Corrigé détaillé des modules 9 et 10

Guide enseignant • BTS SIO1 • Synthèse globale et débogage

Deux séances d'une heure consolident les acquis avant un entraînement au CCF. Aucun nouvel outil Python n'est indispensable. Distribuer le mémo séparé ; garder les aides accessibles, en invitant à les consulter une à une. Les défis sont facultatifs et exclus des 60 minutes.

Organisation des séances

Module	Parcours commun en minutes
9	Reconnaître 6 • Comprendre 10 • Compléter 10 • Traduire 12 • Déboguer 7 • Créer 12 • Bilan 3 = 60
10	Méthode 5 • Syntaxe 8 • Indices 10 • Trace 12 • while 10 • Rapport 12 • Bilan 3 = 60

Chaque dossier élève compte cinq pages : trois pour les activités, une pour les aides et une pour le défi. Écriture sur le document pour les réponses courtes ; au dos ou dans l'éditeur pour les programmes. Sans poste, traces et résultats attendus permettent de travailler toute la séance.

Module 9 reconnaître et comprendre

Reconnaître : nombre = 0 mémorise zéro dans nombre ; if priorite == "U": teste si la priorité vaut "U" pour décider d'exécuter son bloc ; for priorite in priorites: lit les valeurs de la liste une à une ; while stock < 3: répète son bloc tant que le stock est inférieur à 3 ; nombre = compter_urgentes(priorites) appelle la fonction avec la liste et mémorise son résultat dans nombre. Accepter toute formulation juste avec les mots de l'élève, sans exiger les termes techniques. Les extraits sont indépendants ; le dernier illustre un appel, pas une définition.

Trace du compteur : 0, 1, 1, 2 ; retour 2. Initialisation avant le parcours, test dans la boucle, retour après. Un retour à l'intérieur empêcherait le traitement complet. Le compteur mesure des occurrences, pas la somme de valeurs.

Module 9 compléter et traduire

Trous dans l'ordre : 0 ; priorites ; priorite = "U" ; nombre + 1 ; nombre. L'affectation et l'égalité emploient deux écritures différentes en Python.

```
def compter_urgentes(priorites):
    nombre = 0
    for priorite in priorites:
        if priorite == "U":
            nombre = nombre + 1
    return nombre

print(compter_urgentes(["N", "U", "N", "U"]))
```

Tests dans l'ordre de la fiche : 2, 0, 1, 0. La liste vide vérifie zéro passage et le retour du compteur initial. Une définition seule ne produit aucun affichage.

Module 9 corrections et prolongement

Déboguer la condition

Pour ["N", "U", "N"], attendu 1, produit 2 : le code compte les non-urgences. Remplacer != par ==. Avec ["N", "U"], les deux effectifs valent 1 : le test masque le défaut. Demander un exemple où les effectifs diffèrent.

Créer une fonction sur une chaîne

```
Fonction compter_marques(texte : chaîne) : entier
    nombre ← 0
    Pour Chaque caractere dans texte Faire
        Si caractere = "!" Alors
            nombre ← nombre + 1
        Fin Si
    Fin Pour
    Retourner nombre
Fin Fonction
```

```
def compter_marques(texte):
    nombre = 0
    for caractere in texte:
        if caractere == "!":
            nombre = nombre + 1
    return nombre
```

Tests : "Alerte! Vérifier!" → 2 ; "" → 0 ; par exemple "RAS" → 0 ou "!!" → 2. Les espaces sont parcourus mais ne satisfont pas le test. Accepter un nom de variable différent si le sens est clair.

Défi sur les créneaux

```
priorites = ["N", "U", "U", "U"]
places = 2
attente = 0
for priorite in priorites:
    if priorite == "U":
        if places > 0:
            places = places - 1
        else:
            attente = attente + 1
print(places, attente)
```

Trous : priorite = "U" ; places > 0 ; places - 1 ; attente + 1. Pour les données proposées : places 0 et attente 1. Pour [] : places 2 et attente 0. Un N ne modifie rien ; un U consomme un créneau ou rejoint l'attente, jamais les deux. Le pseudo-code complet figure dans l'aide élève.

Bilan et points à observer

Bilan : for valeur in liste: et return. Vérifier surtout que les élèves savent distinguer valeur et indice, expliquer le compteur et proposer un cas vide. Si le temps manque, terminer la traduction de compter_marques à la séance suivante ; conserver la phase de prévision des tests.

