Question 15.2

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!)

   Formulated optimization model minimal cost is $4.34.

   Constraints are:
   - maximum and minimum daily values of each nutrient

   Optimization Solution:
   52.64371 units of foods_Celery,_Raw
   0.25960653 units of foods_Frozen_Broccoli
   63.988506 units of foods_Lettuce,Iceberg,Raw
   2.2929389 units of foods_Oranges
   0.14184397 units of foods_Poached_Eggs
   13.869322 units of foods_Popcorn, Air_Popped

2. Please add to your model the following constraints (which might require adding more variables) and solve the new model:
   a. If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for 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 them.)
   b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.
   c. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. [If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don't really care whether we agree on how to classify foods!]

Formulated optimization model minimal cost is $4.51.

Constraints are:
- maximum and minimum daily values of each nutrient
- chosen foods bounded between .1 and M(large constant)
- only one of broccoli and celery can be in the optimal diet
- at least three proteins must be selected in the optimal diet

Optimization Solution:

42.399358 units of foods_Celery,_Raw
0.1 units of foods_Kielbasa,Prk
82.802586 units of foods_Lettuce,Iceberg,Raw
3.0771841 units of foods_Oranges
1.9429716 units of foods_Peanut_Butter
0.1 units of foods_Poached_Eggs
13.223294 units of foods_Popcorn,Air_Popped
0.1 units of foods_Scrambled_Eggs

If you want to see what a more full-sized problem would look like, try solving your models for the file `diet_large.xls`, which is a low-cholesterol diet model (rather than minimizing cost, the goal is to minimize cholesterol intake). I don't know anyone who'd want to eat this diet – the optimal solution includes dried chrysanthemum garland, raw beluga whale flipper, freeze-dried parsley, etc. – which shows why it's necessary to add additional constraints beyond the basic ones we saw in the video!

[**Note**: there are many optimal solutions, all with zero cholesterol, so you might get a different one. It probably won't be much more appetizing than mine.]

main.py

```python
# import libraries
from pulp import *
import pandas as pd

# load the diet data
df = pd.read_excel(open(
            '/Users/xiaofanjiao/Desktop/diet.xls', 'rb'),
            sheet_name='Sheet1'
            )

# clean data - take first 64 rows not including bottom data
data = df[0:64]

# convert to list "list within a list"
data = data.values.tolist()

# create master foods dictionary
foods = [x[0] for x in data]
calories = dict([(x[0], float(x[3])) for x in data])
cholesterol = dict([(x[0], float(x[4])) for x in data])
totalFat = dict([(x[0], float(x[5])) for x in data])
sodium = dict([(x[0], float(x[6])) for x in data])
carbs = dict([(x[0], float(x[7])) for x in data])
fiber = dict([(x[0], float(x[8])) for x in data])
protien = dict([(x[0], float(x[9])) for x in data])
vitaminA = dict([(x[0], float(x[10])) for x in data])
vitaminC = dict([(x[0], float(x[11])) for x in data])
calcium = dict([(x[0], float(x[12])) for x in data])
iron = dict([(x[0], float(x[13])) for x in data])

# create list for mins and maxes (all foods)
amin = [1500, 30, 20, 800, 130, 125, 60, 1000, 400, 700, 10]
amax = [2500, 240, 70, 2000, 450, 250, 100, 10000, 5000, 1500, 40]

# append collection of contraints for each column
B = []
for j in range(0, 11):
    B.append(dict([(x[0], float(x[j + 3])) for x in data]))

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

# create the optimization problem framework - minimization problem
problem1 = LpProblem('PuLPTutorial', LpMinimize)

# define the variables - continous
foodVars = LpVariable.dicts("foods", foods, 0)

# define the variables - binary
chosenVars = LpVariable.dicts("Chosen", foods, 0, 1, "Binary")

# dictionary of lp variables
x = LpVariable.dicts("x", foods, 0)
```

```
54
55  # define the objective function
56  problem1 += lpSum([cost[f] * foodVars[f] for f in foods])
57
58  # add constraints for all foods
59  for i in range(0, 11):
60      dot_B_x = pulp.lpSum([B[i][j] * foodVars[j] for j in foods])
61      condition1 = amin[i] <= + dot_B_x
62      problem1 += condition1
63
64  for i in range(0, 11):
65      dot_B_x = pulp.lpSum([B[i][j] * foodVars[j] for j in foods])
66      condition2 = amax[i] >= + dot_B_x
67      problem1 += condition2
68
69  # solve the optimization problem!
70  problem1.solve()
71
72  # print the foods of the optimal diet
73  print('Optimization Solution:')
74  for var in problem1.variables():
75      if var.varValue > 0:
76          if str(var).find('Chosen'):
77              print(str(var.varValue) + " units of " + str(var))
78
79  # print the costs of the optimal diet
80  print("Total cost of food = $%.2f" % value(problem1.objective))
81
```

