```
# -*- coding: utf-8 -*-

 ###############################################################################
 #  Logiciel IHMaffichageGraphes d'affichage de graphiques                     #
 #  Copyright INRA, février 2006                                               #
 ###############################################################################

###############################################################################
# Licence :
#
# Le logiciel IHMaffichageGraphes propose et gère une interface permettant
# de configurer/paramétrer des représentations graphiques, puis affiche à
# l'écran les tracés demandés.
#
# Ce logiciel est régi par la licence CeCILL soumise au droit français et
# respectant les principes de diffusion des logiciels libres. Vous pouvez
# utiliser, modifier et/ou redistribuer ce programme sous les conditions
# de la licence CeCILL telle que diffusée par le CEA, le CNRS et l'INRIA
# sur le site "http://www.cecill.info".
#
# En contrepartie de l'accessibilité au code source et des droits de copie,
# de modification et de redistribution accordés par cette licence, il n'est
# offert aux utilisateurs qu'une garantie limitée.  Pour les mêmes raisons,
# seule une responsabilité restreinte pèse sur l'auteur du programme,  le
# titulaire des droits patrimoniaux et les concédants successifs.
#
# A cet égard  l'attention de l'utilisateur est attirée sur les risques
# associés au chargement,  à l'utilisation,  à la modification et/ou au
# développement et à la reproduction du logiciel par l'utilisateur étant
# donné sa spécificité de logiciel libre, qui peut le rendre complexe à
# manipuler et qui le réserve donc à des développeurs et des professionnels
# avertis possédant  des  connaissances  informatiques approfondies.  Les
# utilisateurs sont donc invités à charger  et  tester  l'adéquation  du
# logiciel à leurs besoins dans des conditions permettant d'assurer la
# sécurité de leurs systèmes et ou de leurs données et, plus généralement,
# à l'utiliser et l'exploiter dans les mêmes conditions de sécurité.
#
# Le fait que vous puissiez accéder à cet en-tête signifie que vous avez
# pris connaissance de la licence CeCILL, et que vous en avez accepté les
# termes.
#
# Les fichiers Licence_CeCILL_V2-fr.txt et Licence_CeCILL_V2-en.txt
# ci-joints fournissent le texte de la licence, dans sa version 2,
# en langue française et en langue anglaise.
#
###############################################################################

###############################################################################
#
# Fichier     : affgraphs.py
#
# Auteur(s)   : Nathalie Rousse, Nathalie.Rousse@toulouse.inra.fr
#               de l'INRA - Institut National de la Recherche Agronomique -
#               ( UMR ARCHE / Plate-forme INRA-ACTA-ICTA ).
#
# Description :
#
# Ce fichier contient le code source du logiciel d'affichage de graphiques
# IHMaffichageGraphes écrit en langage Python, ainsi que sa documentation
# sous forme de commentaires.
#
# Le logiciel IHMaffichageGraphes propose et gère une interface permettant
# de configurer/paramétrer des représentations graphiques, puis affiche à
# l'écran les tracés demandés.
#
###############################################################################
# Historique :
#
```

```
# 03/02/06, Nathalie Rousse : création du fichier.
#
# => v060207
#
# 21/09/07, Nathalie Rousse : modification (pour corriger erreur à l'exécution) : aj
out de la ligne "coding: utf-8" en debut de fichier.
#
# => v070921
#
################################################################################