Module 10 syntaxe et indices

Faire adopter une démarche explicite

Avant toute modification, demander le résultat voulu et un exemple précis. Un message d'erreur ne signifie pas que tout le programme est faux. Une exécution sans message ne prouve pas que le résultat est juste. Ne pas noter la mémorisation des noms anglais des exceptions.

Programme qui ne démarre pas

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

Sorties : 0, 1, 2, chacune sur une ligne. Deux corrections : deux-points après range(3) ; indentation du print. Selon l'éditeur, l'erreur de fin de ligne est signalée avant celle du bloc. Le pseudo-code complet se trouve dans les aides élèves.

Erreur pendant le parcours par indices

Trace : i = 0 lit "PC", lecture possible ; i = 1 lit "IMP", lecture possible ; i = 2 demande codes[2], qui n'existe pas. Le programme affiche PC et IMP, puis s'arrête avec IndexError. len(codes) vaut 2 ; cela ne crée pas un indice 2.

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

Autre correction acceptable : for valeur in codes: puis print(valeur). Avec codes = [], aucun passage et aucun affichage. Ne pas remplacer par une borne fixe : la correction doit aussi fonctionner si la liste change de longueur.

Relancer des tests utiles

Après correction, rejouer ["PC", "IMP"], puis [] et éventuellement ["PC"]. Attendus : deux lignes, aucune ligne, une ligne. La réparation doit respecter la longueur réelle, sans masquer le défaut en supprimant un élément.

Module 10 retour et boucle conditionnelle

Retour placé trop tôt

Attendu pour [0, 2, 0] : 2. Trace réelle : initialisation nombre = 0 ; premier passage stock = 0, nombre = 1 ; return nombre quitte la fonction. Le deuxième élément n'est pas lu. Résultat réel : 1.

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

Déplacer return au niveau de for, dans la fonction. Les tests corrigés donnent 2, 0, 0. Avec le code erroné, [] donne None, car aucun passage n'exécute return. [2, 3] donne déjà 0 : ce test seul ne révèle pas l'erreur. Le pseudo-code corrigé est dans l'aide 3 élève.

Boucle sans mise à jour

Dans le code erroné, stock vaut 1 avant les trois premiers tests et reste à 1. La condition reste vraie ; "Prêt" n'est jamais atteint. Ne pas exécuter cette boucle sans moyen de l'interrompre.

```
stock = 1
while stock < 3:
    stock = stock + 1
print(stock)
```

Trous du pseudo-code : stock < 3 ; stock + 1 ; stock. Départ 1 : deux passages, stock final 3, affiché une fois. Départ 3 : zéro passage, stock final 3, affiché une fois. Un simple print dans le bloc ne modifie aucune variable.

Choisir une correction qui respecte la demande

Ajouter une mise à jour tout en conservant print(stock) dans la boucle terminerait le programme, mais afficherait plusieurs valeurs. La consigne demande uniquement le stock final : l'affichage doit être placé après la boucle. Distinguer disparition d'un symptôme et respect de la spécification.

Module 10 rapport et défi final

Exemple de rapport de réparation

Attendu pour stock = 2 : "Commander". Résultat réel : "Suffisant". Cause : le test stock < 2 exclut la frontière 2. Correction : remplacer < par <=. Tests complémentaires : 0 → "Commander" ; 3 → "Suffisant". Accepter 1 à la place de 0, mais encourager le choix des frontières.

```
Si stock <= 2 Alors
    Afficher "Commander"
Sinon
    Afficher "Suffisant"
Fin Si
```

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

Défi sur la présence d'un signe

Texte	Attendu	Réel avant correction
"!"	True	False
"A!"	True	True
""	False	None

Deux erreurs : != renvoie Vrai dès qu'un caractère ordinaire est lu ; le return False dans la boucle conclut trop tôt. "A!" donne le bon résultat pour une mauvaise raison. Ajouter "A" → False et "A!" → True après correction permet de distinguer ces situations.

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

Trous du pseudo-code : caractere = "!" ; Vrai ; Faux. Vrai peut être renvoyé dès qu'un signe est trouvé. Faux ne peut être conclu qu'après le parcours entier. Cette sortie anticipée est donc volontaire, contrairement au compteur de stocks nuls.

Préparer le passage au CCF

Pour la prochaine séance, observer si l'élève sait expliquer son choix de boucle, faire un appel de fonction et proposer un test qui révèle une erreur. Conserver le mémo pour un premier sujet guidé, puis retirer progressivement les aides. Les modules ne reproduisent aucun sujet officiel.