Optimizing the Selection of LEGO Sets

12/14/2021

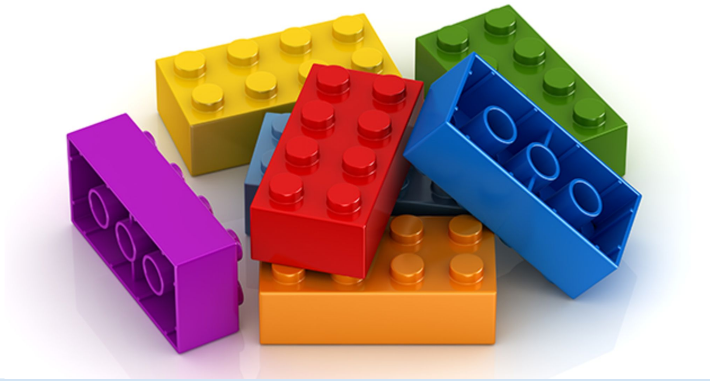
Dan He

Problem Idea

Initial thoughts	Idea	Why important and interesting	Who would care
I am a big fan of LEGO, but I realized that **sometimes it might be hard for a new LEGO player to choose the best experienced sets** (which with highest star ratings) to start.	Create an optimization program that **automatically assign best rating sets for customers** based on their price limit and number of sets they want.	● Create the **optimized experience** for new LEGO players ● Boost customer experience and **loyalty** for LEGO Company	● LEGO Players ● LEGO Company

Optimization Model Formulation

- **Decision Variable:**
  - Assign lego sets “i” to LEGO player “j”?
- **Objective:**
  - maximize the averaged ratings of all the LEGO sets selected
- **Constraints:**
  - Each customer gets the number of sets they want
  - The overall cost should be lower or equal to the customer’s price limit
  - They cannot be assigned products that they own before
  - The total number of each LEGO set should be lower or equal to its inventory



Tools Used

Model Explanation

Create model.x [i , j]

- it would be expected to be 0 and 1, each customer would either be assigned to buy this LEGO set or not. There will be 500 LEGO sets and 200 customers.

