Chercher et corriger une erreur

Module 10 • 60 minutes • BTS SIO1

Nom : _____ Date : _____

Votre mission : expliquer pourquoi un programme ne produit pas le résultat voulu, puis le réparer. Modifiez une chose à la fois et vérifiez avec les mêmes données. Sans poste, simulez les instructions sur papier. Aides en page 4 ; défi facultatif en page 5.

Adopter une méthode 5 min

1. Dire le résultat attendu.
2. Choisir un petit exemple et prévoir sa sortie.
3. Exécuter, lire le message éventuel et comparer.
4. Suivre les variables pour localiser l'erreur.
5. Corriger puis rejouer le test et un autre cas.

Trois situations : le programme ne démarre pas ; il démarre puis s'arrête sur une erreur ; il termine avec un résultat faux. Une ligne signalée aide à chercher, mais la cause peut se trouver plus haut.

Réparer un programme qui ne démarre pas 8 min

On veut afficher 0, 1 et 2, chacun sur sa ligne. Ce programme contient deux erreurs de syntaxe. Le second défaut peut n'apparaître qu'après correction du premier.

```
for i in range(3)
print(i)
```

Résultat attendu : _____. Corrigez les deux lignes, puis vérifiez.

.....

.....

.....

Comprendre une erreur pendant l'exécution 10 min

On veut afficher chaque élément de la liste, une seule fois.

```
codes = ["PC", "IMP"]
for i in range(len(codes) + 1):
    print(codes[i])
```

i	Élément demandé	Lecture possible
0		
1		
2		

À quel indice survient l'erreur ? _____. Que voit-on avant l'arrêt ? _____

Corrigez la borne. Quel test supplémentaire vérifie zéro passage ? _____

Observer une erreur de logique

Compléter une trace et corriger 12 min

Cette fonction doit compter tous les stocks nuls. Elle se termine, mais ne renvoie pas toujours le bon résultat.

```
def compter_vides(stocks):
    nombre = 0
    for stock in stocks:
        if stock == 0:
            nombre = nombre + 1
        return nombre
```

Pour [0, 2, 0], résultat attendu : _____. Suivez le programme exactement tel qu'il est écrit, en vous arrêtant dès que return est exécuté.

Étape réellement exécutée	stock	nombre	Action suivante
Initialisation	Non défini	0	
Premier passage			

Résultat réel : _____. Le deuxième élément est-il lu ? _____. Pourquoi ?

.....

.....

Déplacez return pour corriger le code. Écrivez le pseudo-code corrigé, puis Python au dos. Préparez et vérifiez ces tests.

stocks	Résultat attendu	Résultat vérifié
[0, 2, 0]		
[]		
[2, 3]		

Quand une fonction arrive à sa fin sans rencontrer return, Python renvoie None. Ce n'est pas le nombre zéro. Avec le code initial erroné, quel test du tableau met ce défaut en évidence ?

.....

.....

Sécuriser une répétition et expliquer

Réparer une boucle qui ne finit pas 10 min

On veut ajouter des matériels jusqu'à atteindre un stock de 3. N'exécutez pas ce programme tel quel : suivez seulement ses premiers passages.

```
stock = 1
while stock < 3:
    print(stock)
print("Prêt")
```

Valeurs de stock avant les trois premiers tests : _____, _____, _____.
La condition peut-elle devenir fausse ? _____. Quelle mise à jour manque ?

.....

.....

Complétez d'abord le pseudo-code, puis écrivez Python. Dans la version corrigée, on veut afficher uniquement le stock final, une seule fois.

```
stock ← 1
Tant Que __________ Faire
    stock ← __________
Fin Tant Que
Afficher __________
```

Vérifiez avec un départ à 1, puis à 3. Indiquez le nombre de passages et la valeur finale. Pour interrompre une boucle lancée par erreur, utilisez Arrêter ou Interrompre.

.....

.....

Créer un petit rapport de réparation 12 min

Règle : un stock de 0 à 2 inclus doit afficher "Commander" ; au-delà, "Suffisant". Les stocks sont des entiers positifs ou nuls.

```
stock = 2
if stock < 2:
    print("Commander")
else:
    print("Suffisant")
```

Écrivez au dos : résultat attendu et résultat réel pour 2 ; cause précise ; pseudo-code corrigé ; Python corrigé ; deux autres tests avec leurs résultats attendus. Présentez votre réparation à un voisin sans lui dicter le code.

Bilan individuel 3 min

Avant de corriger, je dois connaître _____.
Après une correction, je _____.
Je sais expliquer une erreur : seul / avec une trace / avec une aide.

Aides graduées du module 10

Syntaxe et indices

Syntaxe — Aide 1 : regardez la fin du for et l'alignement de son bloc.

Aide 2 : ajoutez les deux-points et décalez print de quatre espaces.

Indices — Aide 1 : deux éléments donnent les indices 0 et 1.

Aide 2 : range(len(codes)) suffit. Variante valable : for valeur in codes: puis print(valeur).

Aide 3 avec les pseudo codes complets

```
Pour i de 0 à 2 Faire
    Afficher i
Fin Pour

codes ← ["PC", "IMP"]
Pour Chaque valeur dans codes Faire
    Afficher valeur
Fin Pour
```

Retour et boucle conditionnelle

Retour — Aide 1 : return quitte immédiatement la fonction.

Aide 2 : placez-le au même niveau que for, en restant dans la fonction.

Boucle — Aide 1 : afficher ne change pas stock.

Aide 2 : ajoutez 1 au stock à chaque passage ; affichez après la boucle.

Aide 3 avec les pseudo codes complets

```
Fonction compter_vides(stocks : liste) : entier
    nombre ← 0
    Pour Chaque stock dans stocks Faire
        Si stock = 0 Alors
            nombre ← nombre + 1
        Fin Si
    Fin Pour
    Retourner nombre
Fin Fonction

stock ← 1
Tant Que stock < 3 Faire
    stock ← stock + 1
Fin Tant Que
Afficher stock
```

Rapport — Aide 1 : le cas égal à 2 doit être accepté par le test. Aide 2 : utilisez <=. Aide 3 : Si stock <= 2 Alors Afficher "Commander" Sinon Afficher "Suffisant" Fin Si.

Défi du module 10 – Prépa CCF

Environ 10 minutes • Deux erreurs dans une recherche

La fonction doit renvoyer Vrai si le texte contient au moins un "!", sinon Faux. Elle ne doit pas s'arrêter avant d'avoir trouvé le signe ou terminé la recherche.

```
def contient_alerte(texte):
    for caractere in texte:
        if caractere != "!":
            return True
        return False
```

Sans exécuter, indiquez les résultats attendus et réels pour "!", "A!" et "". Complétez ensuite le pseudo-code corrigé et traduisez-le en Python.

```
Fonction contient_alerte(texte : chaîne) : booléen
    Pour Chaque caractere dans texte Faire
        Si __________ Alors
            Retourner __________
        Fin Si
    Fin Pour
    Retourner __________
Fin Fonction
```

.....

.....

Aides graduées

Aide 1 : on peut conclure Vrai dès que "!" est trouvé. Un autre caractère ne prouve pas que le signe est absent du reste du texte.

Aide 2 : corrigez la comparaison et placez Retourner Faux après le parcours.

Aide 3 avec le pseudo code complet

```
Fonction contient_alerte(texte : chaîne) : booléen
    Pour Chaque caractere dans texte Faire
        Si caractere = "!" Alors
            Retourner Vrai
        Fin Si
    Fin Pour
    Retourner Faux
Fin Fonction
```