Python - TP 7
Équations différentielles

La première partie de ce TP étudie une situation pratique ainsi que deux modélisations mathématiques : à l'aide d'une suite, ou à l'aide d'équations différentielles obtenues de deux façons différentes.
Le seconde partie, indépendante de la première, permet de découvrir une méthode de résolution numérique d'équations différentielles linéaires d'ordre 1.

I Mesure de température : statistiques et équations différentielles

La température ambiante, notée T_a, est de ..... Relevons ensuite la température d'une tasse de café en fonction du temps. Pour tout $t \in \mathbb{R}^+$, on note $T(t)$ la température du café au bout de t minutes.

temps (min)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
température (°C)																					

Les données sont récapitulées dans une liste, nommée `T` de sorte que pour tout $i \in [\![0, \texttt{len(T)} - 1]\!]$, `T[`$i$`]` est la température du café au bout de i minutes.

☞ Rappel...
On rajoute l'option `label="..."` dans la commande `plt.plot()` puis la commande `plt.legend()` avant `plt.show()`.

1. Représenter ces données en fonction du temps sur un graphique (**graphique 1**), en légendant le graphique.

2. A quel type de suite cette représentation graphique nous fait-elle penser ?

3. Créer une liste `deltatemp` dont le terme de rang i donne l'écart entre la température à l'instant i et la température ambiante.

4. Créer une liste `tauxperte` dont le terme de rang i donne le taux de perte entre l'instant $i+1$ et l'instant i.

5. Représenter les données de `tauxperte` en fonction de `deltatemp` (**graphique 2**).

On commentera ensuite la loi de refroidissement de Newton :

"le taux de perte de chaleur d'un corps est proportionnel à la différence de température entre ce corps et le milieu ambiant"

♣ Indication...
Cela revient à regarder si, pour tout entier naturel n, la quantité $T(n+1) - T(n)$ est proportionnelle à $T(n) - T_a$.

6. Le but de cette question est de déterminer deux droites approximant ce nuage de points.

6.a. **6.a.i.** Calculer la moyenne des nombres $\frac{\texttt{tauxperte[i]}}{\texttt{deltatemp[i]}}$. On notera `a1` ce nombre.

6.a.ii. Représenter la droite d'équation $y = \texttt{a1}x$ sur **graphique 2**.

6.b. Soient $s = (s_1, s_2, ..., s_n)$ et $t = (t_1, t_2, ..., t_n)$ deux séries statistiques.

6.b.i. On appelle **covariance de s et t** le nombre, noté $\text{Cov}(s,t)$, défini par :

$$\text{Cov}(s,t) = \frac{1}{n}\sum_{i=1}^{n}(s_i - \overline{s})(t_i - \overline{t})$$

où $\overline{s}$ et $\overline{t}$ désignent les moyennes respectives de s et t.
Calculer la covariance des séries statistiques `deltatemp` et `tauxperte`. On la notera `covariance`.

> ☞ **Rappels...**
> La commande `numpy.mean()` permet de calculer la moyenne d'une liste.
> La commande `numpy.std()` permet de calculer l'écart-type d'une liste.

6.b.ii. On appelle **droite des moindres carrés de la série t en fonction de la série s** la droite d'équation $y = ax + b$, où :

$$a = \frac{\text{Cov}(s,t)}{\sigma_s^2} \quad ; \quad b = \overline{t} - a\overline{s}$$

avec σ_s l'écart-type de la série statistique s.
Déterminer le coefficient directeur et l'ordonnée à l'origine de la droite des moindres carrés de `tauxperte` en fonction de `deltatemp`.

> ☞ **Pour info...**
> Voir cours de deuxième année pour des justifications...

6.b.iii. Représenter cette droite sur **graphique 2**.

6.c. Modélisation : $\forall n \in \mathbb{N},\ \text{T}(n+1) - \text{T}(n) =$ .....

Sous cette modélisation, le terme général de $(\text{T}_n)_{n\in\mathbb{N}}$ est donné par : $\forall n \in \mathbb{N},\ \text{T}(n) =$ .....

