MSDS650 Week 8 Integer Programming Assignment - Nathan Worsham

I chose for this assignment to do the Python "whiskas" problem. I first needed to import modeling functions from pulp:

```
>>> from pulp import *
```

Now setting a variable with `LpProblem` function, giving it a name and setting the paramater to a minimization problem:

```
>>> prob = LpProblem("The Whiskas Problem",LpMinimize)
>>> prob
The Whiskas Problem:
MINIMIZE
None
VARIABLES
```

Next the 2 variables representing chicken (x1) and beef (x2) are to be created. Using the `LpVariable` command, several options are set. The first option set is the name of the variable–chicken or beef. The second is the lower bound of the variable, in this case 0 for both. The third variable is the upper bound, and its default value is `None`–which means positive infinity according to the help file–but to get to the fourth option, which they only set for chicken, you have to specify it as a parameter in between options cannot be left blank. Only on chicken are they setting the fourth option to `LpInteger` whereas `LpContinous` is the default and–confirmed later–what is left set on the beef variable (honestly I'm not sure here why both are not set to integer but since there are only 2 variables and combined they have to equal 100%, if one is forced to be an integer, then so will the other).

```
>>> x1=LpVariable("ChickenPercent",0,None,LpInteger)
>>> x1
ChickenPercent
>>> x2=LpVariable("BeefPercent",0)
>>> x2
BeefPercent
```

Now the `prob` variable is given problem data, the objective function, followed by an explanation of what the function is. Chicken costs $0.013 per gram and beef $0.008.

```
>>> prob += 0.013*x1 + 0.008*x2, "Total Cost of Ingredients per can"
>>> prob
The Whiskas Problem:
MINIMIZE
0.008*BeefPercent + 0.013*ChickenPercent + 0.0
VARIABLES
BeefPercent Continuous
0 <= ChickenPercent Integer
```

Constraints are added to the problem. First the amount of chicken and beef in a can, cannot exceed 100%. Next there is a protein requirement on the can of a minimum (or greater than or equal to) of 8% protein, chicken provides 0.1 protein per gram and beef provides 0.2. There is a minimum fat requirement of 6%, chicken provides 0.08 fat per gram and beef 0.1. Fibre requirements on the can state there is a maximum of 2% per can, chicken has 0.001 fibre per gram and beef 0.005. Last there is a salt maximum of 0.4%, chicken has 0.002 grams of salt per gram and beef 0.005.

```
>>> prob += x1 + x2 == 100, "PercentagesSum"
>>> prob
The Whiskas Problem:
MINIMIZE
0.008*BeefPercent + 0.013*ChickenPercent + 0.0
```

```
SUBJECT TO
PercentagesSum: BeefPercent + ChickenPercent = 100

VARIABLES
BeefPercent Continuous
0 <= ChickenPercent Integer

>>> prob += 0.100*x1 + 0.200*x2 >= 8.0, "ProteinRequirement"
>>> prob
The Whiskas Problem:
MINIMIZE
0.008*BeefPercent + 0.013*ChickenPercent + 0.0
SUBJECT TO
PercentagesSum: BeefPercent + ChickenPercent = 100

ProteinRequirement: 0.2 BeefPercent + 0.1 ChickenPercent >= 8

VARIABLES
BeefPercent Continuous
0 <= ChickenPercent Integer

>>> prob += 0.080*x1 + 0.100*x2 >= 6.0, "FatRequirement"
>>> prob += 0.001*x1 + 0.005*x2 <= 2.0, "FibreRequirement"
>>> prob += 0.002*x1 + 0.005*x2 <= 0.4, "SaltRequirement"
>>> prob
The Whiskas Problem:
MINIMIZE
0.008*BeefPercent + 0.013*ChickenPercent + 0.0
SUBJECT TO
PercentagesSum: BeefPercent + ChickenPercent = 100

ProteinRequirement: 0.2 BeefPercent + 0.1 ChickenPercent >= 8

FatRequirement: 0.1 BeefPercent + 0.08 ChickenPercent >= 6

FibreRequirement: 0.005 BeefPercent + 0.001 ChickenPercent <= 2

SaltRequirement: 0.005 BeefPercent + 0.002 ChickenPercent <= 0.4

VARIABLES
BeefPercent Continuous
0 <= ChickenPercent Integer
```

