"""

Projet 4 : résolution de systèmes linéaires

Résolution de systèmes d'équations linéaires carrées par combinaison linéaire

"""

\_\_author\_\_ **=** "Charlotte Nachtegael"

\_\_date\_\_ **=** "10 décembre 2015"

**def** solve\_system**():**

"""

:return: solution of a system of linear equation or None if the system is impossible or undetermined

"""

system **=** encode\_system**()**

sysTriangulaire **=** triangulation**(**system**)**

solution **=** substitution**(**sysTriangulaire**)**

**print(**"Solution:"**)**

**print(**solution**)**

**def** encode\_system**():**

"""

:return: system of n equations with n variables

"""

# input of variables

not\_numbers **=** **False**

all\_different **=** **False**

**while** **not** not\_numbers **or** **not** all\_different**:**

**print(**"Please choose your variables (letters between a and z separated by a space)"**)**

var **=** input**()**

list\_var **=** var**.**split**()**

**try:**

**for** i **in** range**(**len**(**list\_var**)):**

**if** **not** **(**96 **<** ord**(**list\_var**[**i**])** **<** 123**):**

**raise** TypeError

not\_numbers **=** **True**

**for** i **in** range**(**len**(**list\_var**)** **-** 1**):**

**for** j **in** range**(**i **+** 1**,** len**(**list\_var**)):**

**if** list\_var**[**i**]** **==** list\_var**[**j**]:**

**raise** Exception

all\_different **=** **True**

**except** **(**TypeError**,** Exception**):**

**print(**"Nice try, but all your variables must be letters between 'a' and 'z' and different ! Try again :)"**)**

list\_var**.**append**(**'ti'**)**

# input of the coefficients for the equations

nb\_equations **=** len**(**list\_var**)**

list\_coefficients\_all **=** **[]**

**for** i **in** range**(**1**,** nb\_equations**):**

**print(**"Equation"**,** i**,** ":"**)**

nb\_coeff **=** **False**

all\_numbers **=** **False**

**while** **not** nb\_coeff **or** **not** all\_numbers**:**

**print(**"Coefficients of"**,** var**,** "and the independent term :"**)**

coeff **=** input**()**

list\_coeff **=** coeff**.**split**()**

**try:**

# test of enough number of coefficients

**if** len**(**list\_coeff**)** **!=** len**(**list\_var**):**

**raise** ValueError

**else:**

nb\_coeff **=** **True**

# all the input are numbers

**for** j **in** range**(**len**(**list\_coeff**)):**

list\_coeff**[**j**]** **=** int**(**list\_coeff**[**j**])**

all\_numbers **=** **True**

**except:**

**print(**

"Error : All coefficients must be integers or you did not give the good amount of coefficients !"**)**

list\_coefficients\_all**.**append**(**list\_coeff**)**

# encode the equations in dictionaries

system **=** **[]**

**for** i **in** range**(**len**(**list\_coefficients\_all**)):**

equation **=** **{}**

**for** j **in** range**(**len**(**list\_coefficients\_all**[**0**])):**

equation**[**list\_var**[**j**]]** **=** list\_coefficients\_all**[**i**][**j**]**

system**.**append**(**equation**)**

**print(**"Linear system encoded :"**)**

print\_system**(**system**,** list\_var**)**

**return** system

**def** print\_system**(**system**,** list\_var**):**

"""

:param system: system (list) of several equations in the form of dictionaries

:param list\_var: list of the variables of the equations

:return: a print of the equations in the system

"""