Add Constraints

- Inventory
- Price limit
- Item already exist
- Item count

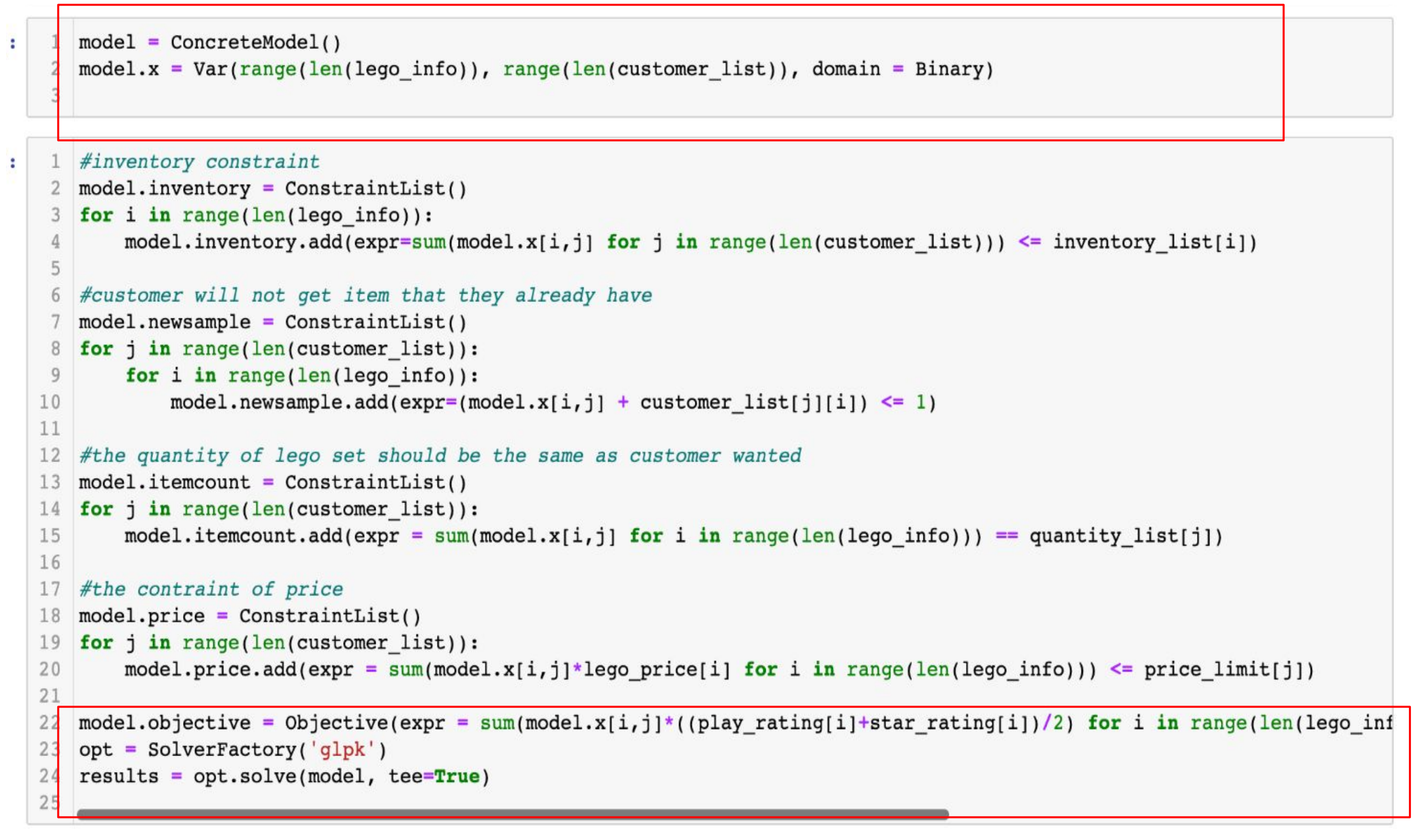
Average play rating and star rating

- (play rating + star rating)/2

Set objective: model.x [i , j]*averaged rating

- maximize the objective

Main Tool: Python (with “glpk” package)

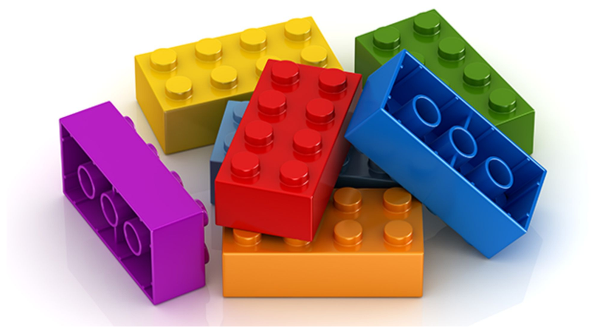


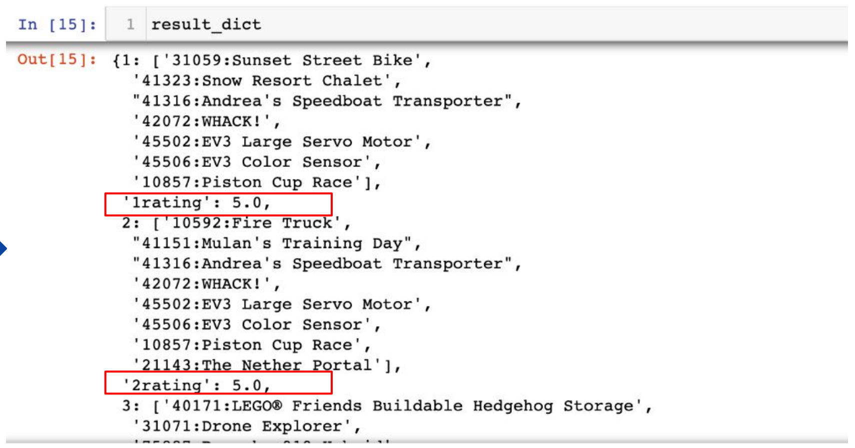
```
model = ConcreteModel()
model.x = Var(range(len(lego_info)), range(len(customer_list)), domain = Binary)