The excercise has us write out to a file the problem data. It really doesn't look much different than me just typing the variable as I did above. This is likely just meant for if you weren't using IDLE to go through this excercise step by step I suppose.

```
>>> prob.writeLP("WhiskasModel.lp")
b18608:week8 worshamn$ cat WhiskasModel.lp
\* The Whiskas Problem *\
Minimize
Total_Cost_of_Ingredients_per_can: 0.008 BeefPercent + 0.013 ChickenPercent
Subject To
FatRequirement: 0.1 BeefPercent + 0.08 ChickenPercent >= 6
FibreRequirement: 0.005 BeefPercent + 0.001 ChickenPercent <= 2
```

```
PercentagesSum: BeefPercent + ChickenPercent = 100
ProteinRequirement: 0.2 BeefPercent + 0.1 ChickenPercent >= 8
SaltRequirement: 0.005 BeefPercent + 0.002 ChickenPercent <= 0.4
Bounds
0 <= ChickenPercent
Generals
ChickenPercent
End
```

Let Lp decide which solver to use by not specifiying one in the `obj.solve()` command. Then display the status of the solution–which could be "Not Solved" or "Infeasible"–by calling the LpStatus dictionary and the key of `obj.status`.

```
>>> prob.solve()
1
>>> print("Status:", LpStatus[prob.status])
('Status:', 'Optimal')
```

Now show the optimum values for each variable and the total cost of ingredients per can. This is accomplished with `.name` and `.varValue` for the variables, and `value(prob.objective)` for the total cost.

```
>>> for v in prob.variables():
...   print(v.name, "=", v.varValue)
...
('BeefPercent', '=', 66.0)
('ChickenPercent', '=', 34.0)
>>> print("Total Cost of Ingredients per can = ", value(prob.objective))
('Total Cost of Ingredients per can = ', 0.97)
```

The exercise comes up with a solution of 33.33% for chicken, 66.67% for beef and 96 cents for the total cost, whereas I received 34%, 66%, and 97 cents. It would seem that the author possibly didn't use the integer setting when gathering the results. In that exercise, the author was only using 2 of the variables to make the problem easier to explain, but now the exercise repeats the same steps from above but with all variables. Starting with creating a list of variables and a dictionary of constraints. This is likely so that not everything has to be typed out and can be iterated to make the code more elegant.

```
>>> Ingredients = ['CHICKEN', 'BEEF', 'MUTTON', 'RICE', 'WHEAT', 'GEL']
>>> Ingredients
['CHICKEN', 'BEEF', 'MUTTON', 'RICE', 'WHEAT', 'GEL']
>>> costs = {'CHICKEN': 0.013,'BEEF': 0.008,'MUTTON': 0.010,'RICE': 0.002,'WHEAT': 0.005,
'GEL': 0.001}
>>> costs
{'MUTTON': 0.01, 'WHEAT': 0.005, 'BEEF': 0.008, 'CHICKEN': 0.013, 'RICE': 0.002,
'GEL': 0.001}
>>> proteinPercent = {'CHICKEN': 0.100,'BEEF': 0.200,'MUTTON': 0.150,'RICE': 0.000,
'WHEAT': 0.040,'GEL': 0.000}
>>> proteinPercent
{'MUTTON': 0.15, 'WHEAT': 0.04, 'BEEF': 0.2, 'CHICKEN': 0.1, 'RICE': 0.0, 'GEL': 0.0}
>>> fatPercent = {'CHICKEN': 0.080,'BEEF': 0.100,'MUTTON': 0.110,'RICE': 0.010,'WHEAT':
 0.010,'GEL': 0.000}
>>> fatPercent
{'MUTTON': 0.11, 'WHEAT': 0.01, 'BEEF': 0.1, 'CHICKEN': 0.08, 'RICE': 0.01, 'GEL': 0.0}
>>> fibrePercent = {'CHICKEN': 0.001,'BEEF': 0.005,'MUTTON': 0.003,'RICE': 0.100,'WHEAT':
 0.150,'GEL': 0.000}
>>> fibrePercent
{'MUTTON': 0.003, 'WHEAT': 0.15, 'BEEF': 0.005, 'CHICKEN': 0.001, 'RICE': 0.1, 'GEL': 0.0}
```

```
>>> saltPercent = {'CHICKEN': 0.002,'BEEF': 0.005,'MUTTON': 0.007,'RICE': 0.002,'WHEAT':
 0.008,'GEL': 0.000}
>>> saltPercent
{'MUTTON': 0.007, 'WHEAT': 0.008, 'BEEF': 0.005, 'CHICKEN': 0.002, 'RICE': 0.002, 'GEL': 0.0}
```