**for** i **in** range**(**len**(**system**)):**

**if** system**[**i**][**list\_var**[**0**]]** **>** 0**:**

**if** system**[**i**][**list\_var**[**0**]]** **==** 1**:**

**print(**list\_var**[**0**],** end**=**' '**)**

**else:**

**print(**str**(**system**[**i**][**list\_var**[**0**]])** **+** list\_var**[**0**],** end**=**' '**)**

**elif** system**[**i**][**list\_var**[**0**]]** **<** 0**:**

**if** system**[**i**][**list\_var**[**0**]]** **==** **-**1**:**

**print(**'-' **+** list\_var**[**0**],** end**=**' '**)**

**else:**

**print(**str**(**system**[**i**][**list\_var**[**0**]])** **+** list\_var**[**0**],** end**=**' '**)**

**for** j **in** range**(**1**,** len**(**list\_var**)):**

**if** list\_var**[**j**]** **==** 'ti'**:**

**if** system**[**i**][**list\_var**[**j**]]** **>** 0**:**

**print(**'+'**,** system**[**i**][**list\_var**[**j**]],** end**=**' '**)**

**elif** system**[**i**][**list\_var**[**j**]]** **<** 0**:**

**print(**'-'**,** abs**(**system**[**i**][**list\_var**[**j**]]),** end**=**' '**)**

**elif** system**[**i**][**list\_var**[**j**]]** **>** 0**:**

**if** system**[**i**][**list\_var**[**0**]]** **!=** 0**:**

**print(**'+'**,** end**=**' '**)**

**if** system**[**i**][**list\_var**[**j**]]** **==** 1**:**

**print(**list\_var**[**j**],** end**=**' '**)**

**else:**

**print(**str**(**system**[**i**][**list\_var**[**j**]])** **+** list\_var**[**j**],** end**=**' '**)**

**elif** system**[**i**][**list\_var**[**j**]]** **<** 0**:**

**print(**'-'**,** end**=**' '**)**

**if** system**[**i**][**list\_var**[**j**]]** **==** **-**1**:**

**print(**list\_var**[**j**],** end**=**' '**)**

**else:**

**print(**str**(**abs**(**system**[**i**][**list\_var**[**j**]]))** **+** list\_var**[**j**],** end**=**' '**)**

**print(**'='**,** 0**)**

**def** triangulation**(**system**):**

"""

:param system: system (list) of several equations in the forms of dictionaries

:return: a triangular system if possible, None if undetermined or impossible to solve

"""

**def** check\_equations**(**sysTriangulaire**,** variables**):**

"""

:param sysTriangulaire: system (list) of several equations in the forms of dictionaries

:param variables: list of variables used in the equations

:return: the system if it is solvable, None if undetermined or impossible to solve

"""

nb\_eq\_total **=** len**(**sysTriangulaire**)**

nb\_eq\_check **=** 0

zero\_variable **=** **False**

# check the coefficient for each variables until one equation with no variable is found

**while** **not** zero\_variable **and** nb\_eq\_check **<** nb\_eq\_total**:**

nb\_zero **=** 0

**for** x **in** range**(**len**(**variables**)):**

**if** sysTriangulaire**[**nb\_eq\_check**][**variables**[**x**]]** **==** 0**:**

nb\_zero **+=** 1

nb\_eq\_check **+=** 1

**if** nb\_zero **==** len**(**variables**):**

zero\_variable **=** **True**

**if** **not** zero\_variable**:**

res **=** sysTriangulaire

**else:**

res **=** **None**

**return** res

sysTriangulaire **=** **[]**

variables **=** **[]**

nb\_eq **=** len**(**system**)**

# compute the triangular system

**for** elem **in** system**[**0**]:**

**if** elem **is** **not** 'ti'**:**

variables**.**append**(**elem**)**

variables**.**sort**()**

nb\_variables **=** len**(**variables**)**

**for** i **in** range**(**nb\_variables **-** 1**):**

**for** j **in** range**(**i**,** nb\_eq **-** 1**):**

**for** k **in** range**(**j **+** 1**,** nb\_eq**):**

system**[**k**]** **=** compute\_combili**(**system**[**j**],** system**[**k**],** variables**[**i**])**

sysTriangulaire**.**append**(**system**[**i**])**

sysTriangulaire**.**append**(**system**[-**1**])**

res **=** check\_equations**(**sysTriangulaire**,** variables**)**

**return** res

**def** compute\_combili**(**equation1**,** equation2**,** var\_elimi**):**

"""