```

```
#inventory constraint
model.inventory = ConstraintList()
for i in range(len(lego_info)):
    model.inventory.add(expr=sum(model.x[i,j] for j in range(len(customer_list))) <= inventory_list[i])

#customer will not get item that they already have
model.newsample = ConstraintList()
for j in range(len(customer_list)):
    for i in range(len(lego_info)):
        model.newsample.add(expr=(model.x[i,j] + customer_list[j][i]) <= 1)

#the quantity of lego set should be the same as customer wanted
model.itemcount = ConstraintList()
for j in range(len(customer_list)):
    model.itemcount.add(expr = sum(model.x[i,j] for i in range(len(lego_info))) == quantity_list[j])

#the contraint of price
model.price = ConstraintList()
for j in range(len(customer_list)):
    model.price.add(expr = sum(model.x[i,j]*lego_price[i] for i in range(len(lego_info))) <= price_limit[j])

model.objective = Objective(expr = sum(model.x[i,j]*((play_rating[i]+star_rating[i])/2) for i in range(len(lego_inf
opt = SolverFactory('glpk')
results = opt.solve(model, tee=True)

```

Assumptions and Challenges

- **Assumptions**
  - The average play rating and star rating represents the customer experience
  - Maximize the average rating can optimize the customer LEGO experience
- **Challenges**
  - Print out each customer's LEGO sets rating
  - challenging because the objective is maximizing the total averaged rating of all LEGO sets
- **Changes made**
  - After the objective calculated, print out each average rating per customer in a for loop
  - Each rating corresponding to its customer id





```
In [15]:  result_dict

Out[15]: {1: ['31059:Sunset Street Bike',
           '41323:Snow Resort Chalet',
           "41316:Andrea's Speedboat Transporter",
           '42072:WHACK!',
           '45502:EV3 Large Servo Motor',
           '45506:EV3 Color Sensor',
           '10857:Piston Cup Race'],
          '1rating': 5.0,
          2: ['10592:Fire Truck',
           "41151:Mulan's Training Day",
           "41316:Andrea's Speedboat Transporter",
           '42072:WHACK!',
           '45502:EV3 Large Servo Motor',
           '45506:EV3 Color Sensor',
           '10857:Piston Cup Race',
           '21143:The Nether Portal'],
          '2rating': 5.0,
          3: ['40171:LEGO® Friends Buildable Hedgehog Storage',
           '31071:Drone Explorer',
```

Hypothetical Client(s)

- Me/LEGO players
- LEGO Company
  - Can be launched in their website to help users choose their sets
- An reference tool for someone who is really interested in LEGO investment
  - Help to select most popular items so that they can sell these sets when price goes up

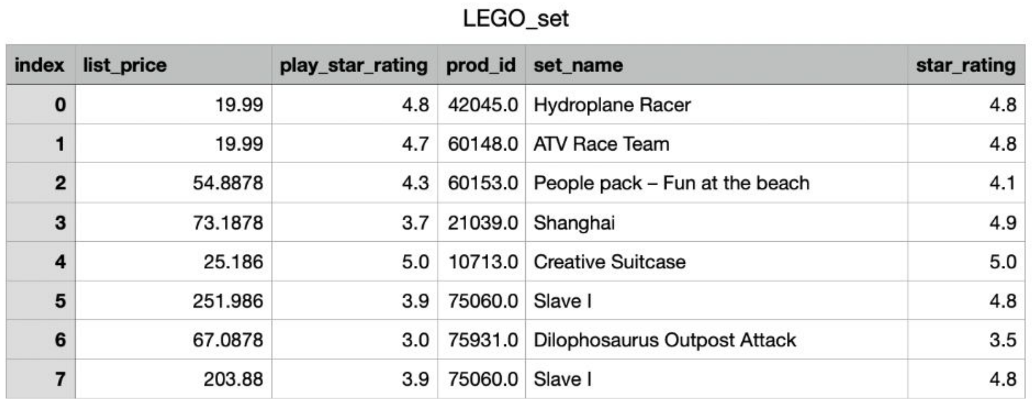
Benefits and Limitations

- **Benefits**
  - User friendly, help them choose the best sets in an efficient way
  - Save time and reduce errors for beginners
- **Limitations**
  - The program will need a long time to find the optimized solution if the inventory is not fully packed
- **Next Steps**
  - Optimize the model by leveraging some optimization algorithms to shorten the time of running the program

