Calul matriciel

<sxh python; title : calcul_matriciel.py> #!/usr/bin/env python # -*- coding: UTF-8 -*- #http://www.siteduzero.com/tutoriel-3-223267-apprenez-a-programmer-en-python.html (pdf) #http://stackoverflow.com/questions/455612/python-limiting-floats-to-two-decimal-points #pour operation matriciel verification mathématique : http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal #http://www.pythonfrance.com/codes/OPERATION-MATRICIELLE_48359.aspx #http://inforef.be/swi/python.htm # travail de CD, ba2 chimie 2011-2012

from Tkinter import *

from sys import *

def is_number(s):

```
  try:
      float(s)
      return True
  except ValueError:
      return False
```

def Supp_lc(m,n,M):

```
    "retourne la matrice A sans la m ième ligne et la n ième colonne"
    Mlin=len(M)
    result=[]
    Rep=[]
    for i in range(Mlin):
          if i!=m:
                for j in range(Mlin):
                      if j!=n:
                            result.append(M[i][j])
                            #Ajoute les éléments tant qu'ils ne sont pas sur
la m ième
                            # ligne ou n ième colonne
    for k in range(0,len (result),Mlin-1):
          Rep.append(result[k:k+Mlin-1])
          #Scinde la liste "result" en "Longueur(result)/(Mlin-1)" listes de
taille "Mlin-1"
    return Rep
```

def Det(A):

```
    " retourne le déterminat de la matrice A"
    if len(A)==1:
          return A[0][0]
    else:
          s=0
```

```
          j=0
          while j<len(A):
                B=Supp_lc(j,0,A)#Supprime 1ere ligne et colonne de A
                if j%2==0: #(-1)^(i+j) * Det("matrice où on retire 1ere
ligne et colonne")
                           # et on considère toujours i=0 donc dépend juste
de j
                      s=s+A[j][0]*Det(B)
                else:
                      s=s-A[j][0]*Det(B)
                j=j+1
          return s
```

def invmat(A):

```
    "Donne l'inverse d'une matrice carrée  A"
    d=Det(A)
    if d==0:
          return 'La Matrice n\'est pas inversible'
    else:
          B=Mul_coff(1./d, Comat(A))
          inv=Trans_matrix(B)
    return inv
```

def Comat(A):

```
    "Donne la comatrice d'une matrice A"
    N=len (A)
    k=0
    com=[None]*N
    while k<N:
          com[k]=[0]*N
          l=0
          while l<N:
                B=Supp_lc(k,l,A)
                if (k+l)%2==0:
                      com[k][l]=(Det(B))
                else:
                      com[k][l]=((-1)*Det(B))
                l=l+1
          k=k+1
    return com
```

def Trans_matrix(n):

```
  """Retourne la transposée de la matrice"""
  s=[]
  for i in range(len(n[0])): #correspond à la dimension de la matrice
      t=[]
```

```
    for j in range(len(n)):
        t.append(n[j][i])
    s.append(t)
return s
```

def Mul_coff(a,l):

```
"""Multiplie les coefficients de la matrice par un réel"""
s = []
for i in range(len(l)):
    t =[]
    for j in range(len(l[0])):
        t.append( a* l[i][j])
    s.append(t)
return s
```

def Dif_matrix(h,l):

```
"""Retourne la différence de deux matrices"""
if len(h)==len(l) and len(h[0])==len(l[0]):
    s = []
    for i in range(len(h)):
        t =[]
        for j in range(len(l[0])):
            t.append( h[i][j] - l[i][j])
        s.append(t)
    return s
else:
    print "Les tailles de vos matrices ne sont pas aqéquates"
```

def Sum_matrix(h,l):

```
"""retourne la somme de deux matrices"""
if len(h)==len(l) and len(h[0])==len(l[0]):
    s = []
    for i in range(len(h)):
        t =[]
        for j in range(len(l[0])):
            t.append( h[i][j] + l[i][j])
        s.append(t)
    return s
else:
    print "Les tailles de vos matrices ne sont pas aqéquates"
```

