Where we are headed: LP Implementation in PuLP/Python with Dictionaries and AMPLY

```python
# -*- coding: utf-8 -*-
"""
hotTubAMPLY3.py - last updated 1/13/20
Uses dictionaries and AMPLY (AMPLY is built in to PuLP)
"""

from pulp import *

# Define some file names for use
DAT_FILE = 'blue.dat'  # contains LP problem parameter data
LP_FILE = 'hotTubLP.txt'    # stores formulated model

# Set up the AMPLY data structure
data = Amply("""
set products;
set resources;
param profit{products};
param avail{resources};
param req {resources, products};
""")
# Load the AMPLY data stored in the file DAT_FILE
data.load_file(open(DAT_FILE))

# Create the 'hotTub_model' variable to contain the problem data
hotTub_model = LpProblem("The Hot Tub Problem",LpMaximize)

# Create a dictionary of PuLP variables with keys being the various hot tubs
x = LpVariable.dicts('x', data.products, 0)

# Add objective
hotTub_model += lpSum(data.profit[j]*x[j] for j in data.products), \
    "Total Profit"

# Loop to add each resource constraint
for i in data.resources:
    hotTub_model += \
        lpSum(data.req[i,j]*x[j] for j in data.products) <= data.avail[i], i

# Print model to console and to a textfile
print(hotTub_model)
hotTub_model.writeLP(LP_FILE)

# Solve the problem
result = hotTub_model.solve()
# 0: 'Not Solved', 1: 'Optimal',-1:'Infeasible',
# -2:'Unbounded',-3:'Undefined'

# Print solution information: status, dec var and obj values
print("Status:", LpStatus[result])
for variable in hotTub_model.variables():
        print(variable,'=',value(variable))
print("Total Profit: ",value(hotTub_model.objective))
```

blue.dat

```
set products := Aqua Hydro;

set resources := pumps labor tubing;

param       profit :=
    Aqua    350
    Hydro   300 ;

param       avail  :=
    pumps   200
    labor   1566
    tubing  2880  ;

param   req  :
            Aqua    Hydro :=
    pumps   1       1
    labor   9       6
    tubing  12      16  ;
```

PuLP LpProblem object attributes

I highlighted a few. You should highlight more as you become familiar with useful ones.

Instance Variables

Key	Type	Size	Value
_variable_ids	dict	2	{1696903713344:LpVariable, 1696903713176:LpVariable}
_variables	list	2	[LpVariable, LpVariable]
constraints	OrderedDict	3	OrderedDict object of collections module
dummyVar	NoneType	1	NoneType object of builtins module
initialValues	dict	0	{}
lastUnused	int	1	0
modifiedConstraints	list	3	[LpConstraint, LpConstraint, LpConstraint]
modifiedVariables	list	0	[]
name	str	1	Hot Tub LP
noOverlap	int	1	1
objective	pulp.LpAffineExpression	2	LpAffineExpression object of pulp.pulp module
resolveOK	bool	1	False
sense	int	1	-1
solutionTime	float	1	0.16741438420089594
solver	solvers.PULP_CBC_CMD	1	PULP_CBC_CMD object of pulp.solvers module
sos1	dict	0	{}
sos2	dict	0	{}
status	int	1	1

Object methods

['__class__',
'__delattr__',
'__dir__',
'__eq__',
'__format__',
'__ge__',
'__getattribute__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__init__',
'__le__',
'__lt__',
'__ne__',
'__new__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__setattr__',
'__setstate__',
'__sizeof__',
'__str__',
'__subclasshook__',
'add',
'addConstraint',
'addVariable',
'addVariables',
'assignConsPi',
'assignConsSlack',
'assignVarsDj',
'assignVarsVals',
'coefficients',
'copy',
'deepcopy',
'extend',
'fixObjective',
'getSense',
'get_dummyVar',
'infeasibilityGap',
'isMIP',
'normalisedNames',
'numConstraints',
'numVariables',
'resolve',
'restoreObjective',
'roundSolution',
'sequentialSolve',
'setInitial',
'setObjective',
'setSolver',
'solve',
'unusedConstraintName',
'valid',
'variables',
'variablesDict',
'writeLP',
'writeMPS']