Assignment 06 (DNSC 6211)

This assignment will require you to use the **`pulp`** package.
Part 1: This part will involve understanding how to set up an LP problem using PuLP
Your program should

1. Read a json file (that contains the LP problem),
2. Create and solve the LP problem
3. Provide the output depending on the options provided by the user
4. The program output needs to be either brief (-b), detailed (-d) or the default
    - the **detailed** output contains
        - the optimal value
        - the decision variables and their values
    - the **brief** output mentions the optimal value only
    - the **default** output is a same as the brief output, except that it is part of a sentence
5. The program should have the option (-o) to send the output (**verbose** only) to a file with the name of the file provided by the user.

You are required to use a module file and use <u>at least three functions</u> for input (provided below), processing and output. They are

```
def readLPData(fn):
       """
       Input = name of JSON file
       Returns = JSON object
       """
       import json
       try:
       with open(fn, 'r') as f:
              LPdata = json.load(f)
       return LPdata
       except:
       return None
```

```
def createAndSolveLP(LPData):
"""
Input = JSON object
Returns = a dictionary with 2 elements
   The first element is the key value pair of the objective function and the value of
the
   objective function and the second element is a dictionary whose key value pair is
the
   decision variables and their values
"""
```

```
def writeFile(filename):
"""
Input = name of output file
Output = text file with the detailed output or error message if the write
       operation was unsuccessful
```

" " "

The next two pages show you what the structure of the JSON files. The structure should be self-explanatory. This is an example of how I have chosen to represent an LP problem.

```
{
    "name": "Intro LP",
    "variables":["TV", "Radio"],
    "objCoeffs":[7,5],
    "objective":"MAX",
    "LHS":{"c1":[4,3],
           "c2":[2,1]
        },
    "conditions":{"c1":"<=",
                  "c2":"<="
        },
    "RHS":{"c1":240,
           "c2":100
        }
}
```

Data file 1: introLP.jso n

```
{
  "name": "Intro LP",
  "variables": [
        "TV",
        "Radio"
  ],
  "objCoeffs": [
        7,
        5
  ],
  "objective": "MAX",
  "LHS": {
        "c1": [
        4,
        3
        ],
        "c2": [
        2,
        1
        ]
  },
  "conditions": {
        "c1": "<=",
        "c2": "<="
  },
  "RHS": {
        "c1": 240,
        "c2": 100
  }
}
```

```
{
    "name": "Whiskas",
    "variables":["chicken", "beef", "mutton", "rice", "wheat", "gel"],
    "objCoeffs": [0.013, 0.008, 0.010, 0.002, 0.005, 0.001],
    "objective": "MIN",
    "LHS": {
        "percentage": [1, 1, 1, 1, 1, 1],
        "protein": [0.100, 0.200, 0.150, 0.000, 0.040, 0.000],
        "fat": [0.080, 0.100, 0.110, 0.010, 0.010, 0.000],
        "fiber": [0.001, 0.005, 0.003, 0.100, 0.150, 0.000],
        "salt": [0.002, 0.005, 0.007, 0.002, 0.008, 0.000]
          },
    "conditions": {
        "percentage": "==",
        "protein": ">=",
        "fat": ">=",
        "fiber": "<=",
        "salt": "<="
          },
    "RHS": {
        "percentage": 100,
        "protein": 8.0,
        "fat": 6.0,
        "fiber": 2.0,
        "salt": 0.4
          }
}
```

Data file 2: whiskas.json

```
{
  "name": "Whiskas",
  "variables": [
        "chicken",
        "beef",
        "mutton",
        "rice",
        "wheat",
        "gel"
  ],
  "objCoeffs": [
        0.013,
        0.008,
        0.01,
        0.002,
        0.005,
        0.001
  ],
  "objective": "MIN",
  "LHS": {
        "percentage": [
        1.00,
        1.00,
        1.00,
        1.00,
        1.00,
        1.00
        ],
        "protein": [
        0.10,
        0.20,
        0.15,
        0.00,
        0.04,
        0.00
        ],
        "fat": [
        0.08,
        0.10,
        0.11,
        0.01,
        0.01,
        0.00
        ],
        "fiber": [
        0.001,
        0.005,
        0.003,
        0.100,
        0.150,
        0.000
        ],
        "salt": [
        0.002,
        0.005,
        0.007,
        0.002,
        0.008,
        0.000
        ]
  },
  "conditions": {
        "percentage": "==",
        "protein": ">=",
        "fat": ">=",
        "fiber": "<=",
        "salt": "<="
  },
  "RHS": {
        "percentage": 100,
        "protein": 8,
        "fat": 6,
        "fiber": 2,
        "salt": 0.4
  }
}
```