def Mul_matrix(h,l):

```
"""Retourne le produit de deux matrices"""
if len(h[0]) == len(l):
    s = []  #nouvelle matrice retournée
    for i in range(len(h)):
```

```
          t = [] #création de ligne qu'on ajoutera ensuite à "s"
          for j in range(len(l[0])):
              res = 0  # résultat, chaque fois remis à 0 pour éviter des
valeurs fausses
              for k in range(len(h[0])):
                  res += h[i][k] * l[k][j]
              t.append(res)
          s.append(t)
      return s
  else:
      print "Les tailles de vos matrices ne sont pas aqéquates"
```

def opentop ():

```
  """fen01.destroy () """
  dim = entr1.get()
  if is_number(dim) and float(dim) > 0:
      fen02 = Tk()
      fen02.title("Choisir l'opération")
      var=IntVar()
      r1=Radiobutton(fen02,text="déterminant",variable=var,value=0,command
=affiche)
      r2=Radiobutton(fen02,text="comatrice",variable=var,value=1,command
=affiche1)
      r3=Radiobutton(fen02,text="matrice
inverse",variable=var,value=2,command =affiche2)
      r4=Radiobutton(fen02,text="matrice
transposée",variable=var,value=3,command =affiche3)
      r5=Radiobutton(fen02,text="multiplication par un
réel",variable=var,value=4,command =affiche4)
      r6=Radiobutton(fen02,text="soustraction",variable=var,value=5,command
=affiche5)
      r7=Radiobutton(fen02,text="addition",variable=var,value=6,command
=affiche6)
r8=Radiobutton(fen02,text="multiplication",variable=var,value=7,command
=affiche7)
```

```
      r1.grid(row=1,column=0, sticky=W)
      r2.grid(row=2,column=0, sticky=W)
      r3.grid(row=3,column=0, sticky=W)
      r4.grid(row=4,column=0, sticky=W)
      r5.grid(row=5,column=0, sticky=W)
      r6.grid(row=6,column=0, sticky=W)
      r7.grid(row=7,column=0, sticky=W)
      r8.grid(row=8,column=0, sticky=W)
```

```
      fen02.mainloop()
  else:
      image = None #valeur nulle pour objets autres que des nombres
      while not image:
```

```
        entree = entr1.get()
        image = is_number(entree)
        fen01.mainloop()
    return image
```

def affiche ():

```
fen03 = Tk()
fen03.title('Choissisez les valeurs de votre matrice')
ni=int(entr1.get())
nj=int(entr1.get())
for i in range (ni):
    for j in range (nj):
        entries=Entry(fen03)
        entries.grid(row=i,column=j)
        Zones1.append(entries)
bou3= Button (fen03, text="calcul du déterminant",command= det)
bou3.grid(row=i+1,column =j)
fen03.mainloop()
```

def affiche1():

```
fen03 = Tk()
fen03.title('Choissisez les valeurs de votre matrice')
ni=int(entr1.get())
nj=int(entr1.get())
for i in range (ni):
    for j in range (nj):
        entries=Entry(fen03)
        entries.grid(row=i,column=j)
        Zones4.append(entries)
bou3= Button (fen03, text="calcul de la comatrice",command= comatr)
bou3.grid(row=i+1,column =j)
fen03.mainloop()
```

def affiche2():

```
fen03 = Tk()
fen03.title('Choissisez les valeurs de votre matrice')
ni=int(entr1.get())
nj=int(entr1.get())
for i in range (ni):
    for j in range (nj):
        entries=Entry(fen03)
        entries.grid(row=i,column=j)
        Zones2.append(entries)
bou3= Button (fen03, text="calcul de la matrice inverse",command= inv)
bou3.grid(row=i+1,column =j)
fen03.mainloop()
```

