## Partie 1 : quelques outils pour le traitement des données

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

**Correction**

```
mini([6,2,15,1,15,1])
pour i=1 mini=2
pour i=2 mini=2
pour i=3 mini=1
pour i=4 mini=1
pour i=5 mini=1
```

**Question 2** *Évaluer la complexité temporelle de l'appel* ***mini(t)*** *en fonction du nombre n d'éléments de* `t`*.*
*Vous calculerez pour cela le nombre de comparaisons et d'opérations (addition, soustraction, division, multiplication) réalisées.*

**Correction** il y a (n-1) comparaisons et aucune opération soit une complexité en O(n)

**Question 3** *Proposer une modification de la fonction* ***mini*** *pour que la valeur renvoyée soit le* maximum *et non le* minimum.

**Correction**

```
def max(t):
    if len(t)==0:
        return None
    m=t[0]
    for i in range(1,len(t)):
        if t[i]>=m:
            m=t[i]
    return (m)
```

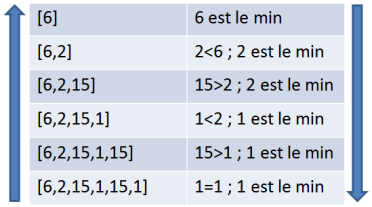
**Question 4** *Définir, du point de vue de la programmation, ce qu'est une fonction récursive. Donner les étapes de l'écriture d'une fonction récursive en donnant comme exemple la recherche du minimum d'une liste.*

**Correction** Il est nécessaire d'avoir une condition d'arrêt et une boucle de récursivité. La condtion d'arrêt pour la fonction **mini_recursive** est le mini de la liste à un seul élément et la boucle de récursivité est la comparaison du mini de la liste tronquée avec l'élément suivant.

**Question 5** *Définir une fonction* ***mini_recursive*** *qui a pour argument une liste de valeurs entières ou flottants et qui renvoie par une méthode récursive le* minimum *de cette liste.*

**Correction**

```
def mini_recursive(t):
    n=len(t)
    if n==1:
        return t[0]
    else:
        p=mini_recursive(t[0:n-1])
        if t[-1]<p:
            p=t[-1]
        return p
```



| | |
|---|---|
| [6] | 6 est le min |
| [6,2] | 2<6 ; 2 est le min |
| [6,2,15] | 15>2 ; 2 est le min |
| [6,2,15,1] | 1<2 ; 1 est le min |
| [6,2,15,1,15] | 15>1 ; 1 est le min |
| [6,2,15,1,15,1] | 1=1 ; 1 est le min |

**Question 6** *Écrire une fonction **position_mini** d'argument une liste et renvoyant **une** position du minimum de cette liste.*

**Correction**

```
def position_mini(t):
    indice=0
    p=t[0]
    for i in range(1,len(t)):
        if t[i]<=p: #ou t[i]<p
            p=t[i]
            indice=i
    return i
```

**Question 7** *Écrire une fonction **min2D** d'argument un tableau **t** d'entiers ou flottants et renvoyant le minimum de **t**. On utilisera la fonction **mini**.*

**Correction**

```
def mini2D(t):
    p=mini(t[0])
    for i in range(1,len(t)):
        p1=mini(t[i])
        if p1<p:
            p=p1
    return p
```

*Évaluer la complexité temporelle de cette fonction. Vous calculerez pour cela le nombre de comparaisons et d'opérations (addition, soustraction, division, multiplication) réalisées.*

**Question 8** *Écrire une fonction **chaine_mini** réalisant effectivement cette opération.*

**Correction**

```
def chaine_mini(L):
    nom=L[0][0]
    valeur=L[0][1]
    for i in range(1,len(t)):
        if L[i][1]<=valeur:
            valeur=L[i][1]
            nom=L[i][0]
    return nom
```

**Question 9** *Écrire enfin une fonction **majores_par** réalisant cette opération.*

**Correction**

```
def majores_par(t,seuil):
    k=0
    for i in range(len(t)):
        if t[i]<seuil:
            k+=1
    return k
```

**Question 10** *Modifier la fonction précédente pour obtenir une fonction **elements_majores_par** retournant la liste des éléments majorés par le seuil.*

**Correction**

```
def elements_majores_par(t,seuil):
    L1=[]
    for i in range(len(t)):
        if t[i]<seuil:
            L1.append(t[i])
    return L1
```

## Partie 2 : Exploitation des mesures de température

**Question 11** *Définir la fonction* $\boldsymbol{g(\theta)=R(\theta)-R_{mesure}}$.

**Correction**

```
def g(x):
    return 100*(1+a*x+b*x**2+c*x**3*(x-100))-R
```

**Question 12** *Écrire les instructions python permettant de tracer* $\boldsymbol{g(\theta)}$ *pour* $\theta \in [0,50]$ *et* $R = 104\Omega$ *par pas de* 0.5.

