Cours sur les tests statique

Avec Python3 et le module Pylint

20/10/2023
Baptiste MEGEVAND

Table des matières

I - Introduction de la qualité d'assurance (QA) dans le développement de logiciels

A. Définition de la QA en développement de logiciels

La qualité d'assurance (QA) dans le contexte du développement de logiciels fait référence à un ensemble de processus et de méthodologies conçus pour améliorer et assurer la qualité des produits logiciels. Cette pratique englobe diverses stratégies, techniques, et outils mis en place à différentes phases du cycle de développement pour garantir que le produit final répond aux normes de qualité établies.
La QA est une approche proactive qui vise à prévenir les défauts dans le produit final en guidant le processus de développement dès le début. Elle ne se limite pas seulement à la détection des bugs ou des problèmes. Au lieu de cela, elle s'efforce de créer un environnement où chaque aspect du cycle de vie de développement, de la conception initiale, le développement, jusqu'aux tests, la documentation, et la sortie du produit, est contrôlé et standardisé pour réduire au maximum les anomalies.

B. Importance de la QA pour la maintenance du code et la réduction des bugs

1. **Prévention des bugs plutôt que correction** : L'un des principaux avantages de la QA est qu'elle se concentre sur la prévention des bugs plutôt que sur leur correction après leur apparition. Intégrer la QA dès les premières étapes du développement permet d'identifier et de résoudre les problèmes avant qu'ils ne progressent vers les phases ultérieures, économisant ainsi du temps et des ressources.
2. **Amélioration de la fiabilité du logiciel** : En assurant que tous les aspects du logiciel sont minutieusement testés et qu'ils adhèrent à des normes de qualité spécifiques, la QA contribue à construire un produit plus fiable. Un logiciel avec moins de bugs et de problèmes techniques offre une meilleure expérience utilisateur, renforçant ainsi la confiance et la satisfaction des clients.
3. **Efficacité des coûts** : Bien que la mise en place de processus de QA puisse nécessiter un investissement initial, la détection précoce et la prévention des défauts peuvent réduire considérablement les coûts de réparation à long terme. Le coût de la correction des bugs augmente considérablement lorsqu'ils sont détectés à des stades ultérieurs du développement ou après la sortie du produit.
4. **Facilitation des mises à jour et de la maintenance** : Un code propre et bien structuré, soutenu par des pratiques de QA, est plus facile à mettre à jour et à maintenir. En standardisant le code et en s'assurant que les pratiques de développement sont suivies, la QA facilite l'introduction de nouvelles fonctionnalités et la gestion des versions successives du logiciel.
5. **Conformité et normes** : La QA aide à assurer que les produits logiciels respectent les réglementations et les normes de l'industrie. Cela est particulièrement important dans des domaines tels que la santé, la finance, et l'automobile, où les logiciels doivent souvent respecter des normes strictes de conformité et de sécurité.

En somme, la QA est une composante essentielle du développement de logiciels qui soutient la création de produits performants, fiables, et professionnels. En intégrant des pratiques de QA, les équipes de développement peuvent non seulement améliorer la qualité de leurs produits finaux, mais aussi optimiser leurs processus de développement, économisant ainsi du temps, des efforts, et des coûts à long terme.

II - Comprendre l'analyse statique du code

A. Qu'est-ce que l'analyse statique ?

L'analyse statique, dans le contexte du développement de logiciels, est une forme d'analyse de qualité qui s'effectue sans exécuter le programme. Elle implique l'examen du code source pour détecter les erreurs, bugs, ou vulnérabilités potentielles en utilisant des outils et méthodes qui évaluent divers aspects du code. Cela peut inclure la syntaxe, l'utilisation de la mémoire, les conventions de codage, et d'autres indicateurs de qualité ou de performance. L'objectif principal est d'identifier les problèmes qui pourraient ne pas être évidents à première vue et de les corriger avant qu'ils ne se manifestent dans les environnements de production.

B. Avantages de l'analyse statique dans la QA

1. **Détection précoce des erreurs** : L'analyse statique permet de détecter les erreurs et les incohérences dès le stade de développement, bien avant les tests d'exécution, facilitant une intervention rapide et une correction moins coûteuse.
2. **Sécurité améliorée** : En identifiant les vulnérabilités potentielles comme les fuites de mémoire ou les injections de code, l'analyse statique contribue à renforcer la sécurité du logiciel, un aspect crucial dans de nombreux domaines d'application.
3. **Maintenabilité du code** : En favorisant un code propre et conforme aux bonnes pratiques, l'analyse statique facilite les futures interventions sur le code, qu'il s'agisse de maintenance, d'ajouts fonctionnels, ou de débogage.
4. **Réduction du temps de révision** : Les outils d'analyse statique automatisent une partie de la révision du code, permettant aux développeurs de se concentrer sur des tâches plus complexes et de réduire le temps global de révision et de test.
5. **Conformité aux normes** : L'analyse statique aide les équipes à respecter les normes de codage prédéfinies et les meilleures pratiques de l'industrie, en assurant une certaine homogénéité et qualité du code.