################################################################################
# Guide d'utilisation
#
# Prealable requis : Ce logiciel est ecrit en langage Python.
# Pour executer ce logiciel, il faut avoir installe un interpreteur Python.
#
# Par exemple si on travaille sous Windows, Python 2.2 fait partie de Cygwin,
# environnement pour Windows. Cygwin (cf "http://cygwin.com") s'execute
# au-dessus de Windows, ressemble par bien des aspects a Linux, Unix.
#
# Commande en ligne d'execution du logiciel : "python affgraphs.py"
#
################################################################################
# Specification
#
# l'IHM est composee de 3 fenetres :
#
# - Fenetre "Configuration" : permet a l'utilisateur de configurer/parametrer
#   les traces representes dans "Representation graphique"
#
# - Fenetre "Representation graphique" : affiche les traces demandes par
#   l'utilisateur dans "Configuration"
#
# - Fenetre "Legende" : affiche les legendes des traces de "Representation
#   graphique"
#
#####
#
# La fenetre "Configuration" permet de choisir :
#
# - les entrees (points des courbes a tracer) :
#   - le nom du fichier d'ou seront extraites les abscisses des points traces
#   - le nom du fichier d'ou seront extraites les ordonnees des points traces
#   Ces 2 fichiers sont recherches par le logiciel dans le repertoire courant,
#   ou ils doivent obligatoirement se situer (en aide au choix, la liste des
#   fichiers du repertoire courant est affichee dans la fenetre).
#
# - la configuration de la representation graphique :
#   - les 4 bornes des axes des coordonnees
#   - le point au niveau duquel sont representes les axes
#   - les pas de graduation des axes
#
#####
#
# Les fenetres  "Representation graphique" et "Legende" :
#
# A chaque fois que l'utilisateur commande "Tracer" (bouton de la fenetre
# "Configuration") : la derniere courbe que l'utilisateur a defini (en
# donnant les "fichiers des coordonnees" dans la fenetre "Configuration")
# est tracee sur la "Representation graphique". La legende associee a la
# courbe est affichee dans la fenetre "Legende"; avec changement de couleur
# a chaque trace de courbe (cf "(a)").
#
# A chaque fois que l'utilisateur commande "Effacer" (bouton de la fenetre
# "Configuration") : toutes les courbes de la "Representation graphique" et
# toutes les legendes de la fenetre "Legende" sont remises a zero/effacees.
```

```
#
#####
#
# "Constantes"
#
# (a) : Apparence/couleurs :
# Les couleurs avec lesquelles les courbes demandees et leur legende sont
# affichees sont prises dans le tableau ListeCouleurs.
#
################################################################################

from Tkinter import *
import os

################################################################################
# Parametres de gestion des couleurs avec lesquelles les courbes et legendes
# sont affichees
################################################################################
indiceCouleur = 0
# Il est suggere de completer ListeCouleurs a loisir
ListeCouleurs = [ 'black', 'red', 'green', 'brown', 'blue', 'cyan', '#FF00FF' ]

################################################################################
#
# Trace de presentation / identification du logiciel
#
################################################################################

dirname = os.getcwd()
filename = "affgraphs.py"

print "\n"
print "****************************************"
print "* "
print "*  Logiciel IHMaffichageGraphes d'affichage de graphiques "
print "* "
print "* "
print "Ce logiciel propose et gère une interface permettant de configurer/paramétrer
 des représentations graphiques, puis affiche à l'écran les tracés demandés."
print "* "
print "* "
print "****************************************"


################################################################################
#
# Les 3 fenetres principales de l'IHM
#
################################################################################

root = Tk()
root.title("Configuration")

graphe = Tk()
graphe.title("Representation graphique")

legende = Tk()
legende.title("Legende")

################################################################################
#
# Et fenetre(s) "temporaire(s)" d'affichage de messages textes, servant a
# envoyer des avertissements, des messages d'erreurs ...
#
# La methode "afficherMessage" affiche le texte "texteMessage" dans une
# fenetre intitulee "titre".
#
################################################################################
```

```
def afficherMessage( titre, texteMessage ) :
        message = Tk()
        message.title( titre )
        miniFram = Frame( message )
        Label( miniFram, text = texteMessage ).pack( side=LEFT )
        miniFram.pack( side=TOP )


##############################################################################
#
# Creation/presentation de la fenetre "Configuration"
#
##############################################################################

#
# Dans la fenetre "Configuration",
# zone de choix des noms des 2 fichiers d'entree des coordonnees (des points
# de la courbe a tracer)
#

fram = Frame( root )
Label( fram, text='************ CHOIX DES FICHIERS DES COORDONNEES ************ ' ).