Supporting Files -- Data

- A spreadsheet contains all LEGO sets name, price, and ratings
- A sheet contains all customer information, including customer id, price limit, items they have
- Another sheet including inventory of LEGO sets

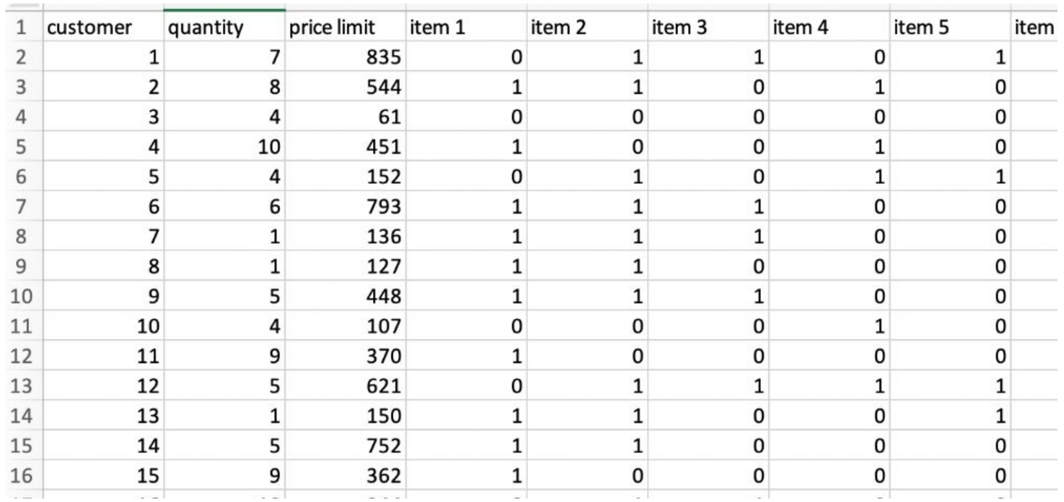
1



LEGO_set

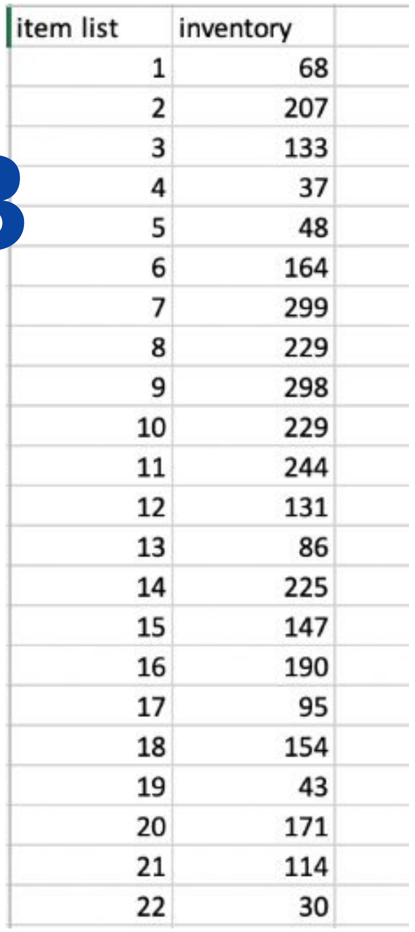
index	list_price	play_star_rating	prod_id	set_name	star_rating
0	19.99	4.8	42045.0	Hydroplane Racer	4.8
1	19.99	4.7	60148.0	ATV Race Team	4.8
2	54.8878	4.3	60153.0	People pack – Fun at the beach	4.1
3	73.1878	3.7	21039.0	Shanghai	4.9
4	25.186	5.0	10713.0	Creative Suitcase	5.0
5	251.986	3.9	75060.0	Slave I	4.8
6	67.0878	3.0	75931.0	Dilophosaurus Outpost Attack	3.5
7	203.88	3.9	75060.0	Slave I	4.8

