Rappel du principe de la méthode d'Euler :

Soit $t \to y(t)$ la solution de l'équation différentielle du premier ordre :
$y'(t) = F(y(t))$ avec $t \in [a,b]$, avec la condition initiale $y(a) = y_0$ consistant en la donnée à l'instant $t_0 = a$ de la position initiale y_0 .
Il est hors de question de représenter en machine chaque $y(t)$ pour $t \in [a,b]$ (car cet intervalle contient une infinité de valeurs). On va discrétiser le segment et se contenter de calculer (de manière approchée) les valeurs de la fonction solution y aux points d'une subdivision régulière de $[a,b]$, constituée des points $t_i = a + i\frac{b-a}{n}$ pour $0 \leqslant i \leqslant n$. Ces $n+1$ points délimitent n intervalles $[t_i, t_{i+1}]$ de même longueur $h = \frac{b-a}{n}$ qu'on appelle le pas de la subdivision.

Il s'agit de construire une suite finie $y_0 ; y_1 ; ... ; y_n$ approchant les valeurs exactes
$y(t_0) ; y(t_1) ; ... ; y(t_n)$.

- Construction de y_1.

Au point t_0, la solution $t \to y(t)$ est approchée par la fonction affine $t \to y(t_0) + (t - t_0)y'(t_0)$.
On construit y_1 en posant :
$y_1 = y(t_0) + (t_1 - t_0)y'(t_0)$
$y_1 = y_0 + hF(y_0)$.

- Construction de y_2.

En première approximation $y(t_2)$ est approchée affinement par :
$y(t_1) + (t_2 - t_1)y'(t_1) = y(t_1) + (t_2 - t_1)F(y(t_1))$.
En remplaçant : $y(t_1)$ par y_1 et $F(y(t_1))$ par $F(y_1)$, on pose :
$y_2 = y_1 + hF(y_1)$.

- En itérant ce procédé, on définit la suite $y_0 ; y_1 ; ... ; y_n$ par :

$$\begin{aligned} &y_0 \text{ donné} \\ &y_1 = y_0 + hF(y_0) \\ &y_2 = y_1 + hF(y_1) \\ &\vdots \\ &y_{i+1} = y_i + hF(y_i) \\ &\vdots \\ &y_n = y_{n-1} + hF(y_{n-1}) . \end{aligned}$$

On peut de même gérer des systèmes différentiels, par exemple de deux équations portant sur deux fonctions inconnues y et z. Un tel système est de la forme $\begin{cases} y' = F(y,z,t) \\ z' = G(y,z,t) \end{cases}$. La méthode précédente s'adapte en posant $\begin{cases} y_{i+1} = y_i + hF(y,z,t) \\ z_{i+1} = z_i + hG(y,z,t) \end{cases}$.

En particulier, toute équation différentielle d'ordre 2 se ramène à un tel système. En effet, une telle équation s'écrit $y'' = H(y,y',t)$, et en posant $z = y'$, l'équation différentielle se traduit par le système $\begin{cases} y' = z \\ z' = H(y,z,t) \end{cases}$.

Un autre point de vue (qui donne le même résultat) consiste à considérer que la méthode d'Euler peut s'appliquer à des vecteurs. Ici on considère le vecteur $Y = \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} y \\ y' \end{pmatrix}$ qui vérifie l'équation différentielle $Y' = S(Y) = \begin{pmatrix} F(y,z,t) \\ G(,y,z,t) \end{pmatrix}$

1. Schéma d'Euler pour la résolution approchée une équation différentielle

Rappels des consignes universelles de début de TP :
Partir d'un script (fichier .py) vierge. L'enregistrer en lui donnant un nom (et une adresse...)
Au tout début de votre script, importer les bibliothèques dont nous aurons besoin :

```
import numpy as np
from matplotlib import pyplot as plt
```

1.a) Écrire une fonction `eulerAut(F,a,b,n,y0)` qui résout de manière approchée l'équation différentielle ***y'=F(y)*** sur l'intervalle *[a,b]* avec la condition initiale *y(a)=y0* en subdivisant celui-ci en *n* sous intervalles, c'est à dire avec un pas *(b-a)/n*, et qui représente graphiquement la solution approchée obtenue. Attention, cette subdivision est formée de *n*+1 points (extrémités comprises).

Pour obtenir la subdivision de l'axe des abscisses donnée par `tx[k]`=*a+k(b-a)/n*, on pourra utiliser `tx=np.linspace(a,b,n+1)`. On peut aussi envisager `np.arange(a,b,h)`

Pour obtenir le tableau des images `ty` (tel que pour $0 \leq k \leq n$, `ty[k]` soit une approximation de *y*(`tx[k]`), on pourra d'abord créer un tableau rempli de 0 par `ty=np.zeros(n+1)` puis y mettre les bonnes valeurs en commençant par `ty[0]=y0` puis en utilisant la relation de récurrence :
`ty[k+1]=ty[k]+F(ty[k])*(b-a)/n` pour calculer les valeurs de proche en proche.

b) Écrire une fonction `eulerNonAut(F,a,b,n,y0)` qui résout cette fois ***y'=F(y,t)***.