pack( side=LEFT )
fram.pack( side=TOP )

#
# Lignes de saisie des noms des 2 fichiers d'entree des coordonnees
#
fram = Frame( root )

# Fichier des abscisses
Label( fram, text='Fichier des abscisses :' ).pack( side=LEFT )
fichierAbscisses = Entry( fram )
fichierAbscisses.pack( side=LEFT, fill=BOTH, expand=1 )
buttSelFicAbsc = Button( fram, text='select' )
buttSelFicAbsc.pack( side=LEFT )

# Fichier des ordonnees
Label( fram, text='       ' ).pack( side=LEFT )
Label( fram, text='Fichier des ordonnees :' ).pack( side=LEFT )
fichierOrdonnees = Entry( fram )
fichierOrdonnees.pack( side=LEFT, fill=BOTH, expand=1 )
buttSelFicOrd = Button( fram, text='select' )
buttSelFicOrd.pack( side=LEFT )

# En aide a la saisie, liste des fichiers du repertoire courant
# dans une liste deroulante
s = Scrollbar(root)
L = Listbox(root)
s.pack( side=RIGHT, fill=Y )
L.pack( side=RIGHT, fill=Y )
s.config( command=L.yview )
L.config( yscrollcommand=s.set )
L.config( selectmode=SINGLE )
# commande de capture de la liste des fichiers du repertoire courant
dirlist = os.listdir( os.getcwd() )
L.delete( 0, END ) # raz de la zone de la liste
for i in range( len(dirlist) ) : L.insert( END, dirlist[i] )

fram.pack( side=TOP )

#
# Dans la fenetre "Configuration",
# zone de choix/configuration de representation graphique
# ("de la derniere a la premiere ligne")
# precision : les  unites sont les memes que celles des valeurs lues
# dans les fichiers des coordonnees
```

```
#

# Ligne de choix/commande des actions/operations
fram = Frame( root )
buttTracer = Button( fram, text='Tracer' )
buttTracer.pack( side=LEFT )
buttEffacer = Button( fram, text='Effacer' )
buttEffacer.pack( side=LEFT )
fram.pack( side=BOTTOM )

# Ligne de saisie des valeurs des graduations des axes
fram = Frame( root )
deltaxyValues = []
Label( fram, text= '   Graduations des axes :   ' ).pack( side=LEFT )
for label in 'deltaX', 'deltaY' :
        Label( fram, text= '   ' + label + ' :' ).pack( side=LEFT )
        editDeltaxy = Entry( fram, width=6 )
        editDeltaxy.pack( side=LEFT )
        deltaxyValues.append( editDeltaxy )
fram.pack( side=BOTTOM )

# Ligne de saisie des coordonnees de l'origine des axes
fram = Frame( root )
origineCoord = []
Label( fram, text= '   Origine :   ' ).pack( side=LEFT )
for label in 'X0', 'Y0' :
        Label( fram, text= '   ' + label + ' :' ).pack( side=LEFT )
        editOrigine = Entry( fram, width=6 )
        editOrigine.pack( side=LEFT )
        origineCoord.append( editOrigine )
fram.pack( side=BOTTOM )

# Ligne de saisie des valeurs des bornes
fram = Frame( root )
limites = []
for label in 'minX', 'maxX', 'minY', 'maxY' :
        Label( fram, text= '   ' + label + ' :' ).pack( side=LEFT )
        edit = Entry( fram, width=6 )
        edit.pack( side=LEFT )
        limites.append( edit )
fram.pack( side=BOTTOM )

fram = Frame( root )
Label( fram, text='************ CHOIX DE REPRESENTATION GRAPHIQUE ************' ).pa
ck( side=LEFT )
fram.pack( side=BOTTOM )


###############################################################################
#
# Creation/presentation de la fenetre "Legende"
#
# Il s'agit de la fenetre ou sont affichees les legendes des traces, dans la
# couleur adequate
#
###############################################################################

# Zone de texte, avec scrollbar
text = Text( legende )
s = Scrollbar( legende )
text.focus_set()
s.pack( side = RIGHT, fill = Y )
text.pack( side = LEFT, fill = Y )
s.config( command = text.yview )
text.config( yscrollcommand = s.set )

# Objets pour gerer les couleurs des legendes des traces
ListeTexteTags = []
```

```
ListeCouleurTags = []


################################################################################
#
# Creation/presentation de la fenetre "Representation graphique"
#
################################################################################

# Zone du graphe (canevas)
c= Canvas( graphe )
c.pack( side=TOP, fill=BOTH, expand=1 )
c.itemconfig( ALL )


################################################################################
#
#
# Les methodes invoquees par les differentes fenetres
#
#
################################################################################

################################################################################
#
# Methodes relatives a la saisie des noms des fichiers des coordonnees
#
################################################################################

#
# Lit dans la liste des fichiers le nom de celui qui est selectionne,
# et retourne ce nom de fichier.
#
def releverNomFichier() :
        nomFicSelectionne = [ L.get(x) for x in L.curselection() ]
        if len(nomFicSelectionne) > 0 :
                return nomFicSelectionne[0]
        else :
                return ""

