Corrigé du contrôle des sauvegardes

Module 11 • Comme au CCF 1 • Guide enseignant

Premier entraînement guidé de 60 minutes, conçu pour consolider avant davantage d'autonomie. Le contexte et les questions sont originaux. Ce document ne constitue pas un sujet officiel ni une reproduction du format réglementaire d'une épreuve.

Organisation et observation

Parcours : question 1, 7 min ; question 2, 10 min ; question 3, 15 min ; question 4, 10 min ; question 5, 12 min ; question 6, 6 min. Total : 60 min, hors défi facultatif de 10 min. Le sujet compte quatre pages, dont une de défi ; les aides sont séparées sur deux pages. Le mémo des modules précédents est autorisé.

Distribuer d'abord le sujet, puis rendre les aides accessibles. L'élève peut avancer à la question 5 même si ses compteurs ne sont pas terminés : ils y sont considérés corrects. Sans poste, prévoir les sorties et suivre les variables ; reporter l'exécution à une séance équipée. Ne pas confondre manque de temps et incompréhension.

1 Comprendre les données

1a. len(etats) vaut 5 ; etats[2] vaut "ABSENT".

1b. Les indices des deux erreurs sont 1 et 4.

1c. Deux erreurs et une absence. Accepter une correction spontanée après rappel des indices ; observer si l'élève distingue position, valeur et quantité.

2 Analyser la fonction

Indice i	**0**	**1**	**2**	**3**	**4**
État etats[i]	OK	ERREUR	ABSENT	OK	ERREUR
Compteur après le passage	0	1	1	1	2

2b. La fonction renvoie 2. "ABSENT" ne satisfait pas le test etats[i] == "ERREUR".

2c. Initialiser avant la boucle conserve le cumul entre les passages ; initialiser dans le bloc remettrait le compteur à zéro à chaque passage. Le retour après le parcours permet de traiter toute la liste, y compris lorsque celle-ci est vide.

Repérer ce que les réponses montrent

Une trace juste mais une explication hésitante peut indiquer une compréhension encore fragile. Faire expliquer un seul passage avec "ABSENT". Le compteur n'augmente pas, mais l'élément a bien été parcouru. Éviter de demander des noms techniques lorsque l'explication avec les mots de l'élève est correcte.

Compter et réparer

3 Compléter puis traduire

Trous dans l'ordre : 0 ; Longueur(etats) - 1 ; etats[i] = "ABSENT" ; nombre + 1 ; nombre.

```
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
```

```
def compter_absentes(etats):
    nombre = 0
    for i in range(len(etats)):
        if etats[i] == "ABSENT":
            nombre = nombre + 1
    return nombre

etats = ["OK", "ERREUR", "ABSENT", "OK", "ERREUR"]
print(compter_absentes(etats))
```

3b. L'appel affiche 1.

3c. Résultats attendus : 1, 0, 2. La liste vide conserve le compteur à 0. Deux absences successives vérifient que la fonction ne s'arrête pas au premier élément et n'efface pas le cumul.

4 Déboguer la règle

4a. Pour ["OK", "ABSENT"], attendu 0, réel 1.

4b. La version erronée compte tous les états différents de "OK" : les erreurs et les absences. Corriger la condition en if etats[i] == "ERREUR":. Le reste ne change pas.

4c. Exemple discriminant : ["ABSENT"] → attendu 0, réel avant correction 1. Autre exemple : ["ERREUR", "ABSENT"] → attendu 1, réel 2. Une liste sans "ABSENT" ne révèle pas ce défaut ; ["ERREUR"] → 1 dans les deux versions. Demander un test contenant une absence, pas simplement un autre test qui réussit.

Décider et tester

5 Créer la décision

```
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
```

```
def intervention_necessaire(etats):
    anomalies = compter_erreurs(etats) + compter_absentes(etats)
    if anomalies > 0:
        return True
    else:
        return False
```

Les deux compteurs corrects doivent être définis avant l'appel final. Les données sont limitées aux trois états autorisés ; leur somme donne donc le nombre d'anomalies. Aucun pourcentage ni calcul élaboré n'est nécessaire.

```
etats = ["OK", "ERREUR", "ABSENT", "OK", "ERREUR"]
print(intervention_necessaire(etats))
```

5b. Le programme affiche True : deux erreurs et une absence donnent trois anomalies. Pour [], les deux compteurs renvoient 0 et la décision vaut False, conformément à la convention explicite du sujet.

Variantes acceptables

Accepter return anomalies > 0, ou un test utilisant or entre les deux comptes positifs. Un parcours direct avec retour True dès qu'une anomalie est trouvée est correct sur le plan algorithmique, mais ne remplit pas la consigne de réutilisation des deux fonctions : valoriser la logique puis demander une adaptation. Un print à la place de return ne satisfait pas le contrat de la fonction.

6 Tests complémentaires et bilan

Liste	Attendu	Intérêt
["ABSENT"]	True	Une absence seule suffit.
["ERREUR"]	True	Une erreur seule suffit.
["OK", "OK"]	False	Aucune anomalie.
[]	False	Cas vide et zéro passage.

L'élève choisit deux tests, un de chaque résultat. Aucun modèle de rédaction unique n'est exigé. Bilan attendu : la liste vide donne False parce que le sujet fixe cette règle et qu'aucune anomalie n'est présente. Le niveau d'aide sert à ajuster la suite, sans pénalisation automatique.

Défi et exploitation des retours

Défi sur les indices

Trous : [] ; etats[i] != "OK" ; i ; indices. En pseudo-code, la borne finale est incluse ; le parcours de 0 à -1 ne fait ici aucun passage. En Python, range(len(etats)) fournit exactement les indices valides.

```
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
```

```
def indices_a_verifier(etats):
    indices = []
    for i in range(len(etats)):
        if etats[i] != "OK":
            indices.append(i)
    return indices
```

Pour la liste principale : [1, 2, 4]. Tests : [] → [] ; ["OK"] → [] ; ["ABSENT"] → [0]. On ajoute i pour repérer le poste concerné ; ajouter etats[i] renverrait des textes et ferait perdre la position. Accepter une autre solution correcte ; faire ensuite expliciter le lien avec range demandé dans le défi.

Grille de retour sans note obligatoire

Point observé	Seul	Avec aide	À reprendre
Lire une valeur et son indice			
Suivre un compteur			
Traduire et appeler une fonction			
Expliquer une erreur avec un test			
Réutiliser des fonctions pour décider			

Noter les questions qui ont demandé une reformulation, le niveau d'aide et le temps réel. Si l'obstacle est la lecture de l'énoncé, préciser celui-ci ; si le compteur est fragile, reprendre une trace courte ; si le retour pose problème, comparer renvoyer et afficher. Prévoir ensuite un sujet de difficulté voisine avec moins d'aides selon les résultats observés.