表长度变为 8 位
    manufacture_time.append(0)  # 将列表长度变为 8 位
    for obj in allo:
        materiel_time.append(obj.materiel_time)
        manufacture_time.append(obj.manufacture_time)

    operation_c = []  # 工序库存
```

```
    materiel_c = []  # 资材库存
    for obj in cash:
        operation_c.append(obj.operation_c)
        materiel_c.append(obj.materiel_c)

    # 判断产品种类  计算循环次数 / 返回列表的行数
    list_num = 0
    if MPS_name == '眼镜':
        list_num = len(part_quantity)
    elif MPS_name == '镜框':
        list_num = level_2 + 1

    # 计算需求量 = 装配个数*父物料需求量/（1-损耗率） - 工序库存 - 资材库存
    need_num = {}  # 需求量 注意不能用[]，这是用列表装了字典的内容
    if MPS_name == '眼镜':
        for i in get_range(list_num):
            need_num[i] = math.ceil(part_quantity[i] * MPS_num / (1 - m_wastage[i]) -
                                    operation_c[i] - materiel_c[i])    # math.ceil 向上取整
    elif MPS_name == '镜框':
        for i in get_range(list_num):
            need_num[i] = math.ceil(part_quantity[i + 1] * MPS_num / (1 - m_wastage[i +
                                    1]) - operation_c[i + 1] - materiel_c[i + 1])

    # 计算下达/完成日期 下达日期 = 完成日期 - 作业提前期 - 配料提前期 - 供应商提前期
    # 注：换算完工日期的方法：由于 relativedelta 函数模块导入不成功，所以换用datetime.timedelta()方式
    mps_time = datetime.datetime.strptime(str(MPS_time), '%Y-%m-%d')
    start_time = {}  # 日程下达时间
    bingo_time = {}  # 日程完成日期
    if MPS_name == '眼镜':
        for i in get_range(list_num):
            if i == 0:  # 成品/树根
                bingo_time[i] = mps_time
                start_time[i] = bingo_time[i] - datetime.timedelta(days=work_time[i])
            elif i == 1 or i == 6 or i == 7:  # 父物料/树的第一层
                bingo_time[i] = start_time[0]
                start_time[i] = bingo_time[i] - datetime.timedelta(
                    days=(work_time[i] + materiel_time[i] + manufacture_time[i]))
            else:  # 子物料/树的第二层
                bingo_time[i] = start_time[1]
                start_time[i] = bingo_time[i] - datetime.timedelta(
                    days=(work_time[i] + materiel_time[i] + manufacture_time[i]))
    elif MPS_name == '镜框':
```

```
            for i in get_range(list_num):
                if i == 0:  # 成品/父物料/树根
                    bingo_time[i] = mps_time
                    start_time[i] = bingo_time[i] - datetime.timedelta(days=work_time[i + 1])
                else:  # 子物料/树的第一层
                    bingo_time[i] = start_time[0]
                    start_time[i] = bingo_time[i] - datetime.timedelta(days=(work_time[i + 1]
                                    + materiel_time[i + 1] + manufacture_time[i + 1]))

        # 制作返回的列表
        plan_list = []  # 传递给前端的采购计划和生产计划表
        for i in get_range(list_num):
            if MPS_name == '眼镜':
                list_i = [m_way[i], m_num[i], m_name[i], need_num[i],
                    start_time[i].strftime('%Y-%m-%d'), bingo_time[i].strftime('%Y-%m-%d')]
                plan_list.append(list_i)
            elif MPS_name == '镜框':
                list_i = [m_way[i + 1], m_num[i + 1], m_name[i + 1], need_num[i],
                    start_time[i].strftime('%Y-%m-%d'), bingo_time[i].strftime('%Y-%m-%d')]
                plan_list.append(list_i)

        # 将列表按照日程下达日期由早到晚排序
        for i in get_range(list_num - 1):
            for j in get_range(list_num - 1 - i):
                if plan_list[j][4] > plan_list[j + 1][4]:
                    temp = plan_list[j + 1]
                    plan_list[j + 1] = plan_list[j]
                    plan_list[j] = temp

        return render(request, 'mps.html', {"plan_list": plan_list, "msg": "提交成功"})
```

5）资产负债表公式计算 [views.py]