#
# Action effectuee sur commande "select" du fichier :
# Recupere le nom du fichier selectionne dans la liste.
# Puis si effectivement un fichier est selectionne dans la liste, affiche son
# nom dans la zone du nom du fichier.
#

def majNomFicAbsc() :
        nomFic = releverNomFichier()
        if ( nomFic != "" ) :
                fichierAbscisses.delete( 0, END )
                fichierAbscisses.insert( END, nomFic )

def majNomFicOrd() :
        nomFic = releverNomFichier()
        if ( nomFic != "" ) :
                fichierOrdonnees.delete( 0, END )
                fichierOrdonnees.insert( END, nomFic )

################################################################################
#
# Methode relative a la saisie des bornes
#
################################################################################

#
# Ajuste et affiche les valeurs des 4 bornes X et Y
#
```

```
def minimax( valeurs = [ -50.0, 50.0, -50.0, 50.0 ] ):
        for i in range(4) :
                edit = limites[i]
                try : valeurs[i] = float( edit.get() )
                except : pass
                edit.delete( 0, END )
                edit.insert( END, '%.2f' % valeurs[i] )
        return valeurs

#
# Ajuste et affiche les coordonnees de l'origine des axes x0, y0
#
def origine( valeurs = [ 100.0, 0.5 ] ):
        for i in range(2) :
                editOrigine = origineCoord[i]
                try : valeurs[i] = float( editOrigine.get() )
                except : pass
                editOrigine.delete( 0, END )
                editOrigine.insert( END, '%.2f' % valeurs[i] )
        return valeurs

#
# Ajuste et affiche les valeurs de graduation des axes deltax, deltay
#
def deltaxy( valeurs = [ 20.0 , 0.2 ] ):
        for i in range(2) :
                editDeltaxy = deltaxyValues[i]
                try : valeurs[i] = float( editDeltaxy.get() )
                except : pass
                editDeltaxy.delete( 0, END )
                editDeltaxy.insert( END, '%.2f' % valeurs[i] )
        return valeurs

###############################################################################
#
# Methodes relatives au trace des courbes
#
# A chaque demande de "tracer", il s'agit de representer a l'ecran la courbe.
# La courbe est definie par ses points dont les coordonnees sont donnees dans
# les "fichiers des coordonnees".
#
# L'affichage necessite une conversion entre :
# Rreel, le repere "reel" (unites des valeurs contenues dans les fichiers) et
# Recran, le repere de l'ecran (unites d'affichage).
#
# Les points ont pour coordonnees :
# - (x,y) dans le repere Rreel : { [minx,maxx],[miny,maxy] }
# - (X,Y) dans le repere Recran : { [0,CX],[0,CY] }
# !!! l'axe des ordonnees [0,CY] du repere Recran est oriente vers le bas !!!
#
###############################################################################

#
# Convertit les coordonnees (x,y) relatives au repere Rreel
# en coordonnees (X,X) relatives au repere Recran
#
def convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, x, y ) :
        X = CX * ( x - minx ) / ( maxx - minx )
        Y = CY * ( - y + maxy ) / ( maxy - miny )
        return X,Y

