

Python — Programmation orientée objet et récursivité

NSI · Première
Contrôle n°1 · 60 min · sur 20

Exercice 1 — Noms et objets

4 pts 11 min

Distinguer alias et copie

Réponds à chaque question.

LA RÈGLE DU COURS

Une affectation ne fait pas ce que beaucoup croient, et l'effet ne se manifeste pas sur tous les types.

1. Que fait une affectation de variable ? (1 pt)

Elle donne un second nom au même objet ; elle ne le copie pas.

→ Modifier l'un modifie donc ce que voit l'autre.

2. Pour quels types la question ne se pose-t-elle pas, et pourquoi ? (1 pt)

Pour les objets immuables — entiers, chaînes, tuples — puisqu'on ne peut pas les modifier en place.

→ C'est pourquoi l'effet déroute : les débutants testent d'abord sur des entiers.

3. Dans quels trois cas la question se pose-t-elle pour un objet modifiable ? (1 pt)

À chaque affectation, à chaque passage en argument de fonction et à chaque valeur par défaut.

→ Une fonction qui modifie une liste reçue modifie celle de l'appelant.

4. Comment obtenir une vraie copie indépendante ? (1 pt)

Il faut la demander explicitement.

→ Aucune copie n'est faite implicitement par l'affectation.

Exercice 2 — Classes et héritage

5 pts 14 min

Analyser la programmation orientée objet

Réponds à chaque question.

LA RÈGLE DU COURS

Un modèle, des objets construits à partir de lui, et une relation de spécialisation qui se vérifie par un test.

1. Qu'est-ce qu'une classe, et qu'est-ce qu'une instance ? (1 pt)

Une classe est un modèle décrivant attributs et méthodes ; une instance est un objet construit à partir de ce modèle.

→ Les attributs sont les données, les méthodes les opérations.

2. Que partagent et que ne partagent pas deux objets d'une même classe ? (1 pt)

Ils partagent leurs méthodes et possèdent chacun ses valeurs d'attributs.

→ Le constructeur initialise les attributs propres à chaque instance.

3. Qu'est-ce que l'encapsulation ? (1 pt)

Le fait de regrouper dans un même objet les données et les opérations qui les manipulent, et de n'exposer qu'une interface d'usage.

→ Son intérêt n'est pas la dissimulation.

4. À quoi sert-elle réellement ? (1 pt)

À garantir que l'objet reste dans un état cohérent.

→ Si toute modification passe par une méthode, cette méthode peut vérifier ce qu'elle reçoit.

5. Quel test permet de décider s'il faut employer l'héritage ? (1 pt)

Le test « est un » : un carré est un rectangle, donc l'héritage se discute.

→ Une voiture n'est pas un moteur : elle en possède un, et c'est une composition.

Exercice 3 — La récursivité

4 pts 11 min

Écrire et analyser une fonction récursive

Réponds à chaque question.

LA RÈGLE DU COURS

Une fonction qui s'appelle elle-même exige deux éléments, et l'oubli du premier a une conséquence précise.

1. Quels deux éléments une fonction récursive exige-t-elle ? (1 pt)

Un cas de base, qui renvoie une valeur sans appel récursif, et un appel récursif qui se rapproche strictement du cas de base.

→ L'oubli du premier est l'erreur la plus fréquente.

2. Que se passe-t-il sans progression stricte vers le cas de base ? (1 pt)

Les appels s'empilent jusqu'à saturer la pile d'appels.

→ Chaque appel non terminé occupe une place dans cette pile.

3. Écris en français le cas de base et le cas général de la factorielle. (1 pt)

Cas de base : la factorielle de 0 vaut 1 ; cas général : la factorielle de n vaut n multiplié par la factorielle de n - 1.

→ La progression est stricte, n diminuant de 1 à chaque appel.

4. Pourquoi une récursivité naïve sur Fibonacci est-elle correcte et inutilisable ? (1 pt)

Parce qu'elle recalcule les mêmes valeurs un nombre considérable de fois.

→ Correction et efficacité sont deux propriétés distinctes.

Exercice 4 — Exceptions et modules

4 pts 13 min

Employer exceptions et modules

Réponds à chaque question.

LA RÈGLE DU COURS

Une situation qu'une fonction ne peut pas traiter se signale, et deux règles d'usage évitent le pire.

1. Que signale une exception ? (1 pt)

Une situation qu'une fonction ne peut pas traiter elle-même.

→ Fichier absent, division par zéro ou conversion impossible en sont des exemples.

2. Quelles sont les deux règles d'usage du bloc try/except ? (1 pt)

Rattraper le type d'exception précis que l'on sait traiter, et ne jamais rattraper toutes les exceptions pour n'en rien faire.

→ Un programme qui masque ses erreurs échoue silencieusement.

3. Pourquoi un échec silencieux est-il pire qu'un arrêt net ? (1 pt)

Parce que le programme continue en produisant des résultats sans signaler qu'une opération a échoué.

→ L'arrêt net, lui, désigne le point du problème.

4. Pourquoi importer un module existant plutôt que réécrire une fonction ? (1 pt)

Parce que le code importé est testé par bien plus d'utilisateurs que le vôtre ne le sera.

→ Ce n'est donc pas de la paresse mais une question de fiabilité.

Repérer une erreur et la corriger

Chaque affirmation est fausse. Écris ce qu'il fallait dire.

LA RÈGLE DU COURS

Trois fautes reviennent. Une copie supposée, un héritage mal fondé, et un rattrapage trop large.

1. « J'ai affecté ma liste à une nouvelle variable : j'ai donc une copie indépendante. » (1 pt)

Les deux noms désignent le même objet en mémoire : toute modification faite par l'un est visible par l'autre.

→ Une copie doit être demandée explicitement.

2. « Ma classe Voiture hérite de la classe Moteur, puisqu'une voiture a un moteur. » (1 pt)

Une voiture n'est pas un moteur : elle en possède un, et c'est une composition.

→ L'héritage se vérifie par le test « est un ».

3. « Je rattrape toutes les exceptions pour que mon programme ne s'arrête jamais. » (1 pt)

Il faut rattraper le type précis que l'on sait traiter, et effectivement le traiter.

→ Un programme qui masque ses erreurs échoue silencieusement, ce qui est pire qu'un arrêt net.

Barème — ce que le correcteur regarde

Les lignes chiffrées couvrent les 5 exercices et totalisent 20 points.

EXERCICE	ATTENDU	POINTS	POINT PERDU SI
Ex.1	Noms, objets et copies	4 points — 1 par question	Une affectation prise pour une copie.
Ex.2	Classes, encapsulation, héritage	5 points — 1 par question	Un héritage qui ne passe pas le test « est un ».
Ex.3	La récursivité	4 points — 1 par question	Un cas de base absent ou jamais atteint.
Ex.4	Exceptions et modules	4 points — 1 par question	Toutes les exceptions rattrapées sans rien en faire.
Ex.5	L'erreur repérée, et la réponse juste donnée	3 points — 1 par ligne	Une correction qui laisse l'alias non voulu.

ET MAINTENANT ?

16 à 20

Le chapitre est acquis. Entraîne-toi maintenant à écrire des fonctions récursives en cinq temps : le cas de base d'abord, le cas général en fonction d'un cas plus petit, la progression stricte vérifiée, les deux premiers cas testés, puis la profondeur atteinte.

10 à 15

Refais l'exercice 1 en te demandant chaque fois si l'objet manipulé est modifiable ou immuable.

moins de 10

Reprends la fiche de cours et retiens la phrase qui explique beaucoup de bugs : affecter une variable ne copie pas l'objet, cela lui donne un second nom.