**Correction**

```
les_theta=[i*0.5 for i in range(101)]
les_g=[g(i) for i in les_theta]
plt.plot(les_theta,les_g)
plt.xlabel('les theta')
plt.ylabel('g(theta)')
plt.grid()
plt.show()
```

**Question 13** *Écrire une fonction* ***dichotomie*** *d'arguments les bornes d'étude* `a` *et* `b`*, la fonction* `g` *ainsi que la précision* `epsilon` *et qui renvoie la valeur de* $\theta$ *à epsilon près solution de* $\boldsymbol{g(\theta)=0}$. *Proposer une valeur pour* `a` *et une valeur pour* `b`. *Justifier votre choix.*

**Correction**

```
def dichotomie(a, b, f, epsilon):
    g = a
    d = b
    while (d - g) > 2 * epsilon:
        m = (g + d)/2
        if f(g) * f(m) <= 0:
            d = m
        else:
            g = m
    return((d + g) / 2)
```

Remarque : `b` et `c` étant très faibles, on utilise généralement la relation approchée suivante :

- $R(\theta) = R(0C) \times (1 + a\theta)$ avec $a = 3,90802.10^{-3}C^{-1}$

**Question 14** *Écrire une fonction* ***tempInfrahoraire*** *ayant pour arguments une liste des temps en seconde sur un intervalle de 6 minutes et la liste des températures en degré celsius correspondantes et qui renvoie deux valeurs : la moyenne des températures sur cet intervalle de temps et l'heure en seconde correspondant au relevé de température. Les pas de temps ne sont pas constants dans la liste des temps. Vous appliquerez pour l'intégration la méthode des rectangles à "droite".*

**Correction**

```
def tempinfrahoraire(Ltemps,Ltheta):
    s=0
    for i in range(1,len(Ltheta)):
        s=s+Ltheta[i]*(Ltemps[i]-Ltemps[i-1]) #intégration àdroite, c'est la valeur de température mesurée à
l'instant t{i+1} qui correspond àla température mesurée sur l'intervalle [t{i},t{i+1}]
```

```
    return s/(Ltemps[-1]-Ltemps[0]),Ltemps[-1]
```

**Question 15** *Écrire les instructions python pour construire les deux listes* `les_t_6min` *et* `les_theta_6min` *à partir des deux listes des relevés pour une journée. Vous utiliserez la fonction* ***tempInfrahoraire****.*

**Correction**

```
les listes de temps en secondes et de theta ont la même longueur
les_t_6min=[]
les_theta_6min=[]
for i in range(len(theta)):
    s=0
    Ltemps=[]
    Ltheta=[]
    while s<=360:
        Ltemps.append(temps[i])
        Ltheta.append(theta[i])
        s=s+temps[i]-temps[i-1]
    moy_theta,heure=tempinfrahoraire(Ltemps,Ltheta)
    les_t_6min.append(heure)
    les_theta_6min.append(moy_theta)
```

## Partie 3 : Recherche dans les archives de Météo-France

La base de données de Météo-France concernant l'année écoulée est constituée de deux tables :

**Question 16** *Qu'appelle-t-on* ***clé primaire*** *pour une table ? Peut-on facilement en définir une pour la table* ***villes*** *? Pour la table* ***mesures*** *?*

**Correction** une clé primaire est la donnée qui permet d'identifier de manière unique un enregistrement dans une table pour la table ville, la clé primaire est le numéro d'insee pour la table mesures, il n'y a pas de clé primaire simple, il faut la construire à partir de deux colonnes ville,jour

**Question 17** *Écrire une requête en langage SQL qui récupère depuis la table* ***mesures*** *les relevés dont la température moyenne définie par* $T_{moy} = \frac{T_{min}+T_{max}}{2}$ *est strictement inférieure à* 0 *degré celsius (avec toutes les colonnes disponibles).*

**Correction**

```
SELECT * FROM mesures WHERE (Tmax+Tmin)/2<0;
```

**Question 18** *Écrire une requête en langage SQL qui récupère depuis la table* ***villes*** *le numéro INSEE, le nom et le département de toutes les villes du quart nord-est de la France, c'est-à-dire celles dont la latitude est supérieure à* 47 *degrés et dont la longitude est supérieure à* 2 *degrés.*

**Correction**

```
SELECT INSEE,nom,dpt FROM villes WHERE lat>47 AND lon>2;
```

**Question 19** *Écrire une requête SQL qui récupère le nom des villes dont la température maximale est supérieure strictement à 30°C.*

**Correction**

```
SELECT DISTINCT nom FROM villes INNER JOIN mesures ON INSEE=ville WHERE Tmax>30;
```

## Partie 4 : Manipulation des données

On admet que l'appel de la fonction a permis de récupérer trois listes : la liste **jours** contenant le numéros du jour et les deux listes **Tmin** et **Tmax** contenant les températures minimales et maximales concernant le jour correspondant.