Again the `prob` variable is used to contain the LpProblem function which again is set to a minimization problem.

```
>>> prob = LpProblem("The Whiskas Problem", LpMinimize)
>>> prob
The Whiskas Problem:
MINIMIZE
None
VARIABLES
```

The `LpVariable` function is used again to set the lower and upper bounds of the variables but this time a `.dicts` is given to both use and create a dictionary. The author this time did not specify the `LpInteger` option but assuming that since this assignment falls under the "Integer Programming Assignment", I decided it was best to add that option.

```
>> ingredient_vars = LpVariable.dicts("Ingr",Ingredients,0,None,LpInteger)
>>> ingredient_vars
{'MUTTON': Ingr_MUTTON, 'WHEAT': Ingr_WHEAT, 'BEEF': Ingr_BEEF, 'CHICKEN':
Ingr_CHICKEN, 'RICE': Ingr_RICE, 'GEL': Ingr_GEL}
```

As previously presumed, since the costs and ingredients are stored in variables that can be iterated through, it is much easier to get this done with one line of code instead of 6. The iterations–using python's list comprehension–are continued, setting the remaining constraints required by what is written on the can.

```
>>> prob += lpSum([costs[i]*ingredient_vars[i] for i in Ingredients]), "Total Cost of
Ingredients per can"
>>> prob
The Whiskas Problem:
MINIMIZE
0.008*Ingr_BEEF + 0.013*Ingr_CHICKEN + 0.001*Ingr_GEL + 0.01*Ingr_MUTTON +
0.002*Ingr_RICE + 0.005*Ingr_WHEAT + 0.0
VARIABLES
0 <= Ingr_BEEF Integer
0 <= Ingr_CHICKEN Integer
0 <= Ingr_GEL Integer
0 <= Ingr_MUTTON Integer
0 <= Ingr_RICE Integer
0 <= Ingr_WHEAT Integer

>>> prob += lpSum([ingredient_vars[i] for i in Ingredients]) == 100, "PercentagesSum"
>>> prob += lpSum([proteinPercent[i] * ingredient_vars[i] for i in Ingredients]) >= 8.0,
"ProteinRequirement"
>>> prob += lpSum([fatPercent[i] * ingredient_vars[i] for i in Ingredients]) >= 6.0,
 "FatRequirement"
>>> prob += lpSum([fibrePercent[i] * ingredient_vars[i] for i in Ingredients]) <= 2.0,
"FibreRequirement"
>>> prob += lpSum([saltPercent[i] * ingredient_vars[i] for i in Ingredients]) <= 0.4,
"SaltRequirement"
>>> prob
The Whiskas Problem:
MINIMIZE
```

```
0.008*Ingr_BEEF + 0.013*Ingr_CHICKEN + 0.001*Ingr_GEL + 0.01*Ingr_MUTTON +
0.002*Ingr_RICE + 0.005*Ingr_WHEAT + 0.0
SUBJECT TO
PercentagesSum: Ingr_BEEF + Ingr_CHICKEN + Ingr_GEL + Ingr_MUTTON + Ingr_RICE
 + Ingr_WHEAT = 100

ProteinRequirement: 0.2 Ingr_BEEF + 0.1 Ingr_CHICKEN + 0.15 Ingr_MUTTON
 + 0.04 Ingr_WHEAT >= 8

FatRequirement: 0.1 Ingr_BEEF + 0.08 Ingr_CHICKEN + 0.11 Ingr_MUTTON
 + 0.01 Ingr_RICE + 0.01 Ingr_WHEAT >= 6

FibreRequirement: 0.005 Ingr_BEEF + 0.001 Ingr_CHICKEN + 0.003 Ingr_MUTTON
 + 0.1 Ingr_RICE + 0.15 Ingr_WHEAT <= 2

SaltRequirement: 0.005 Ingr_BEEF + 0.002 Ingr_CHICKEN + 0.007 Ingr_MUTTON
 + 0.002 Ingr_RICE + 0.008 Ingr_WHEAT <= 0.4

VARIABLES
0 <= Ingr_BEEF Integer
0 <= Ingr_CHICKEN Integer
0 <= Ingr_GEL Integer
0 <= Ingr_MUTTON Integer
0 <= Ingr_RICE Integer
0 <= Ingr_WHEAT Integer
```