2. Application à des équations différentielles linéaires à coefficients constants

2. a) Résoudre de manière approchée sur l'intervalle [0,5] l'équation différentielle ***y'=-y*** avec la condition initiale *y(0)=1*. On superposera le graphe de la solution approché à celui de la solution théorique $y(t)=e^{-t}$.
Faire varier le pas. Qu'observe-t-on ?
Où l'erreur est elle la plus importante ? Pourquoi ?
(gag : qu'obtient-t-on avec un pas valant 1 ? avec un pas supérieur à 1 ? Expliquer)

b) Même consigne avec l'équation différentielle ***y'=y***.
Que se passe-t-il si on résout en temps plus long, par exemple sur [0,10] ou [0,20] (attention pour garder un pas constant il faut ajuster le nombre de points).

c) Même consigne avec l'équation différentielle non autonome $y'=y-2e^{-t}$. Comparer avec la solution théorique $y(t)=e^{-t}$. Expliquer en s'appuyant sur la solution générale $y(t)=Ae^{t}+e^{-t}$.

3. Application à des équations différentielles non linéaires

3) On va résoudre de manière approchée l'équation différentielle $y'=r\left(1-\frac{y}{K}\right)y$ associée à un modèle d'évolution de population appelé le modèle logistique (le taux de croissance de la population est *r*, tant que la population est faible, mais diminue quand la population augmente (car les ressources ne sont pas illimitées) et tend vers 0 lorsque la population approche du nombre maximum *K* d'individus que peut nourrir le milieu).

a) Que fait le code suivant ? L'exécuter ensuite pour vérifier.

```
def fabrique_F_logistique(r,K):
    def F_log(y):
        return(r*y*(1-y/K))
    return(F_log)

F=fabrique_F_logistique(0.1,100)
euler(F,0,150,1000,5)
```

b) Superposer avec le graphe de la solution théorique, de la forme $y(t)=\frac{K}{1+Ae^{-rt}}$ avec A une constante à déterminer à partir de la condition initiale.

c) Observer ce qui se passe lorsqu'on fait varier les paramètres du modèle.

d) Même chose avec le modèle de Gompertz qui donne l'équation différentielle $y'=r\ln\left(\frac{K}{y}\right)$, de solution $y(t)=Ke^{-Be^{-rt}}$ avec B une constante positive à déterminer en fonction des conditions initiales.

4. Quelques équations différentielles d'ordre 2

4.a) Résoudre avec la méthode d'Euler les équations différentielles suivantes et tracer $y(t)$ en fonction de t.

- un oscillateur harmonique : $y''+y=0$ avec les CI : $y(0)=1$ et $y'(0)=0$.
- un oscillateur harmonique amorti : $y''+ky'+y=0$ avec les CI : $y(0)=1$ et $y'(0)=0$ pour $k>0$ petit.
- équation précédente mais avec $k<0$: l'amplitude des oscillations croît !

On testera différentes valeurs du paramètre k.

Expliquer pourquoi le comportement observé diffère autant du comportement théorique lorsque le facteur d'amortissement est nul ou petit (et positif).

b) Tracer cette fois le portrait de phase : on représentera les points de coordonnées $(y(t),y'(t))$.

c) Reprendre le travail précédent avec le pendule simple (non amorti) d'équation $y''+\sin(y)=0$.
On le résout souvent de manière approchée dans le cas des petites oscillations (l'hypothèse y petit permettant d'approximer $\sin(y)$ par y), mais ici on peut travailler avec la vraie équation différentielle pour la résolution numérique.

5. On n'est pas obligé de réinventer la roue à chaque fois

Tester :

```
from scipy.integrate import odeint,quad
help(odeint)    ordinary differential equation integrate
help(quad)      pour le calcul approché d'intégrales (par une méthode de quadrature)
```

Le schéma d'Euler est une méthode rudimentaire, il en existe de plus efficaces (comme Runge-Kutta d'ordre 2 ou d'ordre 4) mais plus compliquées à présenter et à mettre en oeuvre, qui pour le même nombre de points donnent une solution approchée plus proche de la vraie solution... Il ne faut pas s'étonner si les bibliothèques les exploitent (et délaissent Euler).

Tester ces méthodes d'intégration avec des équations résolues précédemment en comparant aux solutions théoriques et à celles fournies par Euler. On pourra évaluer non seulement la précision, mais aussi les temps de calcul.