```
"""
In the videos, we saw the "diet problem". (The diet problem is one of the first large-scale optimization problems to be
studied in practice. Back in the 1930's and 40's, the Army wanted to meet the nutritional requirements of its soldiers
while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the
file diet.xls.

1.  Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum
daily nutrition constraints, and solve it using PuLP.  Turn in your code and the solution. (The optimal solution should
be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)
"""
from pulp import *
import pandas as pd

data = pd.read_excel("C:/Users/Admin/Desktop/MM/Homework 11/diet.xls",
                     sheet_name="Sheet1")
print(data.tail())
data = data[0:64]  # exclude bottom data

# intake restrictions
min_intake = [1500, 30, 20, 800, 130, 125, 60, 1000, 400, 700, 10]
max_intake = [2500, 240, 70, 2000, 450, 250, 100, 10000, 5000, 1500, 40]

# create foods dictionary
data = data.values.tolist()
foods = [x[0] for x in data]
food_dict = []
for i in range(3, 14):
    food_dict.append(dict([(x[0], float(x[i])) for x in data]))

# cost
cost = dict([(x[0], float(x[1])) for x in data])

# initiate optimization problem
prob = LpProblem('myProblem', LpMinimize)

# create variables
var_food = LpVariable.dicts("foods", foods, 0)
var_chosen = LpVariable.dicts("chosen", foods, 0, 1, LpBinary)
amount = LpVariable.dicts("amount", foods, 0)

# objective function - linear sum of costs (cost * food)
prob += lpSum([cost[food] * amount[food] for food in foods])

# constraints
for i in range(0, 10):  # nutrient constraint
    nutrients = pulp.lpSum([food_dict[i][food] * amount[food] for food
 in foods])
    prob += max_intake[i] >= nutrients
    prob += min_intake[i] <= nutrients

```

```
50 # optimize
51 prob.solve()
52 print('Solution:')
53 for var in prob.variables():
54     if var.varValue > 0:  # if solution for food value is more than 0
55         if str(var).find('chosen'):  # if food is chosen for solution
56             print(str(var.varValue) + " units of " + str(var))
57 print("Total cost of food = $%.2f" % value(prob.objective))
58
59
60 """
61 2.  Please add to your model the following constraints (which might
   require adding more variables) and solve the new
62 model:
63 a.  If a food is selected, then a minimum of 1/10 serving must be
   chosen. (Hint: now you will need two variables for
64 each food i: whether it is chosen, and how much is part of the diet.
   You'll also need to write a constraint to link
65 them.)
66 """
67 # initiate optimization for problem 1
68 prob1 = LpProblem('myProblem1', LpMinimize)
69
70 # objective function
71 prob1 += lpSum([cost[food] * amount[food] for food in foods])
72
73 # constraints
74 for i in range(0, 10):  # nutrient constraint
75     nutrients = pulp.lpSum([food_dict[i][food] * amount[food] for
   food in foods])
76     prob1 += max_intake[i] >= nutrients
77     prob1 += min_intake[i] <= nutrients
78
79 for food in foods:  # minimum food unit constraint
80     prob1 += var_food[food] >= 0.1 * var_chosen[food]
81
82 # optimize
83 prob1.solve()
84 print('Solution for Problem 1:')
85 for var in prob1.variables():
86     if var.varValue > 0:  # if solution for food value is more than 0
87         if str(var).find('chosen'):  # if food is chosen for solution
88             print(str(var.varValue) + " units of " + str(var))
89 print("Total cost of food = $%.2f" % value(prob1.objective))
90
91
92 """
93 b.  Many people dislike celery and frozen broccoli. So at most one,
   but not both, can be selected.
94 """
95 # initiate optimization for problem 2
96 prob2 = LpProblem('myProblem2', LpMinimize)
97
98 # objective function
99 prob2 += lpSum([cost[food] * amount[food] for food in foods])
100
101 # constraints
```

```
102 for i in range(0, 10):  # nutrient constraint
103     nutrients = pulp.lpSum([food_dict[i][food] * amount[food] for
    food in foods])
104     prob2 += max_intake[i] >= nutrients
105     prob2 += min_intake[i] <= nutrients
106 
107 for food in foods:  # minimum food unit constraint
108     prob2 += var_food[food] >= 0.1 * var_chosen[food]
109 
110 prob2 += var_chosen['Frozen Broccoli'] + var_chosen['Celery, Raw'
    ] <= 1  # at most one of the items constraint
111 
112 # optimize
113 prob2.solve()
114 print('Solution for Problem 2:')
115 for var in prob2.variables():
116     if var.varValue > 0:  # if solution for food value is more than 0
117         if str(var).find('chosen'):  # if food is chosen for solution
118             print(str(var.varValue) + " units of " + str(var))
119 print("Total cost of food = $%.2f" % value(prob2.objective))
120 
121 
122 """
123 c.  To get day-to-day variety in protein, at least 3 kinds of meat/
    poultry/fish/eggs must be selected. [If something is
124 ambiguous (e.g., should bean-and-bacon soup be considered meat?),
    just call it whatever you think is appropriate - I
125 want you to learn how to write this type of constraint, but I don't
    really care whether we agree on how to classify
126 foods!]
127 """
128 # initiate optimization for problem 2
129 prob3 = LpProblem('myProblem3', LpMinimize)
130 
131 # objective function
132 prob3 += lpSum([cost[food] * amount[food] for food in foods])
133 
134 # constraints
135 for i in range(0, 10):  # nutrient constraint
136     nutrients = pulp.lpSum([food_dict[i][food] * amount[food] for
    food in foods])
137     prob3 += max_intake[i] >= nutrients
138     prob3 += min_intake[i] <= nutrients
139 
140 for food in foods:  # minimum food unit constraint
141     prob3 += var_food[food] >= 0.1 * var_chosen[food]
142 
143 prob3 += var_chosen['Frozen Broccoli'] + var_chosen['Celery, Raw'
    ] <= 1  # at most one of the items constraint
144 
145 prob3 += var_chosen['Ham,Sliced,Extralean'] + var_chosen['Frankfurter
    , Beef'] + var_chosen['Hamburger W/Toppings'] \
146         + var_chosen['Hotdog, Plain'] + var_chosen['Scrambled Eggs'
    ] + var_chosen['Kielbasa,Prk'] \
147         + var_chosen['Poached Eggs'] + var_chosen['Pork'] +
    var_chosen['Roasted Chicken'] \
148         + var_chosen['Sardines in Oil'] + var_chosen['White Tuna in
```

```
148 Water'] \
149         >= 3  # at least 3 kinds of meat/poultry/fish/eggs constraint
150
151 # optimize
152 prob3.solve()
153 print('Solution for Problem 3:')
154 for var in prob3.variables():
155     if var.varValue > 0:  # if solution for food value is more than 0
156         if str(var).find('chosen'):  # if food is chosen for solution
157             print(str(var.varValue) + " units of " + str(var))
158 print("Total cost of food = $%.2f" % value(prob3.objective))
```