CORRIGÉ

1re

Python — Programmation orientée objet et récursivité

Corrigé détaillé
4 exercices

> Ce qui est noté au contrôle, ce n'est pas le résultat seul : c'est le raisonnement posé, puis le calcul mené jusqu'au bout avec l'unité.

1 Vocabulaire. Répondre en une phrase précise.

1. a. Elle donne un **second nom** au même objet ; elle ne le **copie pas**.
2. b. La **classe** est un **modèle** décrivant attributs et méthodes ; l'**instance** est un objet construit à partir de ce modèle.
3. c. Regrouper données et opérations dans un objet et n'exposer qu'une interface. Son intérêt est de garantir un état **cohérent**, non de dissimuler.
4. d. Un **cas de base** sans appel récursif, et un **appel récursif** qui s'en rapproche **strictement**.
5. e. À signaler une situation que la fonction ne peut pas traiter elle-même, pour qu'un appelant décide quoi faire.

2 Une fonction reçoit une liste et y ajoute un élément. L'appelant constate que sa propre liste a changé.

1. a. Parce que le paramètre et la variable de l'appelant **nomment le même objet**. La fonction n'a pas reçu une copie : elle a reçu un **second nom**.
2. b. **Non**. Un entier est **immuable** : on ne peut pas le modifier en place, seulement faire pointer le nom vers un autre entier — ce qui n'affecte pas l'appelant.
3. c. En **copiant explicitement** la liste avant de la modifier, et en travaillant sur la copie.
4. d. Parce qu'ils testent d'abord sur des **entiers** et des **chaînes**, qui sont **immuables** : le phénomène ne s'y manifeste pas. Ils en tirent une règle générale fausse, que les listes viennent contredire plus tard.

3 Récursivité. Raisonner sur la terminaison et le coût.

1. a. **Cas de base** : la **factorielle** de 0 vaut 1. **Cas général** : la factorielle de n vaut n multiplié par la factorielle de n − 1.
2. b. En déroulant :
   $4 \times 3 \times 2 \times 1 = 24$
3. Soit **24**.
4. c. Les appels **s'empilent sans fin** jusqu'à saturer la **pile d'appels**, et le programme s'interrompt sur une erreur de profondeur.
5. d. Elle est **correcte** — elle renvoie les bonnes valeurs — et **inutilisable** au-delà de quelques dizaines de termes, parce qu'elle **recalcule** les mêmes valeurs un nombre considérable de fois.
6. e. Que **correction** et **efficacité** sont deux propriétés **distinctes**, et qu'une fonction peut satisfaire la première sans la seconde. Tester qu'un programme donne le bon résultat ne dit rien de sa capacité à le donner sur des entrées réalistes.

4 Objet et exceptions. Analyser.

1. a. Le test est « **est un** ». Un carré **est un** rectangle : l'**héritage** se discute. Une voiture **n'est pas** un moteur : elle en **possède** un, c'est une **composition**.
2. b. Parce que si toute modification passe par une **méthode**, celle-ci peut **vérifier** ce qu'elle reçoit. Un attribut modifiable de l'extérieur ne permet aucune vérification.
3. c. Parce que le programme **masque ses erreurs** et continue dans un état incohérent : il échouera plus tard, ailleurs, de façon incompréhensible. Un **arrêt net** est préférable à un échec silencieux.
4. d. Parce que le code importé est **testé** par bien plus d'utilisateurs que le vôtre ne le sera, et souvent optimisé.
5. e. **Non**. Une dépendance ajoute une **surface** à maintenir, doit être à jour, et peut disparaître ou changer d'interface. Pour une fonction de trois lignes, l'importer coûte plus qu'il ne rapporte. L'argument vaut pour ce qui est **difficile à écrire correctement** — dates, encodages, calcul numérique —, pas pour tout.

Corrige-toi toi-même. Compte tes points exercice par exercice : c'est ainsi qu'on voit où l'on perd, et ce n'est presque jamais là où on le croit.

Ce qui est évalué	Points	Ce qui fait perdre le point
Expliquer qu'une affectation crée un alias	5 pts	Croire qu'affecter une liste la copie
Distinguer classe et instance	3 pts	Confondre le modèle et l'objet
Justifier un héritage par la relation est un	4 pts	Faire hériter une voiture d'un moteur

Écrire une récursivité qui se termine	5 pts	Omettre le cas de base
Distinguer correction et efficacité	3 pts	Conclure qu'un résultat juste suffit