Running through the remaining steps which are the same as the simplified excercise.

```
>>> prob.writeLP("WhiskasFullModel.lp")
b18608:week8 worshamn$ cat WhiskasFullModel.lp
\* The Whiskas Problem *\
Minimize
Total_Cost_of_Ingredients_per_can: 0.008 Ingr_BEEF + 0.013 Ingr_CHICKEN
 + 0.001 Ingr_GEL + 0.01 Ingr_MUTTON + 0.002 Ingr_RICE + 0.005 Ingr_WHEAT
Subject To
FatRequirement: 0.1 Ingr_BEEF + 0.08 Ingr_CHICKEN + 0.11 Ingr_MUTTON
 + 0.01 Ingr_RICE + 0.01 Ingr_WHEAT >= 6
FibreRequirement: 0.005 Ingr_BEEF + 0.001 Ingr_CHICKEN + 0.003 Ingr_MUTTON
 + 0.1 Ingr_RICE + 0.15 Ingr_WHEAT <= 2
PercentagesSum: Ingr_BEEF + Ingr_CHICKEN + Ingr_GEL + Ingr_MUTTON + Ingr_RICE
 + Ingr_WHEAT = 100
ProteinRequirement: 0.2 Ingr_BEEF + 0.1 Ingr_CHICKEN + 0.15 Ingr_MUTTON
 + 0.04 Ingr_WHEAT >= 8
SaltRequirement: 0.005 Ingr_BEEF + 0.002 Ingr_CHICKEN + 0.007 Ingr_MUTTON
 + 0.002 Ingr_RICE + 0.008 Ingr_WHEAT <= 0.4
Bounds
0 <= Ingr_BEEF
0 <= Ingr_CHICKEN
0 <= Ingr_GEL
0 <= Ingr_MUTTON
0 <= Ingr_RICE
0 <= Ingr_WHEAT
Generals
Ingr_BEEF
Ingr_CHICKEN
```

```
Ingr_GEL
Ingr_MUTTON
Ingr_RICE
Ingr_WHEAT
End
>>> prob.solve()
1
>>> print("Status:", LpStatus[prob.status])
('Status:', 'Optimal')
>>> for v in prob.variables():
...   print(v.name, "=", v.varValue)
...
('Ingr_BEEF', '=', 60.0)
('Ingr_CHICKEN', '=', 0.0)
('Ingr_GEL', '=', 40.0)
('Ingr_MUTTON', '=', 0.0)
('Ingr_RICE', '=', 0.0)
('Ingr_WHEAT', '=', 0.0)
>>> print("Total Cost of Ingredients per can = ", value(prob.objective))
('Total Cost of Ingredients per can = ', 0.52)
```

The optimal solution–which matches what the author said it should come out to–is 60% beef and 40% gel, resulting in a total cost of 52 cents. Though the result seems a bit odd and not terrible appetizing because of 40% of gel, the equation did what it was asked to do. In real life though this may not fly, but then again it may considering how some corporations are which makes me shudder to think what they probably already do with some foods for human consumption. The only thing I could think of that could be an issue would be if they were trying to sell this particular line as "Chicken" when in fact it contains no chicken.

References

pythonhosted.org, 2009. Retrieved from https://pythonhosted.org/PuLP/CaseStudies/a_blending_problem.html