Corrigé détaillé des fiches 3 et 4

BTS SIO1 • Deux séances de 60 minutes • Document enseignant

Ces deux séances consolident le choix entre plusieurs cas, puis introduisent les répétitions. Le pseudo-code sert de support au raisonnement avant la traduction en Python. Chaque fiche reste autonome pour l'impression et la mise en téléchargement.

Organisation et prérequis

Distribuer 4 pages pour la fiche 3 et 6 pages pour le module 4 révisé. Dans le module 4, les pages 5 et 6 proposent le défi et l'extra facultatifs. Ils ne s'ajoutent pas au parcours d'une heure. La complétion dure 6 minutes et le passage à i et range dure 10 minutes. Laisser les aides accessibles après un premier essai. Les réponses longues se font au dos ou dans l'éditeur Python 3.

Prérequis : affectations, mises à jour simples, if / else. Aucun input(), aucune liste ni fonction ne sont requis. Les données des activités sont des entiers positifs ou nuls ; la demande du défi 3 est strictement positive. Ne pas ajouter une validation des entrées pendant ces séances.

Déroulés minutés

Étape	Fiche 3	Fiche 4
Réactivation	0 à 5 min	0 à 5 min
Comprendre	5 à 15 min	5 à 15 min
Compléter	15 à 25 min	15 à 21 min
Traduire	25 à 35 min	21 à 31 min
Déboguer	35 à 45 min	31 à 38 min
Créer	45 à 57 min	38 à 47 min
Découvrir while	—	47 à 57 min
Bilan individuel	57 à 60 min	57 à 60 min

Objectifs et différenciation

Fiche 3 : justifier l'ordre des tests ; choisir une seule branche ; vérifier les frontières ; distinguer = et == en Python. Module 4 : distinguer i, le nombre de passages et le stock ; comparer range(4) et range(1, 5) ; placer les instructions dans le bon bloc ; expliquer le test et l'arrêt d'un while.

Les durées sont des repères. Si la fiche 3 ralentit : corriger le débogage collectivement. Si la fiche 4 ralentit : guider oralement la création ou la terminer au cours suivant ; conserver une lecture courte du while avant toute future synthèse. Ne pas présenter while comme acquis après cette seule découverte.

Dans chaque défi, masquer l'aide complète pendant le premier essai. Après son utilisation, demander une justification de ligne et un test avec une autre donnée. Observer séparément raisonnement, syntaxe et niveau d'aide.

Suite envisagée

Une fiche 5 pourra reprendre if, for et while avec un exercice de transfert, après vérification des acquis sur while. Le format « comme au CCF » sera calibré à partir des sujets blancs fournis. Les chaînes et listes pourront suivre dans les fiches 6 ou 7 selon les retours de classe.

Module 3 Réponses et explications

Réactiver et comprendre

Réactivation : stock vaut 3 ; stock <= 3 est vraie ; affichage : print(stock). Le signe = affecte une valeur en Python ; == compare deux valeurs.

Stock	Branche choisie	Message
0	Premier test : stock = 0	Rupture
1	Deuxième test : stock <= 3	À surveiller
3	Deuxième test : stock <= 3	À surveiller
4	Sinon	Stock suffisant

Pour 0, les deux comparaisons seraient vraies si on les évaluait séparément. Dans une chaîne Si / Sinon Si / Sinon, la première branche vraie suffit : le second test n'est pas évalué. C'est pourquoi l'ordre est décisif.

Compléter et traduire

Trous : = en pseudo-code, puis <=. Les valeurs 0, 3 et 4 contrôlent respectivement le cas nul, le seuil inclus et le cas au-dessus du seuil.

```
stock = 2
if stock == 0:
    print("Rupture")
elif stock <= 3:
    print("À surveiller")
else:
    print("Stock suffisant")
```

Attendus pour les tests 0, 1, 3 et 4 : Rupture ; À surveiller ; À surveiller ; Stock suffisant. Vérifier que les trois mots if, elif et else sont alignés et suivis des deux-points.

Déboguer

Le programme fautif affiche À surveiller pour stock = 0, puisque 0 <= 3 est vrai. Le elif n'est alors pas testé. Corriger en plaçant stock == 0 et print("Rupture") en premier, puis stock <= 3 avec son message : c'est le programme ci-dessus. Il faut déplacer les tests avec les messages correspondants.

Module 3 Corrigé de la création

```
Pseudo-code
demande ← 2
Si demande = 0 Alors
    Afficher "Aucune demande"
Sinon Si demande <= 2 Alors
    Afficher "Traitement direct"
Sinon
    Afficher "Validation nécessaire"
Fin Si

Python
demande = 2
if demande == 0:
    print("Aucune demande")
elif demande <= 2:
    print("Traitement direct")
else:
    print("Validation nécessaire")
```

Tests : 0 → Aucune demande ; 2 → Traitement direct ; 3 → Validation nécessaire. Bilan : une seule branche est exécutée dans cette chaîne comportant else ; l'égalité Python s'écrit ==.