```
def Count_list(request):
    """资产负债计算  显示公式"""
    # 从数据库中提取所有表的数据
    balance = models.Balance.objects.all()
    asset = models.Asset.objects.filter()

    # 在数据表中获取对应列的数据
    u_name = []    # 需要查找的变量名
    for obj in asset:
        u_name = obj.asset_name

    u_name_id = []    # 需要查找的变量名对应的 id
```

```
    for obj in balance:
        if obj.asset_name == u_name:
            u_name_id = obj.id

    o_name = []     # 找出包含的变量集合
    for obj in balance:
        if obj.guid_num == u_name_id:
            o_name.append(obj.asset_name)

    # 将变量集合转换成字符串公式
    formula = str(u_name) + '='
    for i in get_range(len(o_name)):
        if i == 0:
            formula = formula + str(o_name[i])
        else:
            formula = formula + '+' + str(o_name[i])

    return render(request, 'balance.html', {"formula": formula, "u_name": u_name, "msg": "提
交成功"})
```

6）前端交互 [templates 文件]

• **mps.html**

```
<!--以下为 输入 内容-->
<h3>输入一条 MPS 记录</h3>
<form method="post">
    {% csrf_token %}
    <label>产品名称</label>
    <input type="text" placeholder="名称" name="MPS_name"/>
           
    <label>数量</label>
    <input type="text" placeholder="数量" name="MPS_num"/>
           
    <label>完工日期</label>
    <input id="dt" type="text" autocomplete="off" placeholder="日期" name="MPS_time"/>
           
    <button type="submit">提 交</button>
          
    {{msg}}
</form>

<!--以下为 输出 内容-->
<div style="width:50%">
<h3>相应的采购和生产计划</h3>
```

```
<h4    align="center"><font color="green">MRP 计算结果表</font></h4>
<table class=table4_10 align="center">
      <thead>
            <tr>
                  <th>调配方式</th>
                  <th>物料号</th>
                  <th>物料名称</th>
                  <th>需求数量</th>
                  <th>日程下达日期</th>
                  <th>日程完成日期</th>
            </tr>
      </thead>
      <tbody>
            {% for items in plan_list %}
            <tr>
                  <td>{{ items.0 }}</td>
                  <td>{{ items.1 }}</td>
                  <td>{{ items.2 }}</td>
                  <td>{{ items.3 }}</td>
                  <td>{{ items.4 }}</td>
                  <td>{{ items.5 }}</td>
            </tr>
            {% endfor %}
      </tbody>
</table>
```

• balance.html

```
<!--以下为 输入 内容-->
<form method="post">
      {% csrf_token %}
      <label>变量名称(比如：b1)：</label>
      <input type="text" placeholder="变量名" name="asset_name"/>
              
      <button type="submit">提 交</button>
              
      {{msg}}
</form>

<!--以下为 输出 内容-->
<div style="width:50%">
<h3>根据资产负债表的公式查询表信息</h3>
<h4    align="center"><font color="green">公式计算结果表</font></h4>
<table class=table4_10 align="center">
      <thead>
```

```
            <tr>
                <th>该变量{{ u_name }}对应的公式</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>{{ formula }}</td>
            </tr>
        </tbody>
</table>
```

六、实验体会

通过本次系统的设计，我进一步深入地了解了 MRP 算法及其实现过程，掌握了如何应用 MRP 计算出采购计划和生产计划，以及如何应用资产负债表输出相应的公式。

通过 MPS 算法，我们能够快速地得出组成成品的各物料的采购计划和生产计划；将计划表有序展示，能够很好的帮助生产者安排生产与采购进度。因此这一算法的实现在现实生活中具有较大的应用价值。

本次系统的设计同时也锻炼了我的个人编程能力，加强了我对 Python 语言、Django 框架的掌握，以及提升了对全栈开发的应用熟练度。

本次实验还提高了我解决问题的能力。在实验过程中遇到了一些困难，我通过积极查阅相关资料、与同学交流等方式一一解决，并在此过程中不断地总结提升，收获颇丰。

七、教师评语

该学生完成了 ERP 系统之 MRP 算法、资产负载表计算的实验，（基本/较好很好）地完成了实验任务，算法设计（基本/较/很）合理，实验结果（基本/较/完全）符合要求，实验体会（尚可/较深刻/深刻）。

因此总体评价为_____。

教师签字：

2022 年 10 月 30 日