2



1	customer	quantity	price limit	item 1	item 2	item 3	item 4	item 5	item
2	1	7	835	0	1	1	0	1	
3	2	8	544	1	1	0	1	0	
4	3	4	61	0	0	0	0	0	
5	4	10	451	1	0	0	1	0	
6	5	4	152	0	1	0	1	1	
7	6	6	793	1	1	1	0	0	
8	7	1	136	1	1	1	0	0	
9	8	1	127	1	1	0	0	0	
10	9	5	448	1	1	1	0	0	
11	10	4	107	0	0	0	1	0	
12	11	9	370	1	0	0	0	0	
13	12	5	621	0	1	1	1	1	
14	13	1	150	1	1	0	0	1	
15	14	5	752	1	1	0	0	0	
16	15	9	362	1	0	0	0	0	

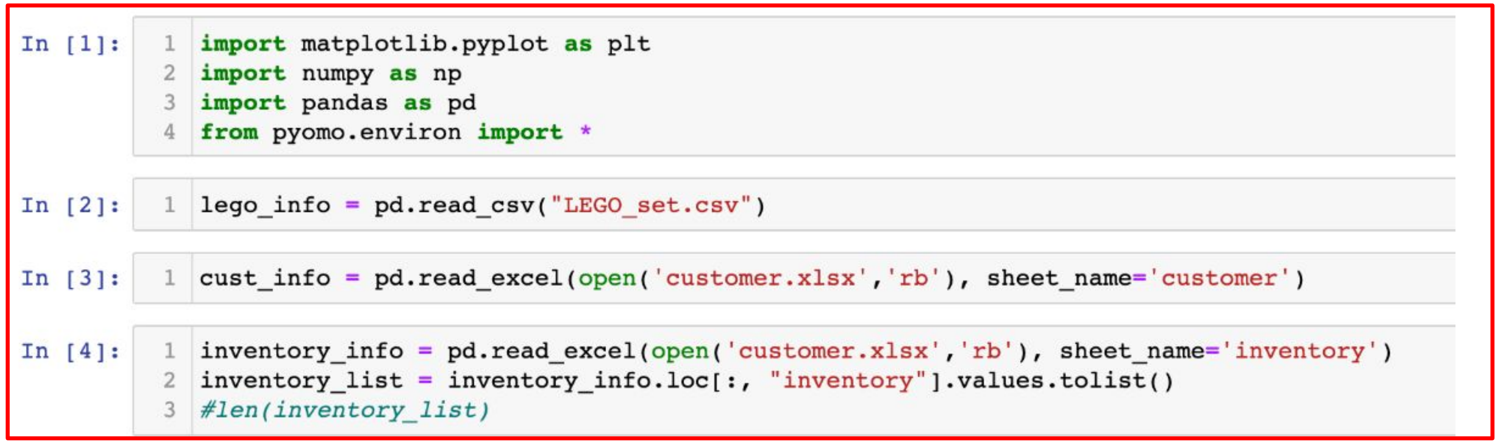
3



item list	inventory
1	68
2	207
3	133
4	37
5	48
6	164
7	299
8	229
9	298
10	229
11	244
12	131
13	86
14	225
15	147
16	190
17	95
18	154
19	43
20	171
21	114
22	30