def affiche3():

```
  fen03 = Tk()
  fen03.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen03)
          entries.grid(row=i,column=j)
          Zones3.append(entries)
  bou3= Button (fen03, text="calculer la transposée de la matrice",command=
transf)
  bou3.grid(row=i+1,column =j)
  fen03.mainloop()
```

def affiche4():

```
  fen03 = Tk()
  fen03.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen03)
          entries.grid(row=i,column=j)
          Zones5.append(entries)
  bou3= Button (fen03, text="Choix du réel",command= choix)
  bou3.grid(row=i+1,column =j)
  entr.append(fen03)
  fen03.mainloop()
```

def affiche5():

```
  fen03 = Tk()
  fen03.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen03)
          entries.grid(row=i,column=j)
          Zones6.append(entries)
  bou3= Button (fen03, text="Choix 2ème matrice",command= matrice2)
  bou3.grid(row=i+1,column =j)
  fen03.mainloop()
```

def affiche6():

```
fen03 = Tk()
fen03.title('Choissisez les valeurs de votre matrice')
ni=int(entr1.get())
nj=int(entr1.get())
for i in range (ni):
    for j in range (nj):
        entries=Entry(fen03)
        entries.grid(row=i,column=j)
        Zones7.append(entries)
bou3= Button (fen03, text="Choix 2ème matrice",command= matrice3)
bou3.grid(row=i+1,column =j)
fen03.mainloop()
```

def affiche7():

```
fen03 = Tk()
fen03.title('Choissisez les valeurs de votre matrice')
ni=int(entr1.get())
nj=int(entr1.get())
for i in range (ni):
    for j in range (nj):
        entries=Entry(fen03)
        entries.grid(row=i,column=j)
        Zones8.append(entries)
bou3= Button (fen03, text="Choix 2ème matrice",command= matrice4)
bou3.grid(row=i+1,column =j)
fen03.mainloop()
```

def det():

```
s = []
n =int(entr1.get())
for i in range(n):
    t = []
    for j in range(n):
        t.append(float(Zones1[n*i+j].get()))
    s.append(t)
fen04=Tk()
r = Det(s)
chaine2=Label (fen04, text= "%.2f" % r)
chaine2.grid()
```

def comatr ():

```
s = []
n =int(entr1.get())
for i in range(n):
    t = []
    for j in range(n):
        t.append(float(Zones4[n*i+j].get()))
```

```
    s.append(t)
fen04=Tk()
r = Comat(s)
for i in range (len(r)):
    for j in range (len(r[0])):
        b = Label(fen04, text= "%.2f" % r[i][j])
        b.grid(row=i,column =j)
```

def inv ():

```
s = []
n =int(entr1.get())
for i in range(n):
    t = []
    for j in range(n):
        t.append(float(Zones2[n*i+j].get()))
    s.append(t)
fen04=Tk()
r = invmat(s)
if type(r) is str:
    b = Label(fen04, text=  r)
    b.grid(row=i,column =j)
else:
    for i in range (len(r)):
        for j in range (len(r[0])):
            b = Label(fen04, text= "%.2f" % r[i][j])
            b.grid(row=i,column =j)
```

def transf():

```
s = []
n =int(entr1.get())
for i in range(n):
    t = []
    for j in range(n):
        t.append(float(Zones3[n*i+j].get()))
    s.append(t)
fen04=Tk()
r = Trans_matrix(s)
for i in range (len(r)):
    for j in range (len(r[0])):
        b = Label(fen04, text= "%.2f" % r[i][j])
        b.grid(row=i,column =j)
```

def choix():

```
fen04=Tk()
fen04.title("choix du réel")
entr2=Entry(fen04)
```

```
  entr2.grid(row=0,column=0)
  entr.append(entr2)
  bou5=Button (fen04, text = "multiplication", command= mult)
  bou5.grid(row=1,column=0)
  fen04.mainloop()
```