:param equation1: dictionary with the coefficients and variables of an equation

:param equation2: dictionary with the coefficients and variables of an equation

:param var\_elimi: string of the variable to eliminate

:return: one dictionnary with the coefficients and variables of an equation resulting

of a linear combination of equation1 and equation 2 with the coefficient of the var\_elimi

equal to 0

"""

**def** ppcm**(**x**,** y**):**

"""

:param x: number 1

:param y: number 2

:return: smallest multiple in common of number 1 and 2

"""

**def** pgcd**(**x**,** y**):**

"""

:param x: number 1

:param y: number 2

:return: the biggest common divisor of number 1 and 2

"""

**if** y **==** 0**:**

res **=** x

**else:**

res **=** pgcd**(**y**,** x **%** y**)**

**return** res

divisor **=** pgcd**(**x**,** y**)**

**if** divisor **==** 0**:**

res **=** 0

**else:**

res **=** x **\*** y **/** divisor

**return** res

# calculate the coefficient for the linear combination

coeff\_eq1**,** coeff\_eq2 **=** equation1**[**var\_elimi**],** equation2**[**var\_elimi**]**

multiple\_in\_common **=** ppcm**(**abs**(**coeff\_eq1**),** abs**(**coeff\_eq2**))**

**if** multiple\_in\_common **==** 0**:**

res **=** equation2

**else:**

c1 **=** multiple\_in\_common **/** coeff\_eq1

c2 **=** multiple\_in\_common **/** coeff\_eq2

# compute the final equation with the coefficient of var\_elimi equal to 0

equation\_final **=** **{}**

**for** elem **in** equation1**:**

**if** abs**(**equation1**[**elem**]** **\*** c1**)** **+** abs**(**equation2**[**elem**]** **\*** c2**)** **==** 0**:**

equation\_final**[**elem**]** **=** 0.0

**elif** abs**(**equation1**[**elem**]** **\*** c1 **-** equation2**[**elem**]** **\*** c2**)** **/** **(**

abs**(**equation1**[**elem**]** **\*** c1**)** **+** abs**(**equation2**[**elem**]** **\*** c2**))** **<** 1e-05**:**

equation\_final**[**elem**]** **=** 0.0

**else:**

equation\_final**[**elem**]** **=** equation1**[**elem**]** **\*** c1 **-** equation2**[**elem**]** **\*** c2

res **=** equation\_final

**return** res

**def** substitution**(**sysTriangulaire**):**

"""

:param sysTriangulaire: triangular system of several equations in the form of dictionaries or None

:return: the solution set of the system in the for of a dictionary

"""

**if** sysTriangulaire **==** **None:**

solution **=** **None**

**else:**

nb\_eq **=** len**(**sysTriangulaire**)**

variables **=** **[]**

solution **=** **{}**

**for** elem **in** sysTriangulaire**[**0**]:**

**if** elem **is** **not** 'ti'**:**

variables**.**append**(**elem**)**

variables**.**sort**()**

**for** i **in** range**(**nb\_eq **-** 1**,** **-**1**,** **-**1**):**

**if** sysTriangulaire**[**i**][**variables**[**i**]]** **==** 0**:**

solution**[**variables**[**i**]]** **=** 0.0

**else:**

solution**[**variables**[**i**]]** **=** sysTriangulaire**[**i**][**'ti'**]** **\*** **(-**1**)** **/** sysTriangulaire**[**i**][**variables**[**i**]]**

**if** solution**[**variables**[**i**]]** **==** **-**0.0**:**

solution**[**variables**[**i**]]** **=** 0.0

**if** i **>** 0**:**

**for** elem **in** solution**:**

sysTriangulaire**[**i **-** 1**][**'ti'**]** **=** sysTriangulaire**[**i **-** 1**][**'ti'**]** **+** **(**sysTriangulaire**[**i **-** 1**][**elem**]** **\*** solution**[**elem**])**sysTriangulaire**[**i **-** 1**][**elem**]** **=** 0

**return** solution

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

solve\_system**()**