Module 3 Corrigé du défi

Compléter le pseudo-code

Trous dans l'ordre : demande > places ; demande = places ; places - demande ; places - demande ; places. Le cas impossible est refusé avant toute soustraction ; un refus conserve le stock de places.

```
places ← 8
demande ← 8
Si demande > places Alors
    Afficher "Réservation refusée"
Sinon Si demande = places Alors
    places ← places - demande
    Afficher "Réservation acceptée, salle complète"
Sinon
    places ← places - demande
    Afficher "Réservation acceptée"
Fin Si
Afficher places
```

Traduction en Python

```
places = 8
demande = 8
if demande > places:
    print("Réservation refusée")
elif demande == places:
    places = places - demande
    print("Réservation acceptée, salle complète")
else:
    places = places - demande
    print("Réservation acceptée")
print(places)
```

Demande	Message	Places restantes
3	Réservation acceptée	5
8	Réservation acceptée, salle complète	0
9	Réservation refusée	8

Repartir de places = 8 avant chaque test. L'affichage du stock est hors des branches pour être exécuté dans tous les cas. Dans le cas d'égalité, places - demande vaut 0. Une solution équivalente avec une autre organisation correcte des tests est recevable.

Relances utiles

Si l'élève soustrait avant de refuser : lui faire tracer la demande 9 et observer le nombre de places. S'il utilise = dans le elif Python : revenir au couple affecter / comparer. Après l'aide complète, demander pourquoi print(places) n'est pas indenté.

Module 4 Corrigé de for

Réactiver et comprendre

Réactivation : 3 puis 4. La mise à jour utilise la valeur courante à chaque fois.

Moment	numero	stock
Avant la boucle	Non défini	2
Passage 1	1	3
Passage 2	2	4
Passage 3	3	5
Passage 4	4	6

L'ajout est exécuté 4 fois ; le stock est affiché une seule fois, après la boucle ; valeur affichée : 6. numero est la variable de parcours ; stock est la variable mise à jour. Elles n'ont pas le même rôle.

Compléter et traduire

Borne : 3 ; expression : stock - 1. Évolution : 6 → 5 → 4 → 3. Une souris sort à chaque passage ; le résultat final est 3.

```
Pseudo-code
stock ← 6
Pour i de 0 à 2 Faire
    stock ← stock - 1
Fin Pour
Afficher stock

Python
stock = 6
for i in range(3):
    stock = stock - 1
print(stock)
```

range(3) fournit 0, 1, 2 : trois passages. Le pseudo-code initial de la page 1 numérote de 1 à 3 ; la mise à jour du stock ne dépend pas de ce numéro et produit le même résultat. Accepter aussi range(1, 4). Le dernier print est hors du bloc pour afficher seulement le résultat final.

Déboguer

Sorties actuelles, une par ligne : 1 ; Terminé ; 2 ; Terminé ; 3 ; Terminé. La borne 4 exclut le numéro 4, et Terminé se trouve dans le bloc répété.

```
for i in range(1, 5):
    print(i)
print("Terminé")
```

Sorties corrigées : 1 ; 2 ; 3 ; 4 ; Terminé. Modifier uniquement la borne ne suffit pas : il faut aussi sortir le dernier affichage du bloc.

Module 4 Corrigé de la création

```
Pseudo-code
stock ← 10
Pour numero de 1 à 4 Faire
    stock ← stock - 1
Fin Pour
Afficher stock

Python
stock = 10
for numero in range(1, 5):
    stock = stock - 1
print(stock)
```

Valeurs successives : 9, 8, 7, 6. Stock final : 6. Le nombre de passages est connu avant la boucle. Accepter range(4) si l'élève explique les quatre passages, même si la numérotation commence alors à 0.

Module 4 Comprendre while

Trace et arrêt

stock avant le test	stock < 5	Conséquence
2	Vraie	Ajout : stock devient 3
3	Vraie	Ajout : stock devient 4
4	Vraie	Ajout : stock devient 5
5	Fausse	Sortie de la boucle

Le corps est exécuté 3 fois, mais la condition est testée 4 fois. Le programme affiche 5 une seule fois. Avec stock = 5 dès le départ, la condition est fausse immédiatement : zéro passage et affichage de 5.

Compléter la variante

```
Pseudo-code
stock ← 4
Tant Que stock < 6 Faire
    stock ← stock + 1
Fin Tant Que
Afficher stock

Python
stock = 4
while stock < 6:
    stock = stock + 1
print(stock)
```

Trous : stock < 6 ; stock + 1. Deux passages, stock 5 puis 6 ; le test devient faux à 6. En remplaçant la mise à jour par print(stock), stock reste à 2 : le programme affiche 2 sans fin car la condition reste vraie. Il faut interrompre l'exécution, puis rétablir la mise à jour.

Bilan attendu

range(3) donne 0, 1, 2 : trois passages. La condition de while est vérifiée avant chaque passage. Pour l'arrêt, surveiller la condition et l'évolution des variables qu'elle utilise. Ici, stock doit évoluer jusqu'à rendre stock < 5 faux.