def mult():

```
  s = []
  n =int(entr1.get())
  a = entr[1].get()
  if is_number(a):
      for i in range(n):
          t = []
          for j in range(n):
              t.append(float(Zones5[n*i+j].get()))
          s.append(t)
      fen04=Tk()
      r = Mul_coff(float(a),s)
      for i in range (len(r)):
          for j in range (len(r[0])):
              b = Label(fen04, text= "%.2f" % r[i][j])
              b.grid(row=i,column =j)
  else:
      image = None
      while not image:
          entree = entr[1].get()
          image = is_number(entree)
          entr[0].mainloop()
      return image
```

def matrice2():

```
  fen04 = Tk()
  fen04.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen04)
          entries.grid(row=i,column=j)
          Zones9.append(entries)
  bou4= Button (fen04, text="Soustraire",command= soust)
  bou4.grid(row=i+1,column =j)
  fen04.mainloop()
```

def matrice3():

```
  fen04 = Tk()
  fen04.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
```

```
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen04)
          entries.grid(row=i,column=j)
          Zones10.append(entries)
  bou4= Button (fen04, text="additionner",command= add)
  bou4.grid(row=i+1,column =j)
  fen04.mainloop()
```

def matrice4():

```
  fen04 = Tk()
  fen04.title('Choissisez les valeurs de votre matrice')
  ni=int(entr1.get())
  nj=int(entr1.get())
  for i in range (ni):
      for j in range (nj):
          entries=Entry(fen04)
          entries.grid(row=i,column=j)
          Zones11.append(entries)
  bou4= Button (fen04, text="Multiplier",command= multip)
  bou4.grid(row=i+1,column =j)
  fen04.mainloop()
```

def soust():

```
  s = []
  z = []
  n =int(entr1.get())
  for i in range(n):
      t = [] #correspont à chacune des lignes de la matrice créée
      p = [] #qu'on ajoute peu à peu à s pour construire la matrice
      for j in range(n):
          t.append(float(Zones6[n*i+j].get()))
          p.append(float(Zones9[n*i+j].get()))
      s.append(t)
      z.append(p)
  fen05=Tk()
  r = Dif_matrix(s,z)
  for i in range (len(r)):
      for j in range (len(r[0])):
          b = Label(fen05, text= "%.2f" % r[i][j])
          b.grid(row=i,column =j)
```

def add():

```
  s = []
  z=[]
```

```
  n =int(entr1.get())
  for i in range(n):
      t = []
      p = []
      for j in range(n):
          t.append(float(Zones7[n*i+j].get()))
          p.append(float(Zones10[n*i+j].get()))
      s.append(t)
      z.append(p)
  fen05=Tk()
  r = Sum_matrix(s,z)
  for i in range (len(r)):
      for j in range (len(r[0])):
          b = Label(fen05, text= "%.2f" % r[i][j])
          b.grid(row=i,column =j)
```

def multip():

```
  s = []
  z=[]
  n =int(entr1.get())
  for i in range(n):
      t = []
      p = []
      for j in range(n):
          t.append(float(Zones8[n*i+j].get()))
          p.append(float(Zones11[n*i+j].get()))
      s.append(t)
      z.append(p)
  fen05=Tk()
  r = Mul_matrix(s,z)
  for i in range (len(r)):
      for j in range (len(r[0])):
          b = Label(fen05, text= "%.2f" % r[i][j])
          b.grid(row=i,column =j)
```

Zones = [] #tableau contenant des adresses memoires Zones1 =[] #des positions des cases de la matrice sur la fenêtre Zones2 =[] Zones3 =[] Zones4 =[] Zones5 =[] Zones6 =[] Zones7 =[] Zones8 =[] Zones9 = [] Zones10= [] Zones11= [] entr=[]

fen01 = Tk() fen01.title(“calculs matriciels”) chaine1 = Label (fen01, text = “introduisez la dimension de la matrice carree :”) chaine1.grid(row =0) entr1= Entry(fen01) entr1.grid(row =0, column =1) bou1=Button(fen01,text='operations matricielles',command=opentop) bou1.grid(row=2,column=1) fen01.mainloop()

</sxh>