

Relevés de températures [it04]

Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel  algoprog  Version 17 mai 2018

Table des matières

1 Relevés de températures / pgmeteo	2
1.1 Énoncé	2
1.2 Problème 1	3
1.3 Problème 2	5
1.4 Problème 3	6
2 Que retenir de cet exercice ?	7
3 Références générales	8

Python - Relevés de températures (Solution)



Mots-Clés Schéma itératif ■

Requis Structures de base, Structures conditionnelles, Structures répétitives ■

Difficulté ● ● ○



Objectif

Cet exercice traite de relevés de températures. Il reprend des algorithmes de base étudiés dans les exercices @[Somme d'une suite] (dans le support de cours) et @[Maximum et son rang] (dans ce module). Il aborde aussi le problème du contrôle des réponses de l'utilisateur.

1 Relevés de températures / pgmeteo

1.1 Énoncé

Une station météo effectue chaque jour un relevé de températures pendant une période donnée de n jours consécutifs. Il s'agit d'écrire un script qui :

- Saisit le nombre n de jours de la période à traiter.
- Pour chaque jour de la période, affiche le numéro du jour et saisit la température relevée ce jour.
- Calcule et affiche la température minimale et le numéro du jour où elle a été relevée.
- Calcule et affiche la température maximale et le numéro du jour où elle a été relevée.
- Calcule et affiche la température moyenne sur la période de relevés.

On demande trois versions qui correspondent aux trois questions suivantes :

1. Dans la première version, on ne fait pas d'hypothèse sur les valeurs des températures. On suppose que l'utilisateur donne un nombre de jours strictement positif.
2. Dans la deuxième version, sans le contrôler, on suppose que l'utilisateur donne un nombre de jours positifs et des températures qui appartiennent à l'intervalle $[-80, 80]$.
3. Dans la troisième version, l'algorithme oblige l'utilisateur à saisir un nombre de jours compris entre 1 et 366, ainsi qu'une température comprise en -80 et 80. L'obligation signifie que tant que l'utilisateur ne donne pas une valeur satisfaisante, l'algorithme lui redemande une valeur.

Remarque

Il peut y avoir des ex-aequo, c.-à-d. plusieurs jours correspondant à la température minimale (resp. maximale). Dans ce cas, l'algorithme n'affichera que la première occurrence du minimum (resp. maximum) rencontrée.

Exemple

Voici un exemple de dialogue correspondant à l'algorithme demandé :

```
Nombre de jours dans la periode? 7
Temperature du jour 1? 5
Temperature du jour 2? 8
Temperature du jour 3? -3
Temperature du jour 4? 4
Temperature du jour 5? 10
Temperature du jour 6? -3
Temperature du jour 7? -1
==> Temperature Mini = -3 jour 3
==> Temperature Maxi = 10 jour 5
==> Temperature Moyn = 2.9
```

1.2 Problème 1

Le problème de la recherche du maximum est analogue à la première question du problème @[Maximum et rang]. Celui du minimum l'est aussi, à condition de changer le sens des inégalités. Le nombre de valeurs à traiter correspond ici au nombre de jours de la période. L'énoncé ne le précisant pas, on décide de déclarer les températures avec le type réel.



Choix des variables

Il faudra donc déclarer des variables réelles pour :

- La température du jour courant `tj`.
- La température minimale `tmin`.
- La température maximale `tmax`.

Et des variables entières pour :

- Le nombre de jours à traiter `njours`.
- Le numéro du jour courant `nj`.
- Le numéro du premier jour où la température est minimale `jmin`
- Le numéro du premier jour où la température est maximale `jmax`.

Enfin pour calculer la température moyenne, il faut calculer la somme et déclarer deux autres variables réelles pour :

- La somme des températures `somme`.
- La moyenne des températures `moyenne`.



Écrivez un script qui déclarent les variables.



Déroulement de l'algorithme

Il se compose des trois parties classiques :

Initialisations

Boucle des traitements de chaque jour de la période

Affichage des résultats



Initialisations

Cette partie commence par la demande à l'utilisateur du nombre de jours de la période. On suppose qu'il donne un nombre strictement positif.

Il faut ensuite lui demander la température du premier jour et donner cette valeur aux variables `tmin` et `tmax` qui sont le minimum et le maximum de référence.

Il faut aussi initialiser à 1 les numéros de jour `jmin` et `jmax`.

Enfin, comme il faudra calculer la somme en cumulant les températures de chaque jour, il faut l'initialiser avec la température du premier jour.



Initialisez les variables.