part 2.py

```
# load the libraries needed
# !pip install pulp

from pulp import *
import pandas as pd

# load the diet data
# load the diet data
df = pd.read_excel(open(
            '/Users/xiaofanjiao/Desktop/diet.xls', 'rb'),
            sheet_name='Sheet1'
            )

# clean data - take first 64 rows not including bottom data
data = df[0:64]

# convert to list "list within a list"
data = data.values.tolist()

# create master foods dictionary
foods = [x[0] for x in data]
calories = dict([(x[0], float(x[3])) for x in data])
cholesterol = dict([(x[0], float(x[4])) for x in data])
totalFat = dict([(x[0], float(x[5])) for x in data])
sodium = dict([(x[0], float(x[6])) for x in data])
carbs = dict([(x[0], float(x[7])) for x in data])
fiber = dict([(x[0], float(x[8])) for x in data])
protien = dict([(x[0], float(x[9])) for x in data])
vitaminA = dict([(x[0], float(x[10])) for x in data])
vitaminC = dict([(x[0], float(x[11])) for x in data])
calcium = dict([(x[0], float(x[12])) for x in data])
iron = dict([(x[0], float(x[13])) for x in data])

# create list for mins and maxes (all foods)
amin = [1500, 30, 20, 800, 130, 125, 60, 1000, 400, 700, 10]
amax = [2500, 240, 70, 2000, 450, 250, 100, 10000, 5000, 1500, 40]

# append collection of contraints for each column
B = []
for j in range(0, 11):
    B.append(dict([(x[0], float(x[j + 3])) for x in data]))

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

# create the optimization problem framework - minimization problem
problem2 = LpProblem('PuLPTutorial', LpMinimize)

# define the variables - continous
foodVars = LpVariable.dicts("foods", foods, 0)

# define the variables - binary
chosenVars = LpVariable.dicts("Chosen", foods, 0, 1, "Binary")
```

```
54
55  # dictionary of lp variables
56  x = LpVariable.dicts("x", foods, 0)
57
58  # define the objective function
59  problem2 += lpSum([cost[f] * foodVars[f] for f in foods])
60
61  # add contraints amount greater than .1 or less than large amount - if chosen
62  for f in foods:
63      problem2 += foodVars[f] <= 10000 * chosenVars[f]
64      problem2 += foodVars[f] >= .1 * chosenVars[f]
65
66  # add constraints for all foods
67  for i in range(0, 11):
68      dot_B_x = pulp.lpSum([B[i][j] * foodVars[j] for j in foods])
69      condition1 = amin[i] <= + dot_B_x
70      problem2 += condition1
71
72  for i in range(0, 11):
73      dot_B_x = pulp.lpSum([B[i][j] * foodVars[j] for j in foods])
74      condition2 = amax[i] >= + dot_B_x
75      problem2 += condition2
76
77  # add contraints to eat at most one of a group of foods
78  problem2 += chosenVars['Frozen Broccoli'] + \
79              chosenVars['Celery, Raw'] <= 1, 'At most one Broccoli / Celery'
80
81  # add contraints that says we require to eat as least 1 from group of food
82  problem2 += chosenVars['Roasted Chicken'] + chosenVars['Poached Eggs'] + \
83              chosenVars['Scrambled Eggs'] + chosenVars['Frankfurter, Beef'] + \
84              chosenVars['Kielbasa,Prk'] + chosenVars['Hamburger W/Toppings'] + \
85              chosenVars['Hotdog, Plain'] + chosenVars['Pork'] + \
86              chosenVars['Bologna,Turkey'] + chosenVars['Ham,Sliced,Extralean'] + \
87              chosenVars['White Tuna in Water'] \
88              >= 3, 'At least three proteins'
89
90  # solve the optimization problem!
91  problem2.solve()
92
93  # print the foods of the optimal diet
94  print('Optimization Solution:')
95  for var in problem2.variables():
96      if var.varValue > 0:
97          if str(var).find('Chosen'):
98              print(str(var.varValue) + " units of " + str(var))
99
100 # print the costs of the optimal diet
101 print("Total cost of food = $%.2f" % value(problem2.objective))
```