Sample outputs are shown below.

```
$ python A04_gwid.py whiskas.json
The optimal value is 0.520
```

```
$ python A04_gwid.py whiskas.json -b
0.52
```

```
$ python A04_gwid.py whiskas.json -d
The mimimum value is 0.520
The decision variables and their values are:
mutton: 0.000
wheat: 0.000
beef: 60.000
chicken: 0.000
rice: 0.000
gel: 40.000
```

```
$ python A04_gwid.py whiskas.json -d -o myout.txt
The mimimum value is 0.520
The decision variables and their values are:
mutton: 0.000
wheat: 0.000
beef: 60.000
chicken: 0.000
rice: 0.000
gel: 40.000
Output being sent to myout.txt
Output written to myout.txt
```

Part 2: This part will require you to use the **`MySQL`** and **`pymongo`** modules.

Your program should do the following

- Read all `json` files (that contains LP problems) in the current directory[1] and store them in a `pymongo` database. This should be done using a function called **`readAndStore()`**. You don't need to assign your own unique key to the **`"_id"`** field – it is done automatically. You can assign another field to be a unique key if you want to[2].
- Then the program should read the LP problems from the `pymongo` database, create and solve them (using **`PuLP`**). The creating and solving of the LP problem is available to you from last week. If the "optimal" solution for a problem cannot be found the program should record it as the string "NA".
- Lastly, the program should provide the output depending on the options given by the user. The program output needs to be sent to either a **text file (-t)**, a table in a **database (-d)**, or to the screen (**default**):
    - the **text file** option requires the problem name and the optimal value to be written to a separate line for each LP problem to a file whose name is provided by the user.
    - the **database** option requires a new row to be added to a table for each LP problem. Each row will contain two attributes: **`problemName`** and the **`optimalValue`**. The program should add the rows to a table whose name is provided by the user. This table should be added to the database called **`LP`**. The program should check if the **`LP`** database exists. If it does not, the program should create it. The program should also check if the table named by the user exists. If it exists, the program should delete the table (i.e., DROP the table) and create another one and name it what the user wanted. Both the fields in the table should be strings of length 20, that is CHAR (20).
    - the **default** output for the program is written to the screen with the problem name and the optimal value written on a separate line for each LP problem

You are required to use a module file and use functions to complete this assignment. I have given you one useful function below:

```
def getFileNames(fn):
      """
      Input = name of JSON file
      Returns = JSON object
      """
    import os
    included_extenstions = ['json' ] ;
    file_names = [fn for fn in os.listdir(os.getcwd())
      if any([fn.endswith(ext)
      for ext in included_extenstions])];
    return file_names
```

Sample outputs are shown below:

```
$ python A05_gwid.py
The optimal value for Whiskas is 0.520
The optimal value for Intro LP is 410.000


$ python A05_gwid.py -d myTable
The optimal value for Whiskas is 0.520
The optimal value for Intro LP is 410.000
Output being sent to table: myTable in the LP database
Output written to table: myTable in the LP database


$ python A05_gwid.py -t myTable.txt
The optimal value for Whiskas is 0.520
The optimal value for Intro LP is 410.000
Output being sent to file: myTable.txt
Output written to file: myTable.txt


$ python A05_gwid.py -t myTable.txt -d myTable
The optimal value for Whiskas is 0.520
The optimal value for Intro LP is 410.000
Output being sent to file: myTable.txt
Output written to file: myTable.txt
Output being sent to table: myTable in the LP database
Output written to table: myTable in the LP database
```

[1] This is a useful reference http://stackoverflow.com/questions/2225564/get-a-filtered-list-of-files-in-a-directory
[2] This is a useful document - http://mycodesite.com/mongodb-basics-and-tips/