Documentation Project Distribution Quantum Program

- main.py

Main script to launch the distribution of students with the D-wave quantum computer

- display.py

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

    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

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

- hybridProgUsefulFunctions.py

Useful functions to launch the program

```
def getListOfStudent(n_students):

    Get the list of student
    :param n_students: number of the students
    :type n_students: integer

    :return students: list of students
    :rtype students: list of integer


def roomWithCoeff(rooms, coeff):

    Add a coefficient to a list of room

    :param rooms: list of rooms
    :param coeff: coefficient to multiply each element of the list

    :type rooms: list of integer
    :type coeff: integer

    :return roomCoeffed: list of room with coefficient
    :rtype roomCoeffed: list of integer


def getIndex(student_index, project_index, n_projects):

    Get the index of the matrix corresponding to the index of students and
projects
    (opposite of get_student_and_project)

    :param student_index: index of the student
    :param project_index: index of the project
    :param n_projects: number of projects

    :type student_index: integer
    :type project_index: integer
    :type n_projects: integer

    :return index: index corresponding to project_student
    :rtype students: integer


def getStudentAndProject(index, n_projects):

    Get the indexes of the matrix corresponding to the index of students and
projects
    (opposite of get_index)

    :param index: index of the project_student
    :param n_projects: number of projects

    :type index: integer
    :type n_projects: integer

    :return (student_index, project_index): index corresponding to project_student
    :rtype (student_index, project_index): tuple of integers
```

```
def getProjectsSolvedFromSched(n_projects, tmpProjects, room):

    Get the projects from a list

    :param n_projects: number of projects
    :param tmpProjects: dictionnary of project not sorted
    :param room: list of number of student allowed per project

    :type n_projects: integer
    :type tmpProjects: dictionnary
    :type room: list of integer

    :return projects: dictionnary of projects associated to students
    :rtype projects: dictionnary of integers

def changeW(w):

    Change the weight of each options

    :param w: {(student, project) : weight}, initial weights
    :type w: {(integer, integer) : integer}

    :return w: param w with the good weights
    :rtype w:  {(integer, integer) : integer}
```

- Student.py

```
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
```

- ProjectSolved.py

Export a lp file from a model

```
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
```