EXERCICES

2de

Algorithmique et programmation Python

Nom : _____
Classe : _____

> Au contrôle, une réponse juste sans calcul ne rapporte presque rien : chaque problème se rédige en quatre temps — **justifier**, **poser** l'égalité, **mener le calcul** jusqu'au bout, **conclure** par une phrase avec l'unité. La place réservée est dimensionnée pour ces quatre temps.

1 Suivre un programme à la main

●○○

a. `a = 3` puis `b = a + 2` puis `a = 10`. Que valent `a` et `b` à la fin ?
b. `n = 7` puis `n = n * 2` puis `n = n - 4`. Que vaut `n` ?
c. Combien de fois s'exécute le corps de `for i in range(3, 9)` ?

2 Corriger un programme

●●○

a. `if x = 5:` — quelle erreur ?
b. Une fonction se termine par `print(m)` mais son résultat, réutilisé plus loin, vaut `None`. Pourquoi ?
c. `n = 10` puis `while n > 0:` puis `print(n)`. Que se passe-t-il ?

3 Écrire une boucle for

●●○

a. Écrire un programme qui calcule la somme des entiers de 1 à 100.
b. Le modifier pour ne sommer que les nombres pairs.
c. Vérifier le premier résultat par une formule.

4 Une fonction avec return

●●○

a. Écrire une fonction `est_pair(n)` qui renvoie `True` si `n` est pair.

b. Écrire une fonction `compte_pairs(liste)` qui renvoie le nombre d'éléments pairs, en réutilisant la première.

5 Algorithme d'Euclide

●●●

a. Dérouler l'algorithme sur 91 et 35.

b. Combien de tours de boucle ?

c. Que se passe-t-il si on l'applique à 17 et 5 ?

6 Test de primalité

●●●

a. Pour tester si 97 est premier, jusqu'à quel diviseur faut-il aller si l'on s'arrête à $\sqrt{n}$?

b. Justifier qu'il est inutile d'aller au-delà.

c. Combien de tests économise-t-on par rapport à un parcours jusqu'à 96 ?

Corrigé détaillé sur **progresschool.fr** — chapitre « Algorithmique et programmation Python ».