DS d'informatique –CORRECTION
Mai 2014

Partie A

Dans tout le devoir, on s'interdit l'usage de la fonction ***min*** préprogrammée en Python et permettant d'obtenir directement le minimum d'une liste donnée en argument. En revanche, on se donne la fonction mini suivante, écrite en Python :

```
1  def mini(t):
2      '''Calcule le minimum d'un tableau d'entiers ou de flottants.'''
3      if len(t) == 0:
4          return None
5      p = t[0]
6      for i in range(len(t)):
7          if t[i] <= p:
8              p = t[i]
9      return p
```

1. Expliquer le déroulement pas à pas (évolution de la valeur des variables) lors de l'appel mini([6,2,15,2,15]), puis donner la valeur renvoyée.

Si la liste t est vide alors *len(t)=0* et la fonction n'a aucun minimum à renvoyer. Rappelons que la première instruction *return* rencontrée met fin à la fonction (les autres instructions ne sont pas lues).
Sinon :

- p prend la première valeur de la liste.
- on parcourt toute la liste (*lent(t)* est sa longueur). A chaque itération, si un élément est inférieur ou égal à la valeur de p courante alors p se voit attribuée la valeur de cet élément.

La valeur renvoyée, lorsque la liste n'est pas vide, est donc minimum des différentes valeurs de la liste.
Voici le saleurs prises par les variables, pas à pas :

p	6	6	2	2	2	2
i		0	1	2	3	4

2. Évaluer la complexité temporelle de l'appel *mini(t)* en fonction du nombre n d'éléments de *t*
Avant d'entrer dans la boucle, on effectue un test et une affectation, soit 2 instructions.
Il y a autant d'itérations que le nombre d'éléments dans la liste. Lors de chaque itération, il y a une comparaison et éventuellement une affectation.
Dans le meilleur des cas, il y a donc : 2+n instructions
Dans le pire des cas, il y a donc : 2+2n instructions

3. Proposer une modification de la fonction mini pour que la valeur renvoyée soit le maximum et non le minimum. On pourra utiliser la numérotation des lignes pour préciser le lieu d'éventuelles modifications et ainsi éviter de réécrire toute la fonction.
Dans la ligne 7, il suffit de remplacer « *if* $t[i] \leq p$ » par « *if* $t[i] \geq p$ ».

4. On souhaite récupérer non plus le minimum d'une liste mais toutes les positions dans le tableau où le minimum est atteint. Dans l'exemple vu plus haut, il y a deux positions où ce minimum est atteint : 1 et 3.
Écrire une fonction ***position_mini*** réalisant effectivement cette opération.

On commence par déterminer le minimum puis on parcourt la liste afin d'enregistrer les indices :

```
def position_mini(t):
    "renvoie les indices du minimum d'une liste "
    if len(t) == 0:
         return None
    p = t[0]
    indice_min=[]
    for i in range(len(t)):
        if t[i] <= p:
            p = t[i]
    for i in range(len(t)):
        if t[i]==p:indice_min.append(i)
    return indice_min
```

Partie B

Votre binôme de TIPE veut essayer de modéliser les variations annuelles de température en les couplant aux variations saisonnières d'ensoleillement. Vous n'avez pas parfaitement compris ses arguments à base de « corps noir » et de « loi de Stéfan », mais il semble assez sûr que l'équation régissant l'évolution de la température (notée θ car exprimée en degrés Celsius) s'écrive :

$$\frac{d\theta}{dt} + \alpha\left[\theta(t) + T_0\right]^4 = \mu\left[1 + \varepsilon\cos(\omega t)\right]$$

Vous écrivez donc la fonction suivante qui permet de résoudre numériquement des équations différentielles du type y0 = f(y; t) en utilisant la méthode d'Euler :

Implémentation de la méthode d'Euler en Python

```
def euler(f,y0,dt):
    y,t,tmax = y0,0,950
    y_arr,t_arr=[y],[t]
    while t < tmax:
        y = y + f(y,t)*dt
        t = t + dt
        y_arr.append(y)
        t_arr.append(t)
    return t_arr,y_arr

def f(theta,t):
    return                          # à completer...
```

1. Expliquer le principe de la méthode d'Euler. On signalera notamment à quoi correspondent les différentes parties du code fourni pour la fonction Euler et la nature et la signification des paramètres d'entrée f, y0 et dt.