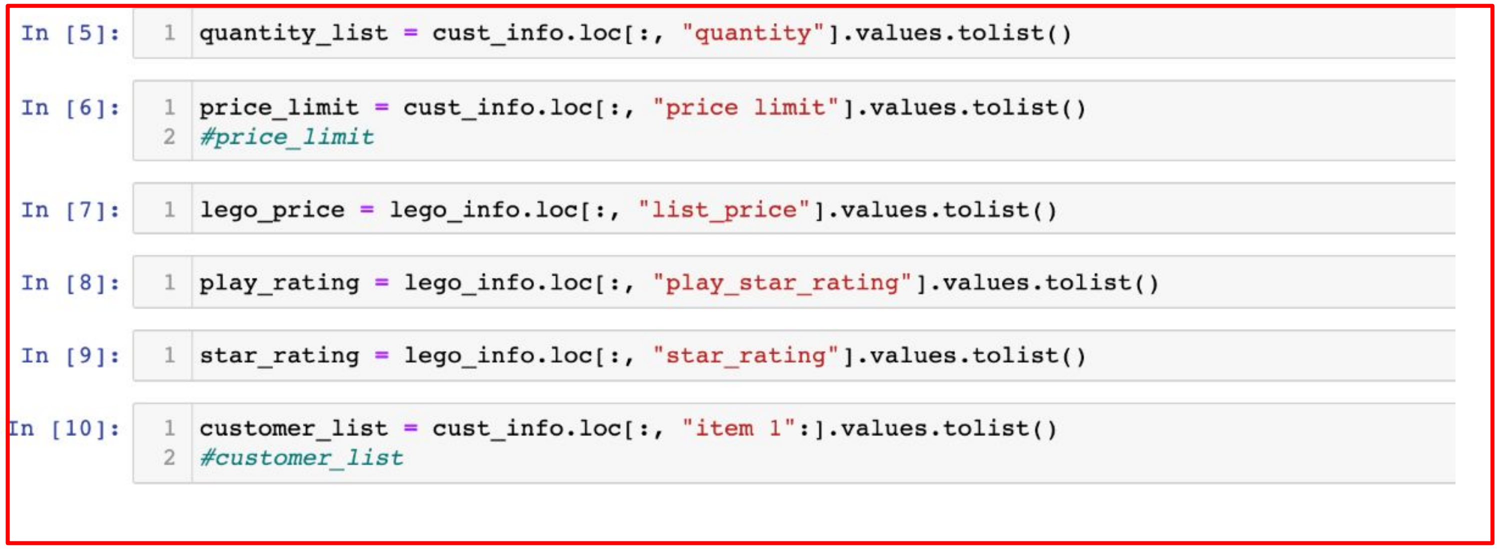
Supporting Files -- Coding Scripts

- Step 1: Loading data



```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        from pyomo.environ import *

In [2]: lego_info = pd.read_csv("LEGO_set.csv")

In [3]: cust_info = pd.read_excel(open('customer.xlsx','rb'), sheet_name='customer')

In [4]: inventory_info = pd.read_excel(open('customer.xlsx','rb'), sheet_name='inventory')
        inventory_list = inventory_info.loc[:, "inventory"].values.tolist()
        #len(inventory_list)
```

