Guide et corrigé du modules 1 et 2

BTS SIO1 • Deux séances de 60 minutes • Document enseignant

Ce module installe l'affectation puis une condition à deux branches dans une situation de stock informatique. Il associe lecture, complétion, traduction, débogage et création. Les nombres restent petits pour concentrer l'effort sur l'algorithmique.

Préparer et distribuer

Imprimer les trois premières pages de chaque fiche élève ; la page 4 est un défi facultatif pour les plus rapides, à proposer dans le temps restant ou en prolongement. La page 3 réunit le mémo et les aides ; elle peut être distribuée séparément ou posée en libre accès. Prévoir un interpréteur Python 3 si disponible (Thonny pour ma part), possibilité d'utiliser basthon en ligne. Toutes les activités restent réalisables sur papier. Aucun recours à input(), aux boucles ou aux fonctions n'est requis.

Prérequis à vérifier : lire une suite d'instructions, additionner et soustraire de petits entiers. Les fiches reprennent explicitement les notions utiles. Ne pas supposer une maîtrise acquise après les deux premières séances de l'année.

Objectifs observables

Séance 1 : suivre une variable après chaque affectation ; distinguer texte et valeur ; produire une suite d'instructions cohérente dans les deux langages. Séance 2 : identifier la branche exécutée ; inclure le seuil ; traduire avec une indentation correcte ; tester sous le seuil, au seuil et au-dessus.

Déroulé de la séance 1

Minutes	Travail et conduite de classe
0 à 10	Comprendre : 3 min de lecture accompagnée puis trace individuelle.
10 à 22	Compléter : recherche puis comparaison en binôme.
22 à 27	Mise en commun et mémo : verbaliser la mise à jour.
27 à 37	Traduire : prévoir avant de lancer.
37 à 47	Déboguer : exiger une explication des deux corrections.
47 à 57	Créer : pseudo-code puis Python, seul ou en binôme.
57 à 60	Bilan individuel sans aide.

Déroulé de la séance 2

0–5 min : réactivation ; 5–15 : comprendre ; 15–25 : compléter ; 25–37 : traduire et tester ; 37–45 : déboguer ; 45–57 : créer ; 57–60 : bilan. Présenter <= oralement avant la première trace ; accompagner le passage au Python pendant la traduction.

Si le rythme ralentit : conserver la trace, la traduction et le bilan. Corriger le débogage collectivement ; faire terminer la version Python de la création au prochain cours. Les prolongements restent facultatifs et ne conditionnent pas la réussite du module.

Corrigé du module 1

Comprendre

Après l'instruction	stock	prets
stock ← 8	8	Non définie
prets ← 3	8	3
stock ← stock - prets	5	3

Affichage : 5. La partie droite utilise les valeurs courantes 8 et 3 ; le résultat remplace 8 dans stock. Ne pas accepter « stock garde 8 et 5 » : une variable n'accumule pas ses anciennes valeurs.

Compléter et traduire

Cases : prets ; retours ; stock. Résultat final : 8. Le prêt retire 4 unités (10 → 6), le retour en ajoute 2 (6 → 8).

```
stock = 10
prets = 4
retours = 2
stock = stock - prets
stock = stock + retours
print(stock)
```

Déboguer

Ligne 3 : stock = stock - prets. L'affectation initiale copiait seulement 3 dans stock. Ligne 4 : print(stock). Avec les guillemets, on affichait le mot stock. Programme corrigé :

```
stock = 8
prets = 3
stock = stock - prets
print(stock)
```

Résultat : 5. Faire distinguer erreur d'intention et erreur de syntaxe : les deux lignes d'origine sont syntaxiquement valides.

Créer

```
Pseudo-code
stock ← 6
livraison ← 5
distribution ← 2
stock ← stock + livraison
stock ← stock - distribution
Afficher stock

Python
stock = 6
livraison = 5
distribution = 2
stock = stock + livraison
stock = stock - distribution
print(stock)
```

Résultat : 9. Accepter d'autres noms explicites et une expression combinée correcte. Bilan : a vaut 5. Prolongement : a vaut 9 et b vaut 2 ; b a reçu la valeur de a à cet instant, pas un lien vivant vers a.

Corrigé du module 2

Réactivation et compréhension

Stock final : 5 ; affichage Python : print(stock).

stock	stock <= 3	Message
2	Vraie	Commander
3	Vraie	Commander
4	Fausse	Stock suffisant

Compléter et traduire

Condition : stock <= 3. Message de la branche Sinon : Stock suffisant. Le test avec 3 vérifie que le seuil est bien inclus. Avec stock = 5, la branche Sinon est exécutée.

```
stock = 5
if stock <= 3:
    print("Commander")
else:
    print("Stock suffisant")
```

Pour les tests 2, 3 et 4 : Commander ; Commander ; Stock suffisant. Vérifier aussi les deux-points et les quatre espaces, sans introduire encore les conditions composées.

Déboguer

Le cas mal traité est stock = 3 : < donne faux et affiche Stock suffisant. Remplacer < par <=. La syntaxe était correcte ; c'est la traduction de la règle qui ne l'était pas.

Créer

```
Pseudo-code
stock ← 7
prets ← 4
stock ← stock - prets
Afficher stock
Si stock <= 3 Alors
    Afficher "Commander"
Sinon
    Afficher "Stock suffisant"
Fin Si

Python
stock = 7
prets = 4
stock = stock - prets
print(stock)
if stock <= 3:
    print("Commander")
else:
    print("Stock suffisant")
```

Avec 4 prêts : afficher 3 puis Commander. Avec 3 prêts : afficher 4 puis Stock suffisant. Bilan : vraie ; Commander ; les lignes décalées appartiennent à la branche. Prolongement : seuil = 3 puis if stock <= seuil: ; avec seuil = 5, un stock de 4 déclenche une commande.

Corrigé des défis finaux

Ces défis sont facultatifs et ne s'ajoutent pas au parcours obligatoire de 60 minutes. Les proposer aux élèves qui ont terminé et vérifié leurs réponses ; ils peuvent aussi servir de prolongement.

Module 1 Conserver le stock de départ

```
Pseudo-code
stock ← 9
stock_depart ← stock
stock ← stock - 3
stock ← stock + 1
Afficher stock_depart
Afficher stock

Python
stock = 9
stock_depart = stock
stock = stock - 3
stock = stock + 1
print(stock_depart)
print(stock)
```

Résultats : 9 puis 7. Avec un stock initial de 12 : 12 puis 10. La variable stock_depart reçoit une copie de la valeur initiale ; modifier stock ne modifie pas stock_depart. Accepter des variables supplémentaires pour les prêts et les retours.

Module 2 Contrôler le prêt avant de retirer

```
Pseudo-code
stock ← 5
demande ← 3
Si demande <= stock Alors
    stock ← stock - demande
    Afficher "Prêt accepté"
Sinon
    Afficher "Prêt refusé"
Fin Si
Afficher stock

Python
stock = 5
demande = 3
if demande <= stock:
    stock = stock - demande
    print("Prêt accepté")
else:
    print("Prêt refusé")
print(stock)
```

Demande 3 : Prêt accepté, stock 2. Demande 5 : Prêt accepté, stock 0. Demande 6 : Prêt refusé, stock 5. Le dernier print est hors des branches, sans indentation. Chaque test repart de stock = 5. Les quantités proposées sont des entiers positifs ; leur validation n'est pas demandée.