7. **7.a.** La première modélisation, obtenue à la question **6.a.**, nous permet d'obtenir en temps continu :

$$\frac{\text{T}(t+dt) - \text{T}(t)}{dt} \simeq \texttt{a1}(\text{T}(t) - \text{T}_a)$$

D'où, après passage à la limite quand $dt \to 0$, on obtient l'équation différentielle :

Déterminer alors l'unique solution de cette équation différentielle vérifiant $\text{T}(0) = 56,8$.

Représenter cette solution sur **graphique 1**.

7.b. La seconde modélisation, obtenue à la question **6.b.**, nous permet d'obtenir en temps continu :

$$\frac{\text{T}(t+dt) - \text{T}(t)}{dt} \simeq \texttt{a}(\text{T}(t) - \text{T}_a) + \texttt{b}$$

D'où, après passage à la limite quand $dt \to 0$, on obtient l'équation différentielle :

Déterminer alors l'unique solution de cette équation différentielle vérifiant T(0) = ...

Représenter cette solution sur **graphique 1**.

II Méthode numérique de résolution d'équations différentielles d'ordre 1

Le but de cette partie est d'étudier et de mettre en pratique une méthode de résolution numérique d'équation différentielle d'ordre 1. Il est normal de se poser la question d'une résolution numérique, puisque l'on sait que, de façon générale, il n'existe pas de méthode de résolution des équations différentielles. Nous présenterons ici la méthode la plus élémentaire : **la méthode d'Euler explicite**.

☞ Pour info...
Et oui, parce-qu'il existe une méthode d'Euler implicite...

Considérons une équation différentielle linéaire d'ordre 1 et un problème de Cauchy de la forme :

$$\begin{cases} y' = \mathrm{F}(t, y) \\ y(a) = y_{\text{ini}} \end{cases}$$

d'inconnue $y \in \mathcal{C}^1([a;b], \mathbb{R})$ et où F désigne une fonction de deux variables, suffisamment régulière en les deux variables... et $y_{\text{ini}} \in \mathbb{R}$.
Sous de bonnes hypothèses sur la fonction F, le célèbre théorème de Cauchy-Lipschitz assure que ce problème de Cauchy possède une unique solution.
Pour résoudre numériquement cette équation différentielle, on considère une subdivision $(t_0, t_1, ..., t_n)$ de $[a;b]$ à pas constant noté h. On a ainsi $h = \dfrac{b-a}{n}$.

Petite remarque
On ne se préoccupera ni de l'ensemble de définition de F, ni de la convergence de la méthode proposée... On fait simplement les choses, en constatant que, pour des exemples bien choisis, ça marche !

Objectif : l'idée de la méthode est de déterminer, pour tout $k \in [\![0;n]\!]$, une valeur approchée de $y(t_k)$, notée y_k. Si h est suffisamment petit, on espère alors obtenir une bonne approximation de la solution y sur $[a;b]$.
Mise en œuvre : concrètement, on pose $y_0 = y_{\text{ini}}$ et on considère que, si h est suffisamment petit :

$$y'(t_k) \simeq \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

Or $y'(t_k) = \mathrm{F}(t_k, y(t_k))$ et $t_k + h = t_{k+1}$. On a donc :

$$y(t_{k+1}) \simeq \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

On définit ainsi la suite $(y_k)_{k \in [\![0;n]\!]}$ par :

$$\begin{cases} y_0 = y_{\text{ini}} \\ \forall k \in [\![0; n-1]\!],\ y_{k+1} = y_k + h\mathrm{F}(t_k, y_k) \end{cases}$$

Programmer la méthode d'Euler pour résoudre numériquement les problèmes de Cauchy suivants (on souhaite obtenir une représentation graphique de l'approximation de la solution) :

1. $y' = y$ et $y(0) = 1$ (sur $[0;10]$) puis comparer avec la solution exacte
2. $y' = y^2$ et $y(1) = 1$ (sur $[1;10]$) puis comparer avec la solution exacte
3. $y' = e^{-y} - e^{-3y} + e^{-5y}$ et $y(0) = 1$ (sur $[0;1000]$)
4. autres plus exotiques, au choix...