**Traitement répétitif**

La boucle doit s'exécuter du jour n°2 au jour dont le numéro est égal au nombre de jours de la période. On peut donc utiliser une boucle **Pour** ayant comme variable de boucle le numéro du jour courant **nj**. Le corps de la boucle doit contenir les actions suivantes :

- Affichage du numéro du jour concerné par la saisie.
- Saisie de la température du jour.
- Comparaison avec la température minimale et mise à jour éventuelle de la température minimale de référence et du jour correspondant.
- Comparaison avec la température maximale et mise à jour éventuelle de la température maximale de référence et du jour correspondant.
- Mise à jour de la somme par cumul de la température.



Écrivez la boucle du traitement.

**Résultats**

Avant d'afficher tous les résultats demandés, il faut calculer la moyenne en divisant la somme par le nombre de jours de la période. on est sûr qu'il n'y aura pas de division par 0 puisqu'on suppose que le nombre de jours est strictement positif.



Calculez et affichez les résultats.



Testez.



Validez votre script avec la solution.

Solution Python

@[pgmeteo.py]

```
def test1():
    """ Probleme 1 """
    njours = int(input("Nombre de jours (>0)? "))
    tj = float(input("Temp. du jour 1? "))
    tmin, tmax = tj, tj
    jmin, jmax = 1, 1
    somme = tj
    for nj in range(2, njours+1):
        print("Temp. du jour ", nj, "? ", sep="", end="")
        tj = float(input())
        if tj < tmin:
            tmin, jmin = tj, nj
        if tj > tmax:
            tmax, jmax = tj, nj
        somme += tj
    tmoy = somme / njours
    print("==> Temp. min = ", tmin, " jour ", jmin, sep="")
    print("==> Temp. max = ", tmax, " jour ", jmax, sep="")
    print("==> Temp. moy = ", tmoy, sep="")
```



Commentaires

En observant l'algorithme obtenu, on peut se demander s'il est nécessaire de tester systématiquement le minimum **et** le maximum. En effet, l'expression booléenne `tmin<=tmax` est toujours vraie. Donc si `tj<tmin` est vraie, par transitivité, `tj<tmax` est vraie si bien que la deuxième conditionnelle ne peut pas être activée. Comme toujours quand on peut le faire, il est préférable de remplacer les deux conditionnelles consécutives, par une alternative **Si-Sinon-Si**. D'où :

```
...
Si (tj < tmin) Alors
| tmin <- tj
| jmin <- nj
Sinon Si (tj > tmax) Alors
| tmax <- tj
| jmax <- nj
Finsi
...
```

1.3 Problème 2

Dans ce problème, il existe une contrainte sur les températures qui doivent appartenir à l'intervalle $[-80, 80]$. On suppose toujours que l'utilisateur répond en cohérence avec ce qui lui est demandé. On va utiliser ces contraintes pour modifier la méthode d'initialisation. En effet, si on initialise la température minimale avec une valeur supérieure à la plus grand valeur possible (par exemple 81), elle sera forcément mise à jour lors de la première comparaison dans la boucle. Il en est de même si on initialise la température maximale avec une valeur inférieure à la plus petite valeur possible (par exemple -81).

Il n'est plus nécessaire de saisir la température du premier jour à l'extérieur de la boucle à condition que celle-ci soit exécutée dès le premier jour. Précisons aussi que les variables `jmin` et `jmax` étant nécessairement affectées lors de la première comparaison dans la boucle, il n'est pas nécessaire de les initialiser. Enfin, la somme doit être initialisée à 0 puisque la première température est traitée dans la boucle.

Avant de modifier l'algorithme, expliquons pourquoi la modification doit être faite sur la version 1.1 (utilisant des **Si** en séquence) et pourquoi elle est **fausse sur la version 1.2** (utilisant le **Si-Sinon-Si**). Il faut nécessairement que les deux températures `tmin` et `tmax` soient affectées au moins une fois dans la boucle. Il en est de même pour les numéros de jours `jmin` et `jmax`. Or la conditionnelle imbriquée pourrait ne pas le permettre. Supposons en effet que l'expression `tj<tmin` soit vraie à chaque passage dans la boucle. Cette situation peut se présenter si les températures apparaissent dans un ordre strictement décroissant qui fait que la température courante est toujours inférieure à la température de référence. Dans ce cas, la clause **Sinon** ne serait jamais activé, la variable `tmax` garderait sa valeur de départ (-81), et la variable `jmax` n'aurait pas de valeur. C'est pourquoi cette méthode d'initialisation par des valeurs extérieures à l'intervalle de validité ne peut être utilisée qu'avec la première version de l'algorithme, celle où les deux variables `tmin` et `tmax` sont toutes deux testées par les conditionnelles consécutives.