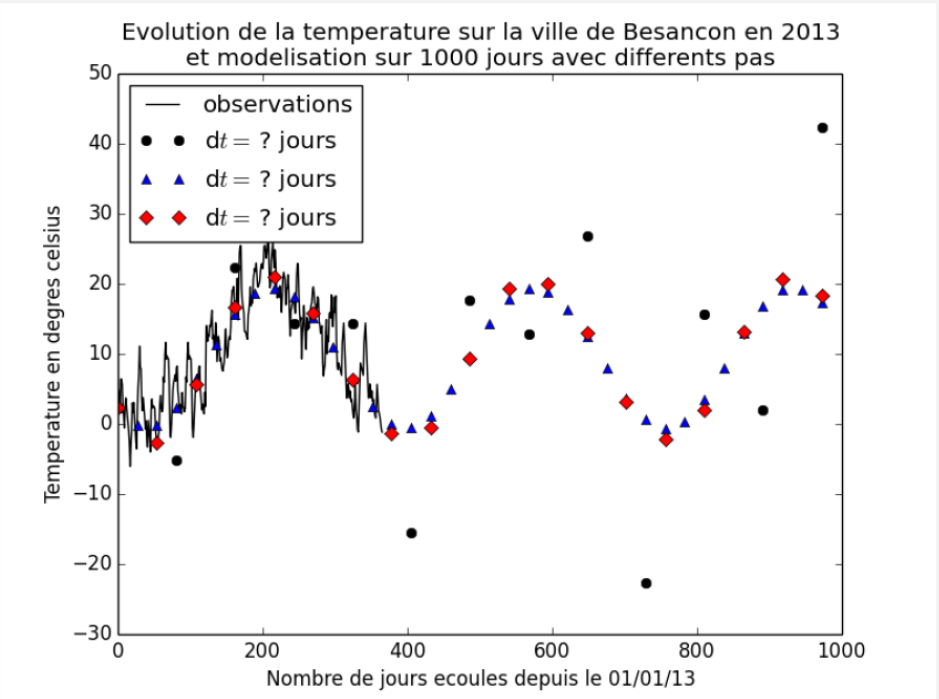
Si l'on souhaite résoudre une équation différentielle du premier degré du type $\begin{cases} y(0) = y_0 \\ y'(t) = f(t, y(t)) \end{cases}$, la méthode d'Euler permet d'obtenir les images des valeurs $0, dt, 2dt, 3dt ...., n \times dt$ (distantes de dt) avec n tel que $n \times dt < t_{max}$ et $(n+1)dt \geq t_{max}$ par une valeur approchée de la véritable solution.
La liste « *y_arr* » contient les images par cette solution approchée des valeurs contenues dans la liste « *t_arr* ».

2. En supposant que votre binôme ait stocké dans les variables globales *alpha, T0, mu, epsilon* et *omega* les valeurs adéquates pour le problème posé, écrire la fonction « def *f(theta,t)* pour que l'appel à la fonction Euler renvoie effectivement une solution numérique de l'équation différentielle régissant l'évolution de θ .

```
from math import cos

def f(theta,t):
    return(-alpha*(theta+T0)**4+mu*(1+epsilon*cos(omega*t)))
```

3. Le graphe suivant, réalisé sous ***Matplotlib***, représente les points expérimentaux (reliés par des traits pleins) auxquels on a superposé trois courbes intégrées avec une valeur différente du « pas d'intégration », respectivement de 27 jours, 54 jours et 81 jours. Associer à chaque pas d'intégration son symbole (rond noir, triangle bleu ou losange rouge) en justifiant. On expliquera en plus le comportement observé des symboles ronds lorsque le temps d'intégration est important.



Les ronds représentent les points pour un pas de 81 jours.
Les losanges représentent les points pour un pas de 54 jours.
Les triangles représentent les points pour un pas de 27 jours.

Au cours du temps, les points ronds se dissocient de plus en plus des autres. En effet, nous savons que lorsque le pas est grand, l'erreur commise (écart entre la solution exacte et la solution approchée) par la méthode d'Euler est de plus en plus grand.

Partie C

```
from math import *

def f(x):
    return(2*x/(1+x**2))

def rectangles(f, a, b, n):
    s=0
    x=a
    h=(b-a)/n
    for i in range(n):
        s=s+h*f(x)
        x=x+h
    return(s)

n=int(input('Saisir le nbre de subdivisions'))
I=rectangles(f,0,1,n)
print('Valeur approchee par méthode rectangles gauche :',I)
print(log(2))
print('pourcentage erreur :',abs(100*(I-log(2))/log(2)))
```