C. Comment l'analyse statique améliore la qualité du code

L'analyse statique joue un rôle crucial dans l'amélioration de la qualité du code en imposant une rigueur et une discipline dans le processus de développement. Voici comment elle contribue concrètement à cette amélioration :

1. **Uniformité et cohérence** : En utilisant des outils d'analyse statique, les équipes de développement peuvent s'assurer que le code respecte un ensemble cohérent de normes et de pratiques, ce qui conduit à une base de code plus uniforme et donc plus facile à comprendre et à maintenir.
2. **Complexité réduite** : Ces outils peuvent signaler les endroits où le code est trop complexe, aidant les développeurs à le refactoriser et à simplifier les structures ou les fonctions compliquées, ce qui réduit les chances d'erreur et améliore la lisibilité.
3. **Focus sur la qualité dès le début** : En intégrant l'analyse statique dès les premières étapes du cycle de développement, les équipes mettent l'accent sur la qualité dès le début, plutôt que de compter sur la correction des bugs après coup. Cela crée une culture de la qualité, où les développeurs sont plus conscients des problèmes potentiels et travaillent de manière proactive pour les éviter.

4. **Meilleure compréhension du code** : En soulignant les domaines problématiques, les développeurs gagnent une compréhension plus profonde des défis et des nuances de leur code, ce qui les aide à prendre de meilleures décisions de programmation à l'avenir.

III - Introduction à Pylint

A. Qu'est-ce que Pylint ?

Pylint est un outil d'analyse de code source très populaire dans la communauté Python. Il scanne le code pour détecter les erreurs de syntaxe, les problèmes stylistiques, les structures de code complexes ou redondantes, et les anomalies qui pourraient indiquer des bugs potentiels. Pylint est hautement configurable et extensible, offrant aux développeurs la possibilité d'ajuster les règles de vérification et les conventions de codage selon les besoins spécifiques de leur projet.

Conçu comme un outil de qualité de code, Pylint aide à maintenir la cohérence et la propreté dans des bases de code Python, en encourageant les développeurs à suivre les meilleures pratiques de codage et les conventions stylistiques standardisées, notamment celles définies dans le guide de style Python PEP 8.

B. Caractéristiques principales de Pylint

- **Analyse de conformité** : Pylint vérifie la conformité du code avec le guide de style Python PEP 8, qui définit les conventions de codage pour la lisibilité et la cohérence du code Python. Il signale les écarts par rapport à ces normes, aidant ainsi les équipes à maintenir une base de code uniforme.
- **Détection d'erreurs** : Au-delà des problèmes de style, Pylint identifie les erreurs de programmation courantes, certaines desquelles pourraient être non détectées lors des tests habituels. Cela comprend les variables inutilisées, les importations inutiles, les erreurs de syntaxe, et plus encore.
- **Refactoring de code** : Pylint propose des suggestions pour améliorer la structure du code sans en changer le comportement. Cela comprend la simplification des boucles et des conditions, la réduction de la complexité des fonctions, et la suppression du code redondant.
- **Vérification de la complexité du code** : L'outil analyse la complexité du code en se basant sur différentes métriques. Il aide à identifier les parties du code qui sont excessivement compliquées, suggérant ainsi des zones pour le refactoring.
- **Extensibilité** : Pylint permet aux développeurs de personnaliser les règles de l'analyseur en fonction de leurs préférences ou des exigences du projet. Il est possible de désactiver certaines vérifications ou d'ajouter des vérifications personnalisées.
- **Intégration avec des environnements de développement** : Pylint peut être intégré dans des environnements de développement intégrés (IDE) et d'autres outils pour fournir des analyses en temps réel, améliorant ainsi l'efficacité du développement.

IV - Pratique

1. Installation et configuration de base de Pylint :

1.1 Installation :

- Utilisez pip pour installer pylint : *pip install pylint*
- Pour vérifier l'installation : *pylint --version*

1.2 Configuration de base :

Pylint utilise un fichier pylintrc pour la configuration. Pour générer un fichier de configuration par défaut : *pylint --generate-rcfile > ~/.pylintrc*

2. Exécution de Pylint sur un script Python simple :

2.1 Exemple de script

```
def example1():
    print('Hello, world!')
```

2.2 Exécution de Pylint:

Pour lancer pylint sur le script : *pylint your_script_name.py*

3. Interprétation des résultats de Pylint:

3.1 Scores :

- Pylint évaluera votre code et lui attribuera une note de 10.
- Un score de 10 signifie que votre code est "parfait" selon Pylint.

3.2 Messages :

- Pylint retourne des messages de **type C (Convention), R (Refactor), W (Warning), E (Error) et F (Fatal)**.
- Il donne également des suggestions pour améliorer la qualité du code.

4. Personnalisation des contrôles de Pylint:

4.1 Désactivation des messages :

- Pour désactiver un message spécifique : *pylint your_script_name.py --disable=message_id*
- Vous pouvez également désactiver plusieurs messages en séparant les id par des virgules.

4.2 Configuration des règles :

- Pour configurer les règles, modifiez le fichier *.pylintrc*
- Exemple : Changer la longueur maximale d'une ligne à 120. Modifiez la valeur max-line-length sous la section [FORMAT].

5. Utilisation des commentaires inline pour contrôler Pylint:

5.1 Désactiver une règle pour une ligne spécifique:

```
print('This is a long line that will cause a warning')  # pylint: disable=line-too-long
```

5.2 Désactiver une règle pour un bloc de code:

```
# pylint: disable=unused-variable
x = 10
y = 20
# pylint: enable=unused-variable
```

6. Intégration de Pylint dans un environnement de développement intégré (IDE):

Visual Studio Code:

- Assurez-vous d'avoir installé l'extension Python pour VS Code.
- Ouvrez les paramètres et recherchez "Python Linting". Activez "Enabled" et choisissez "Pylint" comme linter.