#
# Trace un encadrement de la zone d'affichage,
# puis trace les axes et graduations des axes conformement a la configuration
# (xO, yO) et (deltax, deltay) demandee par l'utilisateur.
#
def tracerCadreEtAxes() :
```

```
        # valeurs des bornes des axes des coordonnees (relatives a Rreel)
        minx, maxx, miny, maxy = minimax()
        # valeurs des coordonnees de l'origine des axes (relatives a Rreel)
        x0, y0 = origine()
        # valeurs de graduation des axes (relatives a Rreel)
        deltax, deltay = deltaxy()
        # les dimensions X et Y du canevas (taille du repere Recran)
        CX = c.winfo_width()
        CY = c.winfo_height()

        # Trace du cadre
        c.create_rectangle( 0+2,0+2, CX-4,CY-4, outline='red'  )

        # Trace des axes et graduations
        X0,Y0 = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, x0, y0 )
        # axe des abscisses
        z = c.create_line( 0+2,Y0, CX-4,Y0 )
        c.itemconfig( z, fill='red' )
        # axe des ordonnees
        z = c.create_line( X0,0+2, X0,CY-4 )
        c.itemconfig( z, fill='red' )
        # graduation des abscisses
        x = x0 + deltax
        while ( x < maxx ) :
                X,Y = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, x, maxy
)
                z = c.create_line( X,0+2, X,CY-4 )
                c.itemconfig( z, fill='darkgrey' )
                x = x + deltax
        x = x0 - deltax
        while ( x > minx ) :
                X,Y = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, x, maxy
)
                z = c.create_line( X,0+2, X,CY-4 )
                c.itemconfig( z, fill='darkgrey' )
                x = x - deltax
        # graduation des ordonnees
        y = y0 + deltay
        while ( y < maxy ) :
                X,Y = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, maxx, y
)
                z = c.create_line( 0+2,Y, CX-4,Y )
                c.itemconfig( z, fill='darkgrey' )
                y = y + deltay
        y = y0 - deltay
        while ( y > miny ) :
                X,Y = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY, maxx, y
)
                z = c.create_line( 0+2,Y, CX-4,Y )
                c.itemconfig( z, fill='darkgrey' )
                y = y - deltay
        return minx, maxx, miny, maxy, CX, CY, x0, y0, deltax, deltay

#
# Trace une croix centree sur le point de coordonnees (X,Y) relativement a Recran
#
def tracerCroix( X, Y ) :
        c.create_line(X-2,Y-2,X+2,Y+2 )
        c.create_line(X+2,Y-2,X-2,Y+2 )

#
# Action effectuee sur commande "tracer" :
# Trace le cadre, les axes et graduations des axes (par appel de tracerCadreEtAxes)
# puis trace la courbe, avec des croix sur les points de la courbe.
#
def tracer():
# Rappel :
# (x,y) coordonnees dans Rreel repere { [minx,maxx],[miny,maxy] }
```

```
# (X,Y) coordonnees dans Recran repere { [0,CX],[0,CY] }
# !!! l'axe des ordonnees [0,CY] du repere Recran est oriente vers le bas !!!

        #
        # Recupere les coordonnees (x,y) des points de la courbe
        # dans les 2 "fichiers des coordonnees"
        #

        # lit dans le fichier "nomFic" la liste des coordonnes "coord"
        # (selon l'appel, il s'agira de la liste des abscisses ou de la liste des or
donnees)
        def lire( nomFic, coord ) :
                mode='r'
                bufsize=-1
                size=-1
                try :
                        # tentative d'ouverture du fichier
                        fic = file ( nomFic, mode, bufsize )
                except IOError :
                        afficherMessage( "Erreur", "Probleme d'ouverture du fichier
%s (code erreur : IOError). Le fichier %s est-il bien present dans le repertoire cou
rant ?" % ( nomFic, nomFic ) )
                else :
                        try :
                                # lecture du contenu du fichier
                                ListeCar = fic.readlines( size )
                        except :
                                afficherMessage( "Erreur", "Probleme de lecture dans
 le fichier %s (methode 'readlines')" % nomFic )
                        else :
                                # conversion des valeurs "texte" lues en nombres ree
ls ranges dans "coord"
                                for i in range( len(ListeCar) ) :
                                        coord.append( float( ListeCar[i] ) )

        # Lecture des abscisses dans le fichier approprie
        abscisses = []
        nomFicAbsc = fichierAbscisses.get()
        if ( nomFicAbsc != "" ) :
                lire( nomFicAbsc, abscisses)
        else :
                afficherMessage( "Erreur", "Probleme avec le fichier des abscisses :
 il n'a pas ete defini" )

        # Lecture des ordonnees dans le fichier approprie
        ordonnees = []
        nomFicOrd = fichierOrdonnees.get()
        if ( nomFicOrd != "" ) :
                lire( nomFicOrd, ordonnees)
        else :
                afficherMessage( "Erreur", "Probleme avec le fichier des ordonnees :
 il n'a pas ete defini" )

        # Trace de debug :
        # print "datas '%s' dans [ %f, %f ] ; datas '%s' dans [ %f, %f ]" % ( nomFic
Absc, min(abscisses), max(abscisses), nomFicOrd, min(ordonnees), max(ordonnees) )

        #
