Aides pour contrôler les sauvegardes

Module 11 • Comme au CCF 1 • Document élève séparé

Consultez seulement la question qui vous bloque. Essayez avec l'aide 1, puis l'aide 2 si nécessaire. L'aide 3 donne un pseudo-code complet pour les tâches de programmation. Notez votre niveau d'aide sur le sujet ; l'objectif est de pouvoir expliquer votre solution.

Questions 1 et 2 lire les données et la fonction

Aide 1 : écrivez les indices 0, 1, 2… sous la liste. À chaque passage, lisez etats[i].

Aide 2 : le compteur démarre à 0. Il augmente seulement si l'élément lu est exactement "ERREUR". "ABSENT" est un autre texte.

Aide 3 : pour les trois premiers éléments, i vaut 0, puis 1, puis 2 ; le compteur vaut successivement 0, 1, 1. Poursuivez avec les deux suivants. Le return renvoie le compteur une fois le parcours terminé.

Question 3 compter les absences

Aide 1 : reprenez le principe de compter_erreurs, en changeant l'état recherché.

Aide 2 : initialisez à 0 ; parcourez les indices de 0 à Longueur(etats) - 1 ; ajoutez 1 quand etats[i] vaut "ABSENT" ; renvoyez après la boucle.

Aide 3 avec le pseudo code complet

```
Fonction compter_absentes(etats : liste) : entier
    nombre ← 0
    Pour i de 0 à Longueur(etats) - 1 Faire
        Si etats[i] = "ABSENT" Alors
            nombre ← nombre + 1
        Fin Si
    Fin Pour
    Retourner nombre
Fin Fonction
```

Pour appeler en Python : utilisez le nom de la fonction suivi de la liste entre parenthèses. Entourez cet appel de print(...) si vous souhaitez afficher le résultat.

Question 4 corriger le test

Aide 1 : "ABSENT" est différent de "OK". Est-ce pour autant une sauvegarde "ERREUR" ?

Aide 2 : le test doit comparer etats[i] à "ERREUR" avec une égalité. Pour révéler le défaut, prévoyez une liste contenant au moins un "ABSENT".

Aide 3 avec le pseudo code complet

```
Fonction compter_erreurs(etats : liste) : entier
    nombre ← 0
    Pour i de 0 à Longueur(etats) - 1 Faire
        Si etats[i] = "ERREUR" Alors
            nombre ← nombre + 1
        Fin Si
    Fin Pour
    Retourner nombre
Fin Fonction
```

Aides pour décider et aller plus loin

Question 5 décider si une intervention est nécessaire

Aide 1 : une erreur ou une absence suffit. Vous pouvez compter ces deux catégories avec les fonctions déjà écrites.

Aide 2 : additionnez les deux résultats. Si ce total est strictement supérieur à 0, renvoyez Vrai ; sinon Faux. Les fonctions de comptage doivent être définies avant l'appel final en Python.

Aide 3 avec le pseudo code complet

```
Fonction intervention_necessaire(etats : liste) : booléen
    anomalies ← compter_erreurs(etats) + compter_absentes(etats)
    Si anomalies > 0 Alors
        Retourner Vrai
    Sinon
        Retourner Faux
    Fin Si
Fin Fonction
```

Question 6 choisir les tests

Aide 1 : cherchez un cas avec anomalie et un cas sans anomalie.

Aide 2 : un seul élément suffit pour un test utile. N'oubliez pas le cas vide.

Aide 3 : ["ABSENT"] doit donner True ; [] doit donner False. Expliquez chaque résultat avant de lancer le programme.

Défi renvoyer des indices

Aide 1 : le résultat est une liste ; le parcours doit fournir les positions.

Aide 2 : commencez par [] ; utilisez range(len(etats)) ; testez etats[i] != "OK", puis ajoutez i avec append.

Aide 3 avec le pseudo code complet

```
Fonction indices_a_verifier(etats : liste) : liste
    indices ← []
    Pour i de 0 à Longueur(etats) - 1 Faire
        Si etats[i] != "OK" Alors
            Ajouter i à indices
        Fin Si
    Fin Pour
    Retourner indices
Fin Fonction
```

Pour une liste vide, le parcours ne fait aucun passage et renvoie []. Pour ["ABSENT"], l'unique indice est 0. Relisez la consigne pour distinguer l'indice d'un poste et le texte de son état.