**Question 20** *Proposer un ordre de grandeur du nombre d'octets utiles du fichier* ***saintEtienne_2016.txt*** *dont chaque caractère est codé en ASCII.*

**Correction** nb de jours codage de 1 à 365 soit (9+2*90+3*265)caractères nb de ' ;' 2*365 4 caractères maxi pour la température mini et maxi : 4*2*365 soit 4634 octets (ASCII 1 octet)

**Question 21** *Écrire une fonction* ***moyenne*** *qui prend en entrée deux listes* ***a*** *et* ***b*** *de même taille (condition qui ne doit* ***pas*** *être vérifiée) et renvoie une liste de même taille contenant dans la case d'indice* ***i*** *la valeur moyenne des valeurs des flottants stockés dans les deux listes* ***a*** *et* ***b*** *à l'indice* ***i****.*

**Correction**

```
def moyenne(a,b):
    Lm=[]
    for i in range(len(a)):
        Lm.append((a[i]+b[i])/2)
    return Lm
```

**Question 22** *En appliquant la fonction précédente, écrire l'instruction Python qui stocke dans la variable* ***Tmoy*** *la liste des températures moyennes journalières à partir des données stockées dans les listes* ***Tmin*** *et* ***Tmax****.*

**Correction**

```
Tmoy=moyenne(Tmin,Tmax)
```

**Question 23** *On considère qu'il est nécessaire de couper les arrivées d'eau extérieures pour risque de gel quand la température moyenne sur la journée est strictement inférieure à* 0 *degré celsius. En utilisant une des fonctions programmées dans la première partie, stocker dans la variable* ***nb_jours_gel*** *le nombre de jours où il a fallu couper l'eau des conduites extérieures pour la ville de Saint Étienne.*

**Correction**

```
nb_jours_gel=majores_par(Tmoy,0)
```

**Question 24** *On s'intéresse ici aux écarts entre les températures minimale et maximale au cours d'une même journée, écrire les lignes de commande qui permettent de créer une liste* ***ecart*** *contenant les écarts* $Tmax - Tmin$ *pour chaque jour et de créer la variable mini_ecart contenant le minimum des écarts pour l'année* 2016*.*

**Correction**

```
ecart=[Tmax[i]-Tmin[i] for i in range(len(Tmin))]
mini_ecart=mini(ecart)
```

**Question 25** *On souhaite étudier les variations d'un jour sur l'autre de la température maximale.*
*Donner les lignes de commandes permettant d'obtenir,* ***sans créer de nouvelles listes****, la plus grande variation de la température maximale entre deux jours consécutifs.*

**Correction**

```
variation=0
for i in range(0,len(Tmax)-1):
    e=abs(Tmax[i+1]-Tmax[i])
    if e>variation:
        variation=e
print (variation)
```

## Partie 5 : une modélisation physique

**Question 26** *Écrire la fonction* ***f*** *d'arguments deux flottants* $t$ *et* $\theta$ *et retournant* $f(\theta, t)$*.*

**Correction**

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

**Question 27** *Justifier que pour tout $k$ tel que $1 \leq k \leq N$, on a $\theta_k = \theta_{k-1} + f(\theta_{k-1}, t_{k-1}) \times dt$.*

**Correction**

$\frac{d\theta(t)}{dt} = f(\theta(t), t)$
$\frac{\theta(t+h)-\theta(t)}{h} \approx f(\theta(t), t)$
ainsi $\theta_k = \theta_{k-1} + f(\theta_{k-1}, t_{k-1}) \times dt$

**Question 28** *Écrire la fonction **euler** d'argument trois flottants theta0, dt et tmax qui retourne deux listes : **temps** et **theta** contenant respectivement les valeurs $t_k$ et $\theta_k$ pour $0 \leq k \leq N$.*

**Correction**

```
def euler(theta0,dt,tmax):
    temps=[0]
    theta=[theta0]
    t=dt
    while t<=tmax:
        temps.append(t)
        theta.append(theta[-1]+dt*f2(theta[-1],t))
        t=t+dt
    return temps,theta
```