Définissez les constantes `TPMIN=-80` et `TPMAX=80` représentant la contrainte $[-80, 80]$.



Modifiez ensuite votre script selon cette analyse.



Testez.



Validez votre script avec la solution.

Solution Python @[pgmeteo.py]

```
TPMIN = -80.0
""" Température MINimale """
TPMAX = 80.0
""" Température MAXimale """

def test2():
    """ Probleme 2 """
    njours = int(input("Nombre de jours (>0)? "))
    tmin, tmax = TPMAX, TPMIN
    jmin, jmax = 0, 0
    somme = 0.0
    for nj in range(1, njours+1):
        print("Temp. ([", TPMIN, ", ", TPMAX, "] du jour ", nj, "? ", sep="", end="")
        tj = float(input())
        if tj < tmin:
            tmin, jmin = tj, nj
        if tj > tmax:
            tmax, jmax = tj, nj
        somme += tj
    tmoy = somme / njours
    print("==> Temp. min = ", tmin, " jour ", jmin, sep="")
    print("==> Temp. max = ", tmax, " jour ", jmax, sep="")
    print("==> Temp. moy = ", tmoy, sep="")
```

1.4 Problème 3

Nous ne voulons plus faire confiance à l'utilisateur, il faut le contraindre à fournir des valeurs ayant un sens pour le problème. Dans le cas présent, l'utilisateur doit donner un nombre de jours compris entre 1 et 366, et des températures comprises entre `TPMIN=-80` et `TPMAX=80`. On va donc lui poser la question jusqu'à ce qu'il donne une réponse correcte. Comme il y a forcément une première saisie on peut envisager une boucle **Répéter**. Par exemple dans le cas des températures, on peut envisager une boucle comme suit :

```
Répéter
    Afficher("Temp ([", TMIN, ", ", TMAX, "] jour ", nj, "? ")
    Saisir(tj)
Jusqu'à (tj >= TMIN Et tj <= TMAX)
```

Supposons que nous soyons au jour 5 et que l'utilisateur donne trois fois de suite une température erronée. Voici ce qu'il y aura comme dialogue :

```
Temp ([-80,80]) jour 5? 150
Temp ([-80,80]) jour 5? -110
Temp ([-80,80]) jour 5? 90
Temp ([-80,80]) jour 5? 15
Temp ([-80,80]) jour 6? ...
```

Si on veut signifier à l'utilisateur l'existence d'une erreur, il faut alors distinguer le message initial des messages d'erreur et l'utilisation d'une répétitive **Répéter** n'est plus possible. Il faut utiliser une répétitive **TantQue** dont le sens sera : tant qu'il y a une erreur donner une nouvelle valeur.

La condition d'entrée dans la boucle **TantQue** doit être la négation de la condition de sortie de la boucle **Répéter**. On aura donc :

```
Afficher("Temp ([" ,TPMIN," ,",TPMAX,"]) jour " , nj,"? ")
Saisir(tj)
TantQue (non (tj >= TPMIN Et tj <= TPMAX)) Faire
    Afficher("Erreur Temp ([" ,TPMIN," ,",TPMAX,"])? ")
    Saisir(tj)
FinTantQue
```

Le dialogue précédent est modifié, rendant repérables les messages correspondant à une erreur :

```
Temp ([-80,80]) jour 5? 150
Erreur Temp ([-80,80])? -110
Erreur Temp ([-80,80])? 90
Erreur Temp ([-80,80])? 15
Temp ([-80,80]) jour 6? ...
```



Complétez votre script selon cette analyse.



Testez.



Validez votre script avec la solution.

Solution Python

```
...
njours = int(input("Nombre de jours ([1..366])? "))
while (not(1 <= njours <= 366)):
    njours = int(input("Erreur, Nombre de jours ([1..366])? "))
...
```

2 Que retenir de cet exercice?



Nous avons vu que le fait de changer la méthode d'initialisation avait des conséquences sur le choix de l'organisation du corps de la boucle. Lorsque l'on modifie une partie d'un

algorithme, il faut toujours se demander quelles sont les conséquences possibles sur le reste.



Le contrôle des réponses apporte une lourdeur certaine dans le déroulement d'un dialogue. Il est toutefois indispensable pour assurer le bon fonctionnement des algorithmes.



Si l'utilisateur ne se décide pas à donner une réponse correcte, le programme ne s'arrêtera jamais. Pour éviter (si nécessaire) cet inconvénient, on pourra introduire une limitation dans le nombre de réponses erronées autorisées.

3 Références générales

Comprend [Tartier-AL1 :c5 :ex18] ■