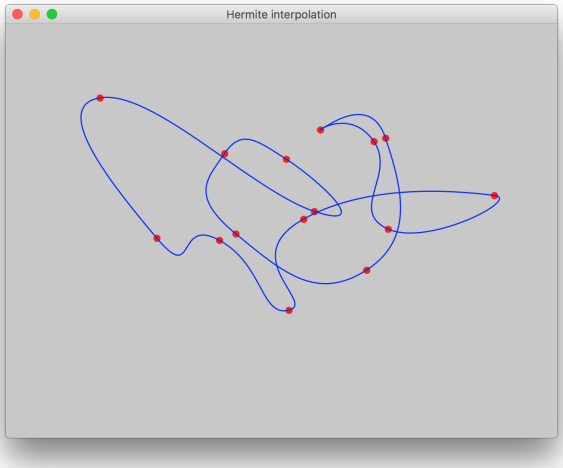
Python 18-19 for dummies : problème 2
Interpolation d'Hermite

L'interpolation d'Hermite définie sur l'intervalle $[X_{i-1}, X_i]$ est définie par l'expression :

$$\begin{aligned} u^{h_i}(x) &= U_i \frac{3h_i s^2 - 2s^3}{h_i^3} + U_{i-1} \frac{h_i^3 - 3h_i s^2 + 2s^3}{h_i^3} \\ &\quad + U'_i \frac{s^2(s-h_i)}{h_i^2} + U'_{i-1} \frac{s(s-h_i)^2}{h_i^2} \end{aligned}$$

où $s = x - X_{i-1}$, tandis que U_i et U'_i sont la valeur et la dérivée de la fonction en $x = X_i$. Finalement, on définit les écarts $h_i = X_i - X_{i-1}$. Cela permet d'obtenir une interpolation cubique par morceaux de classe C_1 d'une fonction $u(x)$ dont on connaît les ordonnées U_i et les dérivées U'_i aux abscisses X_i.

Il s'agit de réaliser une interpolation cubique par morceaux d'Hermite d'une courbe fermée $(x(t), y(t))$ qui sera définie à partir de n points (X_k, Y_k) pour les instants $T_k = k$ avec $k = 1, \ldots, n$. Pour obtenir une courbe fermée, on duplique le premier point : $(X_{n+1}, Y_{n+1}) = (X_1, Y_1)$.



Pour obtenir une estimation des dérivées en un point T_k, on écrit

$$(X'_k, Y'_k) \approx \left(\frac{X_{k+1} - X_{k-1}}{2}, \frac{Y_{k+1} - Y_{k-1}}{2} \right)$$

en adaptant adéquatement les indices au début et à la fin de la courbe...

1. Ecrire une fonction `uh = hermite(x,X,U,dU)` qui détermine les valeurs en x de l'interpolation d'Hermite cubique par morceaux pour les valeurs U_i et les pentes U'_i aux points X_i.

2. Comme on est vraiment trop gentil, nous vous fournissons un canevas contenant déjà une petite partie de la solution car elle calcule l'interpolation linéaire par morceaux....

```
def hermite(x,X,U,dU):

  i = zeros(len(x),dtype=int)
  for j in range(1,len(X)-1):
    i[X[j]<=x] = j

  return (U[i]   * (x-X[i+1]) / (X[i]-X[i+1]) +
          U[i+1] * (x - X[i]) / (X[i+1]-X[i])   )
```

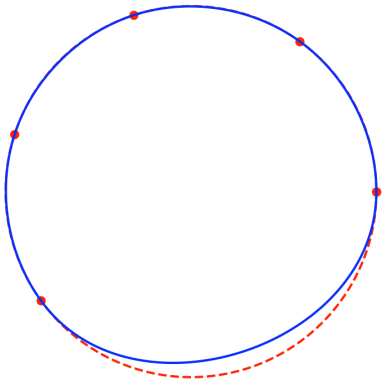
   Ce code contient une partie du code qui permet d'identifier pour chaque élément `x[index]` du tableau `x` l'intervalle `i[index]` à considérer ! Cela permet ensuite de bêtement calculer l'interpolation linéaire en prenant les deux points qui encadrent cet intervalle... Notons que, de manière vectorisée, tout cela fait est avec une unique ligne de code :-)

3. Ensuite pour tester, votre programme, on vous a vous fourni un tout petit programme simple `hermiteTest.py` pour interpoler un cercle.

```
n = 4;
T = arange(0,3*pi/2,3*pi/(2*(n+1)))
T = append(T,[2*pi])

t = linspace(T[0],T[-1],1000)
X  =  cos(T); Y  =  sin(T);
dX = -sin(T); dY =  cos(T);

plt.plot(X,Y,'.r',markersize=10)
plt.plot(cos(t),sin(t),'--r')
plt.plot(hermite(t,T,X,dX),hermite(t,T,Y,dY),'-b')
plt.axis("equal"); plt.axis("off")
plt.show()
```



4. Finalement, nous avons aussi fourni un programme un peu plus rigolo `hermiteTestFun.py` qui permet de définir plein de points en cliquant sur la figure. Un double click permet d'obtenir la courbe fermée obtenue par l'interpolation d'hermine. Pour les petits futés et curieux, le listing de ce programme est donné à la fin de cet énoncé.

5. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera incluse dans un unique fichier `hermite.py`, sans y adjoindre le programme de test fourni ! Cette fonction devra être soumise via le web avant le mercredi 27 février à 23h59 : ce travail est individuel et sera évalué. Pour rappel, toutes vos soumissions seront systématiquement analysées par un logiciel anti-plagiat. Faites vraiment votre programme seul... :-) Maggie veille au grain !

6. Il est possible de calculer efficacement les splines d'Hermite en utilisant la procédure décrite ci-après. Pourquoi cette manière de faire est-elle la plus efficace ? Attention, c'est vraiment la dernière ligne du développement qu'il faut idéalement programmer !

Ecrire efficacement les splines d'Hermite...

$$
\begin{aligned}
u^{h_i}(x) &= U_i \frac{3h_is^2-2s^3}{h_i^3} + U_{i-1}\frac{h_i^3-3h_is^2+2s^3}{h_i^3} + U_i'\frac{s^2(s-h_i)}{h_i^2} + U_{i-1}'\frac{s(s-h_i)^2}{h_i^2} \\
&= U_i \frac{3h_is^2-2s^3}{h_i^3} + U_{i-1}\frac{h_i^3-3h_is^2+2s^3}{h_i^3} + U_i'\frac{s^3-h_is^2}{h_i^2} + U_{i-1}'\frac{s^3-2h_is^2+h_i^2s}{h_i^2} \\
&\downarrow \quad \text{En regroupant les termes par puissances de } s \\
&= U_{i-1} + sU_{i-1}' + s^2 \underbrace{\left(\frac{3(U_i-U_{i-1})}{h_i^2} - \frac{(U_i'+2U_{i-1}')}{h_i}\right)}_{A_i} + s^3 \underbrace{\left(\frac{-2(U_i-U_{i-1})}{h_i^3} + \frac{(U_i'+U_{i-1}')}{h_i^2}\right)}_{B_i}
\end{aligned}
$$

$$
\begin{aligned}
u^{h_i}(x) &= U_{i-1} + s\,U_{i-1}' + s^2\,A_i + s^3\,B_i \\
&= U_{i-1} + s\,\Big(U_{i-1}' + s\,\Big(A_i + s\,B_i\Big)\Big)
\end{aligned}
$$

Un petit programme interactif avec `matplotlib`...

Pour les petits curieux, voici l'implémentation de `hermiteTestFun.py` pour définir de manière magique une courbe à interpoler...

```
import matplotlib
from matplotlib import pyplot as plt
from numpy import *
from hermite import hermite

# ====================== callback pour les événements avec la souris ======
#
#  Observer la gestion distincte du clic simple et double :-)
#  Apres un événement, on redessine la figure avec draw()
#

def mouse(event):
  global X,Y,n
  if (event.dblclick):
    X  = append(X,[X[0]])
    Y  = append(Y,[Y[0]])
    dX = array([X[1]-X[n-1],*(X[2:n+1]-X[0:n-1]),X[1]-X[n-1]])
    dY = array([Y[1]-Y[n-1],*(Y[2:n+1]-Y[0:n-1]),Y[1]-Y[n-1]])
    T  = arange(0,n+1)
    t  = arange(0,n+0.001,0.001)
    x  = hermite(t,T,X,dX)
    y  = hermite(t,T,Y,dY)
    plt.plot(x,y,'-b')
    X,Y = [],[]; n = 0
  else :
    x = event.xdata
    y = event.ydata
    if (x != None and y != None) :
      n = n + 1
      X = append(X,[x])
      Y = append(Y,[y])
      print("New data : " + str(x) + "," + str(y))
      plt.plot([x],[y],'.r',markersize=10)
  fig.canvas.draw()


# ============================== mainProgram ================================

matplotlib.rcParams['toolbar'] = 'None'
matplotlib.rcParams['lines.linewidth'] = 1
plt.rcParams['figure.facecolor'] = 'silver'

X,Y = [],[]; n = 0
fig = plt.figure("Hermite interpolation")
fig.canvas.mpl_connect('button_press_event',mouse)
plt.ylim((0,1)); plt.xlim((0,1.3)); plt.axis("off")

plt.show()
```