

18.1. Introduction: cas de test

Un **cas de test** exprime les exigences d'un programme, d'une manière qui peut être vérifiée automatiquement. Plus précisément, un test affirme quelque chose sur l'état du programme à un moment donné de son exécution.

Nous avons précédemment suggéré que c'est une bonne idée d'écrire d'abord des commentaires sur ce que votre code est censé faire, avant d'écrire réellement le code. Il est encore plus judicieux d'écrire certains cas de test avant d'écrire un programme.

Il y a plusieurs raisons pour lesquelles c'est une bonne habitude d'écrire des cas de test.

- Avant d'écrire du code, nous avons à l'esprit ce qu'il *devrait* faire, mais ces pensées peuvent être un peu vagues. Ecrire des cas de test nous oblige à être plus concrets sur ce qui doit se passer.
- Au fur et à mesure que nous écrivons le code, les cas de test peuvent fournir des commentaires automatisés. Vous avez en fait bénéficié de ces retours automatisés via des cas de test tout au long de ce livre dans certaines des fenêtres de code actif et dans presque tous les exercices. Nous avons écrit le code de ces cas de test, mais nous l'avons gardé caché, afin de ne pas vous confondre et aussi d'éviter de donner les réponses. Vous pouvez obtenir le même avantage en écrivant vos propres cas de test.
- Dans les projets logiciels plus importants, l'ensemble de cas de test peut être exécuté chaque fois qu'une modification est apportée à la base de code. **Les tests unitaires** vérifient que les petits morceaux de code sont correctement implémentés. **Les tests fonctionnels** vérifient que de plus gros morceaux de code fonctionnent correctement. L'exécution des tests peut aider à identifier les situations dans lesquelles un changement de code à un endroit interrompt le bon fonctionnement d'un autre code. Nous ne verrons pas cet avantage des cas de test dans ce manuel, mais gardez à l'esprit que cette introduction aux cas de test prépare le terrain pour une pratique essentielle du génie logiciel si vous participez à un projet de développement logiciel plus vaste.

Il est maintenant temps d'apprendre à écrire du code pour les cas de test.

Python fournit une instruction appelée `assert`.

- Après le mot `assert`, il y aura une expression python.
- Si cette expression est évaluée à la valeur booléenne `False`, l'interpréteur lèvera une erreur d'exécution.
- Si l'expression donne la valeur `True`, rien ne se passe et l'exécution passe à la ligne de code suivante.

Pourquoi voudriez-vous jamais écrire une ligne de code qui ne peut jamais calculer quoi que ce soit d'utile pour vous, mais qui provoque parfois une erreur d'exécution? Pour toutes les raisons que nous avons décrites ci-dessus à propos de la valeur des tests automatisés. Vous voulez un test qui vous alertera qu'une condition que vous supposiez être vraie n'est en fait pas vraie. Il vaut bien mieux être alerté de ce fait tout de suite que d'avoir un résultat inattendu beaucoup plus tard dans l'exécution de votre programme, que vous aurez du mal à retrouver à l'endroit où vous avez eu une erreur dans votre code.

Pourquoi `assert` n'imprime pas quelque chose indiquant que le test a réussi? La raison en est que vous ne voulez pas encombrer votre fenêtre de sortie avec les résultats des tests automatisés qui réussissent. Vous voulez juste savoir quand l'un de vos tests échoue. Dans les projets plus importants, d'autres harnais de

scénarios de test > (toctree.html) 18.2. Vérification des hypothèses sur les types de données > Section suivante 18.2. Vérification des hypothèses sur les types de données (TestingTypes.html)

test sont utilisés à la place de `assert`, comme le `unittest` module python. Ceux-ci fournissent des résultats résumant les tests qui ont réussi ainsi que ceux qui ont échoué. Dans ce manuel, nous utiliserons simplement des `assert` déclarations simples pour les tests automatisés.

Pour écrire un test, nous devons savoir ce que nous *attendons d'*une valeur à un moment donné de l'exécution du programme. Dans le reste du chapitre, nous verrons quelques exemples d' `assert` énoncés et d'idées sur les types d'assertions que l'on pourrait vouloir ajouter dans ses programmes.

Remarque

Une note aux instructeurs: ce chapitre est délibérément structuré afin que vous puissiez introduire les tests tôt dans le cours si vous le souhaitez. Vous devrez couvrir le chapitre 8, sur les conditions, avant de commencer ce chapitre, car ce chapitre couvre les booléens. Les sous-chapitres sur les types de test et les conditions de test peuvent être traités juste après. Le sous-chapitre sur les fonctions de test peut être retardé jusqu'à ce que vous ayez couvert la définition des fonctions.

Vérifie ta compréhension

test-1-1: Quand est exécuté et que x et y ont des valeurs différentes, que va-t-il se passer? `assert x==y`

- ☒ A. Une erreur d'exécution se produira
- ☐ B. Un message est imprimé indiquant que le test a échoué.
- ☐ C. x obtiendra la valeur que y a actuellement
- ☐ D. Rien ne se passera
- ☐ E. Un message est imprimé indiquant que le test a réussi.

Vérifiez moi

Comparez moi

✓ L'expression `"x == y"` est évaluée à `False`, donc une erreur d'exécution se produira

Activity: 1 -- Choix multiple (question19_1_1)

test-1-2: Quand est exécuté et que x et y ont les mêmes valeurs, que va-t-il se passer? `assert x==y`

- ☐ A. Une erreur d'exécution se produira
- ☐ B. Un message est imprimé indiquant que le test a échoué.
- ☐ C. x obtiendra la valeur que y a actuellement
- ☒ D. Rien ne se passera

Un message est imprimé indiquant que le test a réussi.

18.2. Vérification des hypothèses sur les types de données (TestingTypes.html)

Vérifiez moi

Comparez moi

narios de test "> (toctree.html) Section suivante - 18.2. Vérification

✓ L'expression `` x == y`` est évaluée à `` True``

Activity: 2 -- Choix multiple (question19_1_1b)

test-1-3: Les cas de test sont une perte de temps, car l'interpréteur python donnera un message d'erreur lorsque le programme ne s'exécute pas correctement, et c'est tout ce dont vous avez besoin pour le débogage.

☐ Une véritable

☒ B. Faux

Vérifiez moi

Comparez moi

✓ Les cas de test vous permettent de tester certains morceaux de code au fur et à mesure que vous les écrivez, plutôt que d'attendre que les problèmes se manifestent plus tard.

Activity: 3 -- Choix multiple (question19_1_2)

Vous avez tenté 4 des 3 activités de cette page

18. Scénarios de test ">



(toctree.html)

18.2. Vérification des hypothèses concernant les types de données ">



(TestingTypes.html)

✓ Terminé. Bien joué!