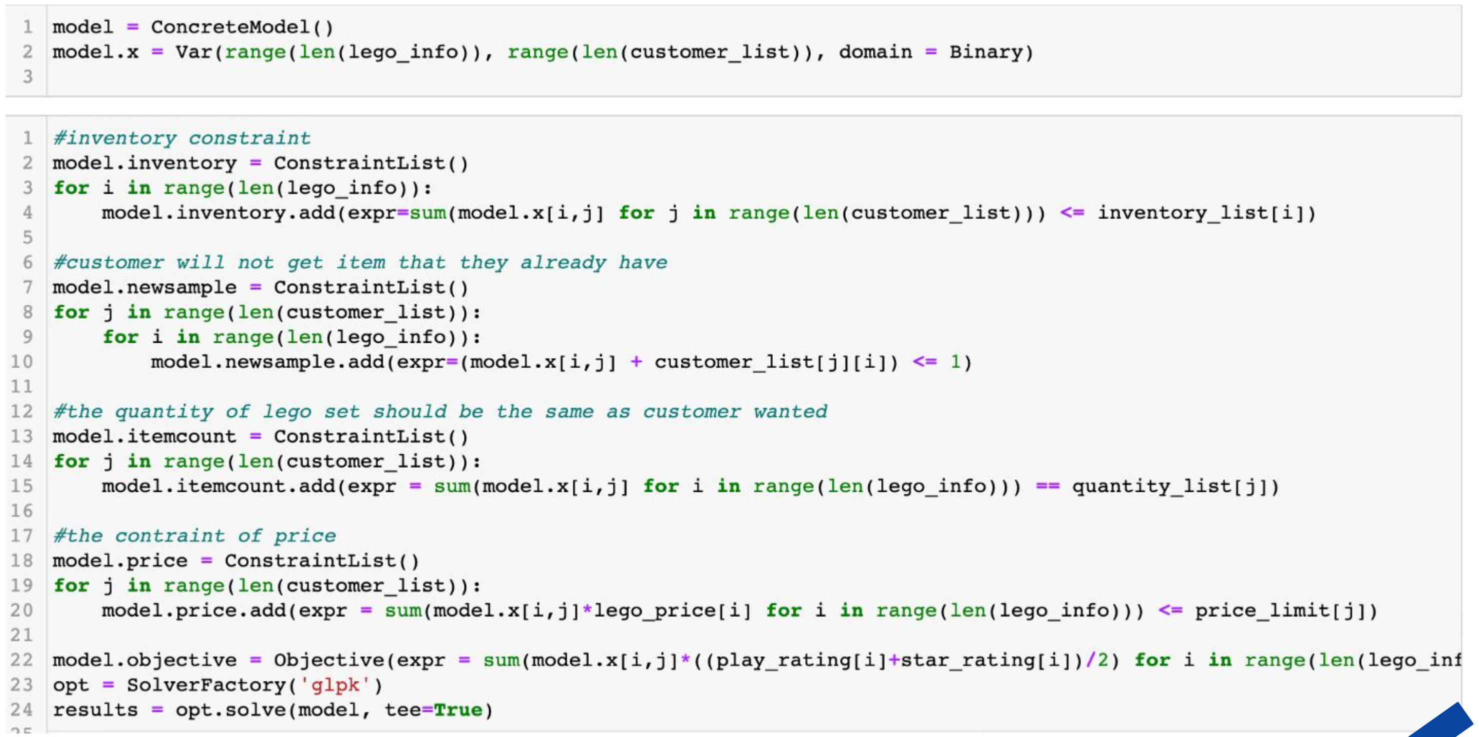
- Step 2: Organize data into different lists



```
In [5]: quantity_list = cust_info.loc[:, "quantity"].values.tolist()

In [6]: price_limit = cust_info.loc[:, "price limit"].values.tolist()
        #price_limit

In [7]: lego_price = lego_info.loc[:, "list_price"].values.tolist()

In [8]: play_rating = lego_info.loc[:, "play_star_rating"].values.tolist()

In [9]: star_rating = lego_info.loc[:, "star_rating"].values.tolist()

In [10]: customer_list = cust_info.loc[:, "item 1":].values.tolist()
         #customer_list
```

