Documentation Project DIstribution Classic Program

- solveProjectsDistribution.py

Main script to launch the distribution of students

```
def getOptionsFromXLS(students, nbProjects):

    Get choices of students from the choices made in the XLS file

    :param students: list of students
    :param nbProjects: number of projects

    :type students: list of integers
    :type nbProjects: integer

    :return w: options of the students
    :rtype w: dict {(student, project) : weight  of options}
```

- fromExcelToPython.py

Script allowing to get the information of the students choices from a XLS file

```
class Student:
    Student class corresponding to the student in M1 at ISEN

    :member id: identifiant of the student
    :member option1: first choice of the student
    :member option2: second choice of the student
    :member option3: third choice of the student
    :member option4: fourth choice of the student
    :member option5: fifth choice of the student


def getStudentsFromXls(fileXls, numberProjects, numberStudents)

    Get the list of student and their options from a xls file

    :param fileXls: the name of the file to get the option of each students
    :param numberProjects: the number of projects
    :param numberStudent: the number of students

    :type fileXls: string
    :type numberProjects: int
    :type numberStudents: int
```

```
    :return students: list of students with their choce of options
    :return studentsID: list of the name of the students

    :rtype students: table of Student
    :rtype studentsID: table of string
```

- lp.py

Export a lp file from a model

```
class Student:

    Student class corresponding to the student in M1 at ISEN

    :member id: identifiant of the student
    :member option1: first choice of the student
    :member option2: second choice of the student
    :member option3: third choice of the student
    :member option4: fourth choice of the student
    :member option5: fifth choice of the student

class Project:

    Project class corresponding to the project proposed by ISEN

    :member id: identifiant of the project
    :member numberofStudentAllowed: number of student allowed for this project


class Model:

    Model class constructed with a list of students and a list of projects

    :member students: list of students from class Student
    :member projects: list of projects from class Project


def printModel(model):

    Print the information of a model

    :parameter model: model corresponding to the current problem
    :type model: class Model


def lp(model :Model, fileName):

    Transform a model into a lp file

    :parameter model: model corresponding to the current problem
    :type model: class Model
```

```
    :parameter fileName: name of the lp file to be written
    :type fileName: string

    :return lpFile: lpFile corresponding to the current problem
    :rtype lpFile: file (.lp => Linear Program)
```

- findSolution.py

Script to launch the solver glpk in the consol

```
def findSolution(lpFile, solutionFile):

    Launch the command in windows shell to find the solution of the lp file

    :param lpFile:
    :param solutionFile:

    :type lpFile:
    :type solutionFile:

    :return solutionFile: the file of the solution
```

- readSolution.py

Read and interface the solution found by the solver GLPK

```
class ProjectSolved:

    Project class corresponding to the project divided by student according to
their choices

    :member id: identifiant of the project
    :member students: list of the students for this project
    :member room: room planned for this project

def readSolution(fileName, nbStudents, nbProjects, room):

    Interfaces the solutions from the result of lp file after using glpk

    :parameter fileName: Name of the file to interface
    :parameter nbStudents: number of students
    :parameter nbProjects: number of projects
    :parameter room: nb of student alowed per project

    :type fileName: String
    :type nbStudents: integer
```

```
    :type nbProjects: integer
    :type room: table of integer



def interface(projects :ProjectSolved, fileName, students: Student, studentID, w,
room):

    Interfaces the results of the division of projects by students

    :param projects: list of the projects
    :param fileName: name of the file to save the results
    :param students: list of the students with the options
    :param studentID: list of name of students
    :param w: options of the students
    :param room: number of students allowed per projects

    :type projects: class Porject
    :type fileName: String
    :type students: table of Student
    :type studentID: table of String
    :type w: dictionary of {(student, project) : integer}
    :type room: table of integer

    :return files: the display in files saved



def calculateProba(projects : ProjectSolved, students):

    Calculate the different distributions of students for each options

    :param projects: list of the projects
    :param students: list of the students with the options

    :type projects: class Project
    :type students: table of Student

    :return files: the display in files saved
```