        # Verification sur les coordonnee Recupere les coordonnees (x,y) des points
de la courbe
        # dans les 2 "fichiers des coordonnees"
        #
        # Quelques verifications pour s'assurer que les coordonnees des points
        # ont ete lues correctement dans les fichiers
        # Calcul du nombre de points

        if ( len(abscisses) <= 0 ) :
                afficherMessage( "Erreur", "Probleme (bloquant) de lecture des absci
```

```
sses dans le fichier des abscisses.\n La courbe n'est pas tracee." )
        if ( len(ordonnees) <= 0 ) :
                afficherMessage( "Erreur", "Probleme (bloquant) de lecture des ordon
nees dans le fichier des ordonnees.\n La courbe n'est pas tracee." )

        nbPoints = min( len(abscisses), len(ordonnees) )

        if ( (nbPoints > 0 ) and ( len(abscisses) != len(ordonnees) ) ) :
                afficherMessage( "Avertissement", "probleme (non bloquant) de cohere
nce :\n il n'a pas ete lu autant d'abscisses dans le fichier des abscisses que d'ord
onnees dans le fichier des ordonnees.\n La courbe est tracee avec les points ayant a
 la fois une abscisse et une ordonnee definies dans les fichiers des coordonnees" )

        # Affichage du trace et de la legende seulement si les coordonnees des point
s de la courbe
        # ont ete lues correctement dans les fichiers
        if ( nbPoints > 0 ) :

                # Gestion de la couleur des traces et legendes :
                # La nouvelle couleur pour le nouveau trace est ListeCouleurs[indice
Couleur]
                # (indiceCouleur est incremente - modulo taille de ListeCouleurs - a
 chaque nouveau trace)
                global indiceCouleur
                indiceCouleur = ( indiceCouleur + 1 ) % len( ListeCouleurs )

                #
                # Gestion des couleurs des legendes et affichage des legendes (dans
"Legende")
                #

                # memorise la legende du nouveau trace
                ListeCouleurTags.append( ListeCouleurs[indiceCouleur] )
                ListeTexteTags.append( "'%s' :  [ %f, %f ] ;    '%s' :  [ %f, %f ] \
n" % ( nomFicAbsc, min(abscisses), max(abscisses), nomFicOrd, min(ordonnees), max(or
donnees) ) )

                # affiche et colore l'ensemble des legendes
                ListeNomTags = []
                text.delete( '1.0', END )
                for i in range( len(ListeTexteTags) ) :
                        ListeNomTags.append( str(i) )
                        text.insert( END, ListeTexteTags[ i ] )
                        text.tag_add( ListeNomTags[ i ], '%d.0'%(i+1), '%d.end'%(i+1
) )
                        text.tag_config( ListeNomTags[ i ], foreground=ListeCouleurT
ags[i] )
                text.pack( side = BOTTOM )

                #
                # Affiche le graphe (dans "Representation graphique")
                #


                # affiche le repere et le cadre
                minx, maxx, miny, maxy, CX, CY, x0, y0, deltax, deltay = tracerCadre
EtAxes()

                # Calcule les coordonnees des points de la courbe, relativement au r
epere Recran
                coordsXY = []
                for i in range( nbPoints ) :
                        x = abscisses[i]
                        y = ordonnees[i]
                        X,Y = convertirCoordonnees( minx, maxx, miny, maxy, CX, CY,
x, y )
                        # trace d'une croix au niveau du point
                        tracerCroix( X, Y )
```

```
                        # ajout du point dans la "ligne"
                        coordsXY.append( X )
                        coordsXY.append( Y )

                # Trace de la ligne
                z = c.create_line( *coordsXY )
                c.itemconfig( z, fill = ListeCouleurs[indiceCouleur] )

#
# Action effectuee sur commande "effacer" :
# Efface les courbes et les legendes
#
def effacer() :

        c.delete( ALL )

        # raz des legendes
        text.delete( '1.0', END )
        global ListeTexteTags
        global ListeCouleurTags
        ListeTexteTags = []
        ListeCouleurTags = []

        tracerCadreEtAxes()


################################################################################
#
# Associations des actions (commandes) aux boutons
#
################################################################################

#
# Commandes de la fenetre "Configuration"
#

# Commandes relatives aux fichiers des coordonnees
buttSelFicAbsc.config( command=majNomFicAbsc )
buttSelFicOrd.config( command=majNomFicOrd )

# Commandes relatives a la representation graphique : tracer/effacer courbes
buttTracer.config( command=tracer )
buttEffacer.config( command=effacer )

# Il n'y a pas de commande dans la fenetre "Representation graphique"

# Il n'y a pas de commande dans la fenetre "Legende"


################################################################################
#
# Main
#
################################################################################

# Initialisation par defaut
minimax( [ -10.0, 210.0, -0.1, 1.3 ] )
origine( [ 100.0, 0.5 ] )
deltaxy( [ 20.0 , 0.2 ] )

root.mainloop()

################################################################################
#
# Fin
#
################################################################################
```