TP1 – Linear programming introduction with Python3

1 PuLP: a Python package for linear programming

Why using Python and PuLP?

- Python is free and open source;
- so PuLP too;
- PuLP is a Python package so it is easy to manipulate data and the linear model with only one implementation language (*c.f.* Figure 1).

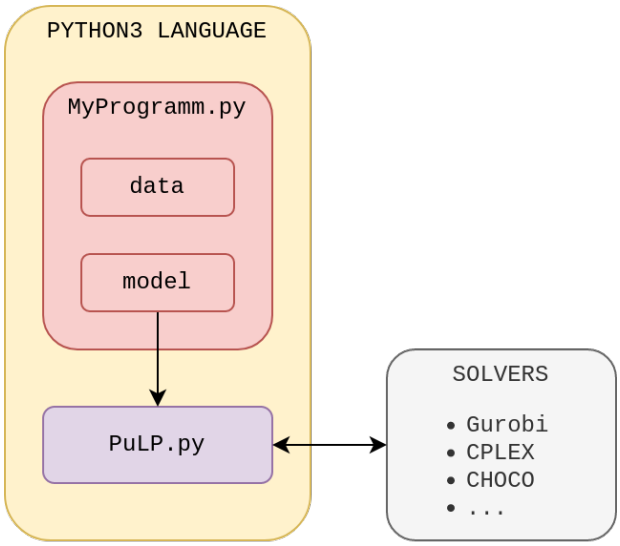

Figure 1: PuLP is a class based package for Python. So the data are loaded with a Python program, and the linear model is written in Python. Then, the PuLP package rewrite the model instanced with the data as the chosen solver wants. Finally, PuLP retrieves the solver's output which can be manipulated by Python code.

2 Beginning with Python3

2.1 TPs organization

Download and extract the directory TP_BASE. Have a look to the README-EN.md file. In short, you always have to place you at the TP_BASE root (you can rename this directory), and load a python environment common for all the TPs.

The first time, you have to configure this environment:

```
# in TP_BASE
./configs/config_venv3.sh  # minimum
# OR
./configs/config_venv3_all.sh  # recommended: minimum + linters
```

Then, load it:

```
# in TP_BASE
source ./.venv_3/bin/activate
```

You know your environment is loaded when the prefix `(.venv_38)` appears at the beginning of your terminal line:

```
(.venv_3) your_command_line
```

Read the `TP_BASE/README-EN.md` file for more details.

2.2 Train you!

Download the `TP1` directory and put it at the same level as `TP0`. Place you in `TP1`:

```
# in TP_BASE
(.venv_3) cd TP1
```

If you have never coding in `Python3`, you should take a look at files `first_prog.py` and `a_module.py`.

Question 1.

Given a comprehension created list of couples, create a dictionary such as, for each couple in the list, the key is the first element and the values (stored in a list) are the last elements of the couple.

You can use either a loop, or a dictionary comprehension. Also, take care of already-existing key thanks to an if/else statement: if the key already exists, add the values to the list of values associated to this key in the dictionnary.

Example:

```
    This is my first list. It has been created by list comprehension.
[(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (2, 5)]

The result dictionnary based on the above list

{0: [1, 2, 3], 1: [2, 3, 4], 2: [3, 4, 5]}
```

Perform the following tasks:

i. Create an empty file `test.py`
ii. Write your program
iii. Execute it in a shell with the command:

```
# in TP_BASE/TP1
(.venv_3) python3 test.py
```

3 Pre-structured program for implementing model in PuLP with Python

Take a look at file `TP_BASE/TP0/base_model.py`. It is a given base-level structure for writing a model in `PuLP` with `Python3`. Functions separate the data instantiation from the model implementation, and from solving model instanced with data. You have to be able to explain each line.

Next sessions you can use this `Python-PuLP` skeleton.

4 Simple example: the diet problem

Let us consider a simple optimization problem. We consider a diet problem: we want to *minimize* the cost of the food we eat while some nutritional *constraints* (calories, sugar and fat levels, etc.) are satisfied. This is the list of food:

1. chocolate candies

2. chocolate ice scream
3. coca cola
4. cheesecake

Denote by x_i the quantity of the food i. The set $x_i, i = 1, \ldots, 4$ represents the variables for the following optimization problem:

$$\min \quad 50\,x_1 + 20\,x_2 + 30\,x_3 + 80\,x_4 \quad \text{(each } x_i \text{ is weighted with the cost)}$$

subject to:

$$\begin{array}{rcrcrcrcrl}
400x_1 & + & 200x_2 & + & 150x_3 & + & 500x_4 & \geq & 500 & \text{calories} \quad (1)\\
3x_1 & + & 2x_2 & & & & & \geq & 6 & \text{chocolate} \quad (2)\\
2x_1 & + & 2x_2 & + & 4x_3 & + & 4x_4 & \geq & 10 & \text{sugar} \quad (3)\\
8x_1 & + & 4x_2 & + & x_3 & + & 5x_4 & \geq & 8 & \text{fat} \quad (4)
\end{array}$$

In the objective function, each variable x_i is weighted by an integer w_i. For each constraint c, you have an integer a_i^c which multiplies each variable x_i. The inequality is bounded by another integer b_c (see this as a limit to not overcome in the case of $\geq$ sign).

As we want to generalize our model without written directly the value of each constant (i.e. w_i, a_i^c, b_c) think about the python objects to use in order to store the constants values.

The file `TP1/tp1_simple_example.py` is the model which partially implements this problem. Take a look at the functions:

1. `set_model_diet()` is where the global model is set;
2. `solve_simple_example()` is where the data are set *i.e.* the constants values and then the model is called to be used with these constants.

Question 2.

After having understood the implementation in `PuLP` and `Python3` language, write the missing constraints. Then, launch the program with (make sure your python environment is set!):

```
# in TP_BASE/TP1
(.venv_3) python3 tp1_simple_example.py
```

Question 3.

Did the solver succeed? In the positive case, is the solution optimal?

Question 4.

> Give the solution i.e. the values of variables.

5 PuLP documentation

5.1 Main menu

`https://coin-or.github.io/pulp/`

5.2 Internal

`https://coin-or.github.io/pulp/technical/index.html`