```
#
# Purdue University Global
#
# IN402 - Modeling and Predictive Analysis
#
# Unit 8 Assignment / Module 5 Part 2 Competency Assessment
#
# Comparing Prescriptive and Predictive Analysis
#
# PyCharm Code
#

# Library imports

import sys

import pulp

# Ignoring warnings

if not sys.warnoptions:
    import warnings

warnings.simplefilter("ignore")

# Output Header

print('Unit 8 Assignment / Module 5 Part 2 Competency Assessment Output\n')

from datetime import datetime

print(datetime.now().strftime("%m/%d/%Y %H:%M:%S"), '\n')

# Instantiate the problem class

model = pulp.LpProblem("Cost_minimising_blending_problem", pulp.LpMinimize)

# Create decision variable lists

cookie_types = ['low-fat', 'regular']

ingredients = ['peanut butter', 'eggs', 'sugar']

# Using pulp's LpVariable object's dictionary functionality,
# provide tuple indices to use them as keys for the ing_weight
# dictionary of decision variables.

# create tuple indices

ing_weight = pulp.LpVariable.dicts("weight lbs",
    ((i, j) for i in cookie_types for j in ingredients),
    lowBound=0,
    cat='Continuous')

# Use lpSum vector calculation for the sub of a list of linear expressions.

# Calculate the sum of a list of linear expressions (objective function)

model+= (
    pulp.lpSum([
    4.32 * ing_weight[(i, 'peanut butter')]
    + 2.46 * ing_weight[(i, 'eggs')]
    + 1.86 * ing_weight[(i, 'sugar')]
    for i in cookie_types])
)

# Create an objective function / Add constraints

# Add constraints for 500 regular and 350 light patties at 0.05 of lbs.

model+= pulp.lpSum([ing_weight['low-fat', j] for j in ingredients])== 350 * 0.05

model+= pulp.lpSum([ing_weight['regular', j] for j in ingredients])== 500 * 0.05

# Low fat cookie has >= 40% peanut butter, regular >= 60%

model += ing_weight['low-fat', 'peanut butter'] >= (
    0.4 * pulp.lpSum([ing_weight['low-fat', j] for j in ingredients]))

model+= ing_weight['regular', 'peanut butter']>= (
    0.6 * pulp.lpSum([ing_weight['regular', j] for j in ingredients]))

# Cookies must be <= 25%

model += ing_weight['low-fat', 'sugar'] <= (
    0.25 * pulp.lpSum([ing_weight['low-fat', j] for j in ingredients]))

model+= ing_weight['regular', 'sugar']<= (
    0.25 * pulp.lpSum([ing_weight['regular', j] for j in ingredients]))

# You have already bought 30 lbs of peanut butter, 20 lbs of eggs and 17 lbs of sugar.

model+= pulp.lpSum([ing_weight[i, 'peanut butter'] for i in cookie_types]) <= 30

model+= pulp.lpSum([ing_weight[i, 'eggs'] for i in cookie_types]) <= 20

model+= pulp.lpSum([ing_weight[i, 'sugar'] for i in cookie_types]) <= 17

# We have at least 23 lbs of peanut butter

model+= pulp.lpSum([ing_weight[i, 'peanut butter'] for i in cookie_types]) >= 23

# Solve the problem / Show the model status

model.solve()

print("Model Status = ", pulp.LpStatus[model.status])

# ***** Status Codes *****
# OPTIMAL - Optimal solution exists and is found.
# INFEASIBLE - The problem has no feasible solution.
# UNBOUNDED - The cost function is unbounded.
# UNDEFINED - Feasible solution hasn't been found (but may exist).
# Not Solved - Is the default setting before a problem has been solved.

# Solve the problem - Show the model parameter results

for var in ing_weight:
    var_value = ing_weight[var].varValue
    print("The weight of {0} in {1} cookies is {2} lbs".format(var[1], var[0], var_value))

# Calculate the total cost for 500 regular and 350 low-fat cookies.

total_cost = pulp.value(model.objective)

print("The total cost is ${} for 350 low-fat cookies and 500 regular cookies". format( round(total_cost, 2)))
```