Supporting Files -- Coding Scripts

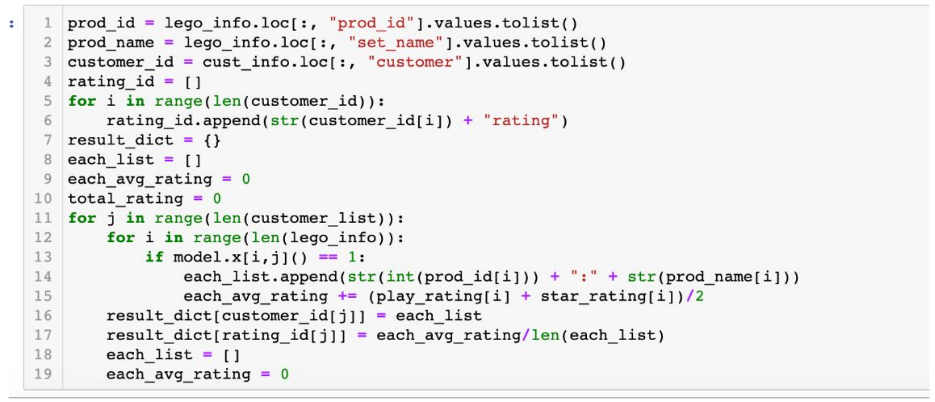
- Step 3: Build model



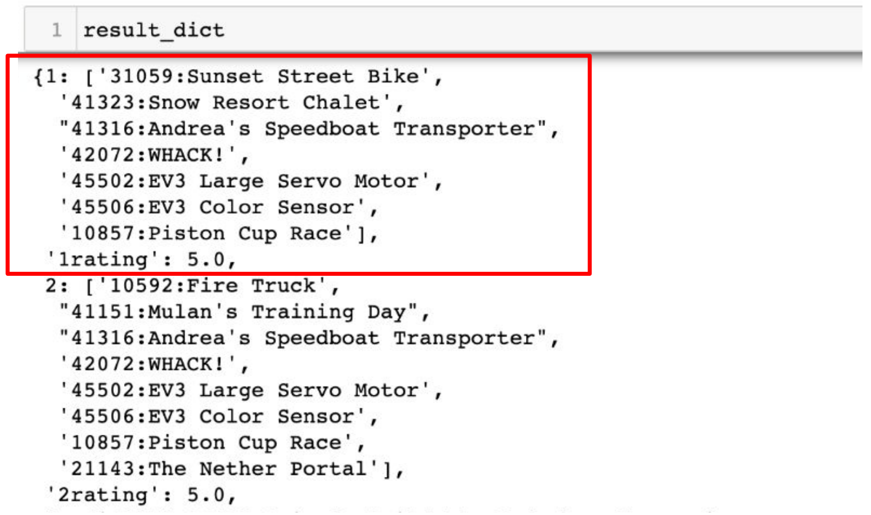
```python
model = ConcreteModel()
model.x = Var(range(len(lego_info)), range(len(customer_list)), domain = Binary)

```

```python
#inventory constraint
model.inventory = ConstraintList()
for i in range(len(lego_info)):
    model.inventory.add(expr=sum(model.x[i,j] for j in range(len(customer_list))) <= inventory_list[i])

#customer will not get item that they already have
model.newsample = ConstraintList()
for j in range(len(customer_list)):
    for i in range(len(lego_info)):
        model.newsample.add(expr=(model.x[i,j] + customer_list[j][i]) <= 1)

#the quantity of lego set should be the same as customer wanted
model.itemcount = ConstraintList()
for j in range(len(customer_list)):
    model.itemcount.add(expr = sum(model.x[i,j] for i in range(len(lego_info))) == quantity_list[j])

#the contraint of price
model.price = ConstraintList()
for j in range(len(customer_list)):
    model.price.add(expr = sum(model.x[i,j]*lego_price[i] for i in range(len(lego_info))) <= price_limit[j])

model.objective = Objective(expr = sum(model.x[i,j]*((play_rating[i]+star_rating[i])/2) for i in range(len(lego_inf
opt = SolverFactory('glpk')
results = opt.solve(model, tee=True)
```

- Step 4: Print out results



```python
prod_id = lego_info.loc[:, "prod_id"].values.tolist()
prod_name = lego_info.loc[:, "set_name"].values.tolist()
customer_id = cust_info.loc[:, "customer"].values.tolist()
rating_id = []
for i in range(len(customer_id)):
    rating_id.append(str(customer_id[i]) + "rating")
result_dict = {}
each_list = []
each_avg_rating = 0
total_rating = 0
for j in range(len(customer_list)):
    for i in range(len(lego_info)):
        if model.x[i,j]() == 1:
            each_list.append(str(int(prod_id[i])) + ":" + str(prod_name[i]))
            each_avg_rating += (play_rating[i] + star_rating[i])/2
    result_dict[customer_id[j]] = each_list
    result_dict[rating_id[j]] = each_avg_rating/len(each_list)
    each_list = []
    each_avg_rating = 0
```



```python
result_dict
```

```
{1: ['31059:Sunset Street Bike',
  '41323:Snow Resort Chalet',
  "41316:Andrea's Speedboat Transporter",
  '42072:WHACK!',
  '45502:EV3 Large Servo Motor',
  '45506:EV3 Color Sensor',
  '10857:Piston Cup Race'],
 '1rating': 5.0,
 2: ['10592:Fire Truck',
  "41151:Mulan's Training Day",
  "41316:Andrea's Speedboat Transporter",
  '42072:WHACK!',
  '45502:EV3 Large Servo Motor',
  '45506:EV3 Color Sensor',
  '10857:Piston Cup Race',
  '21143:The Nether Portal'],
 '2rating': 5.0,
```

Each result will include **customer id, LEGO set id, set name, and average rating (per customer)**

Data References

https://www.kaggle.com/rtatman/lego-database