Ne pas confondre les deux boucles

Dans les exemples de for, les passages sont déterminés par range. Dans l'exemple de while, le programme reteste stock avant de décider de continuer. Ne pas généraliser en disant que while garantit l'arrêt : c'est la condition et la mise à jour choisies qui le rendent possible ici.

Ajuster la prochaine séance

Faire distinguer « passage du corps » et « test de la condition ». Si cette différence reste fragile, ouvrir la prochaine séance avec une trace à deux passages et un cas à zéro passage. Avant une synthèse if / for / while, demander à chacun de choisir une structure et de justifier ce choix en une phrase.

Relevé rapide

Noter A = autonome, H = avec aide, R = à reprendre pour : nombre de passages ; borne de range ; indentation ; mise à jour ; arrêt de while. Le défi reste facultatif et ne sert pas à conclure que toute la classe maîtrise l'imbrication.

Module 4 Corrigé du défi

Compléter et traduire

Trous : 5 ; stock > 0 ; stock - 1 ; stock. La boucle représente les cinq personnes ; la condition est réévaluée pour chacune avec le stock actualisé.

```
stock ← 3
Pour numero de 1 à 5 Faire
    Si stock > 0 Alors
        stock ← stock - 1
        Afficher "Prêt accepté"
    Sinon
        Afficher "Prêt refusé"
    Fin Si
Fin Pour
Afficher stock
```

```
stock = 3
for i in range(5):
    if stock > 0:
        stock = stock - 1
        print("Prêt accepté")
    else:
        print("Prêt refusé")
print(stock)
```

Vérifier les trois situations

Stock initial	Acceptés	Refus	Stock final
3	3	2	0
5	5	0	0
0	0	5	0

Avec stock = 3, les trois premiers passages retirent un casque : stock vaut 2, 1, puis 0. Aux passages 4 et 5, le test devient faux ; les prêts sont refusés et stock reste à 0. Le stock ne descend jamais sous 0 dans les situations proposées.

Avec stock = 5, les cinq prêts sont acceptés. Avec stock = 0, le for effectue quand même cinq passages : chaque demande est refusée. Le dernier print s'exécute une seule fois après la boucle.

Lire les niveaux d'indentation

Le for est à l'alignement de départ. Le if et le else sont décalés de quatre espaces, car ils appartiennent à la boucle. Les instructions des branches sont décalées de huit espaces. Le dernier print revient à l'alignement de départ.

Une réussite avec le pseudo-code complet reste utile : demander ensuite de prévoir le résultat pour un stock initial de 1 (un prêt accepté, quatre refus, stock final 0). Cette vérification distingue la copie du code de sa compréhension.

Module 4 Comprendre i et les formes de range

Comparaison du parcours principal

i est un nom de variable, comme numero. range(4) fournit 0, 1, 2, 3 ; il provoque quatre passages. Après chaque ajout, stock vaut 3, 4, 5, 6. Le programme affiche 6 une fois. Le premier passage a lieu pour i = 0 : la valeur 0 ne signifie pas « aucun passage ».

Passage	i avec range(4)	i avec range(1, 5)	stock après l'ajout
Premier	0	1	3
Deuxième	1	2	4
Troisième	2	3	5
Quatrième	3	4	6

Les deux programmes repartent de stock = 2. Ils produisent le même résultat car i n'est pas utilisé dans la mise à jour stock = stock + 1. Si le bloc utilisait i pour calculer, les résultats pourraient différer. La traduction de la distribution des souris utilise range(3), retourne un stock de 3, et affiche cette valeur après la boucle.

Corrigé de l'extra facultatif

range(2, 9, 2) : départ 2, borne exclue 9, pas 2. Valeurs : 2, 4, 6, 8. La valeur 9 n'est pas atteinte par ce pas ; la borne finale est de toute façon exclue. Avec range(2, 8, 2), les valeurs sont 2, 4, 6 : 8 est exclu.

Écriture	Valeurs	Passages
range(5)	0, 1, 2, 3, 4	5
range(3, 7)	3, 4, 5, 6	4
range(1, 8, 2)	1, 3, 5, 7	4
range(4, 4)	Aucune valeur	0

Petit défi avec un pas

```
Pseudo-code
Pour i de 2 à 10 avec un pas de 2 Faire
    Afficher i
Fin Pour

Python
for i in range(2, 11, 2):
    print(i)
```

Affichages : 2, 4, 6, 8, 10. Accepter range(2, 12, 2), qui donne ici les mêmes valeurs. Le pseudo-code utilise une borne incluse, Python une borne exclue. Un pas de 2 change les valeurs de i, pas automatiquement le stock ni le nombre d'unités ajoutées par le bloc.

L'extra est facultatif : le parcours commun installe les formes à un et deux paramètres. Pour vérifier la distinction entre indice de passage et stock, demander ce qui changerait si stock commençait à 10 : mêmes valeurs de i et mêmes quatre passages, stock final 14.