УДК 004.415.532.3

А.С. Васюра
І.В. Богач
І.В. Вернік

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ НА МОВІ ПРОГРАМУВАННЯ PYTHON

Вінницький національний технічний університет

Анотація

У даній роботі приділено увагу до сучасних тенденцій у сфері автоматизованого тестування, наведено його переваги та недоліки, проведено аналіз мови програмування Python, створено приклад для демонстрації використання цієї мови, а також наведено сучасні Frameworks, що використовують у тестуванні.

Ключові слова: автоматизоване тестування, Python, Framework, unittest, nose, pytest

Abstract

In this article considered the analysis of current trends in the field of automated testing, its advantages and disadvantages are given, Python programming language analysis is analyzed, an example is created for demonstration of the use of this language, and also the modern Frameworks used in the testing are presented..

Keywords: automated testing, Python, Framework unittest, nose, pytest

Вступ

Компанія, що видає програмний продукт повинна відповідає за його якість перед кінцевими споживачами, для цього перед випуском продукту, обов'язково проводяться всі можливі стадії тестування. Раніше все тестування проводилось в ручну, але на сьогоднішній день значного розвитку набуло автоматизоване тестування, яке дозволяє спростити ряд завдань та пришвидшити виконання більшості тестів, саме тому актуальним на даний час є дослідження оптимізації написання автоматизованих тестів.

Результати дослідження

Автоматизоване тестування передбачає використання спеціального програмного забезпечення (крім тестованого) для контролю виконання тестів і порівняння очікуваного та фактичного результату роботи програми. Цей тип тестування допомагає автоматизувати часто повторювані, але необхідні для максимізації тестового покриття завдання [1].

Після створення автоматизованих тестів, їх можна в будь-який момент запустити знову, причому запускаються і виконуються вони швидко і точно. Таким чином, якщо є необхідність частого повторного прогону тестів, значення автоматизації для спрощення супроводу проекту і зниження його вартості важко переоцінити. Адже навіть мінімальні зміни коду можуть стати причиною появи нових багів.

Автоматизація процесу тестування несе в собі як переваги так і недоліки:

– Автоматизація зберігає час, сили і гроші. Одного разу автоматизований тест можна запускати знову і знову, докладаючи мінімум зусиль.

– Вручну можна протестувати практично будь-який додаток, в той час як автоматизувати варто тільки стабільні системи. Деякі види тестування, наприклад, ad-hoc чи дослідне тестування можуть бути виконані тільки вручну.

– Мануальне тестування може бути повторюваним і нудним. У той же час, автоматизація може допомогти цього уникнути – за вас все зробить комп'ютер.

Тому, залежно від характеристики продукту що буде розробляться, вид тестування потрібно обирати завчасно. Також варто одразу визначитись з мовою програмування на якій будуть створюватись автоматичні тести – повноцінні програми, які призначені для тестування.

Python підходить для автоматизації більше, ніж інші мови програмування завдяки своїм характеристикам: він інтерпретується, простий у вивченні і більш лаконічний. Ця мова

кросплатформена, тому, за рідкісним винятком, усі додатки, написані на ньому, будуть працювати НА будь-якій системі. Серед плюсів також великий набір бібліотек і активна підтримка спільнотою [2].

Простий приклад, нехай ми маємо процедуру знаходження суми квадратів

```
def sum_kv(x, y):
    return x*x+y*y
```

Процедура тестування цієї функції буде мати вигляд

```
def test_our_proc():
    assert sum_kv(2, 3) == 13
    assert not sum_kv(3,4)==10
```

Також Python має ряд Framework'ів для проведення автоматизованого тестування [3]:

– unittest – це framework для тестування, що входить в стандартну бібліотеку мови Python. Його архітектура виконана в стилі xUnit.

– nose, девізом nose є фраза "nose extends unittest to make testing easier", що можна перекласти як "nose розширює unittest, роблячи тестування простіше". Nose ідеальний, коли потрібно зробити тести "на швидку руку", без попереднього планування і вибудовування архітектури додатку з тестами. Функціонал nose можна розширювати і настроювати за допомогою плагінів [4].

– pytest досить потужний інструмент для тестування. Як було сказано вище, unittest в своїй базі – xUnit, що накладає певні зобов'язання при розробці тестів (створення класів-спадкоємців від unittest.TestCase, виконання певної процедури запуску тестів і т.п.). При розробці на pytest нічого з цього робити не потрібно, ви просто пишете функції, які повинні починатися з “test_” і використовуєте assert'и, вбудовані в Python (unittest використовується свої) [5].

При виборі фреймворку необхідно оцініть розмір і складність свого проекту. Якщо те, що ви розробляєте, являє собою велику систему з неабиякою кількістю функцій, то full-stack фреймворк - те, що вам потрібно. Якщо ж ви створюєте щодо невеликий додаток або сервіс, тоді варто звернути увагу на мікрофреймворкі.

В роботі наведено приклади тестувань – мануальне та автоматизоване, розглянуто переваги та недоліки кожного з них та переваги використання мови програмування Python та існуючих для неї Framework'и для тестування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. McMeekin K. Test Automation vs. Automated Testing: The Difference Matters [Електронний ресурс] / Kyle McMeekin. – 2017. – Режим доступу до ресурсу: https://www.qasymphony.com/blog/test-automation-automated-testing/ . - Назва з екрану.
2. Shaw A. Getting Started With Testing in Python [Електронний ресурс] / Anthony Shaw. – 2018. – Режим доступу до ресурсу: https://realpython.com/python-testing/. - Назва з екрану.
3. Тестирование в Python (unittest) [Електронний ресурс] // Devpractice – 2017. – Режим доступу до ресурсу: https://devpractice.ru/unit-testing-in-python-part-1/ - Назва з екрану.
4. Nose: тестирование в Python [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://proft.me/2010/12/23/nose-testirovanie-v-python/. - Назва з екрану.
5. Sanko A. PyTest [Електронний ресурс] / Alexey Sanko. – 2015. – Режим доступу до ресурсу: https://habr.com/ru/post/269759/. - Назва з екрану.

Васюра Анатолій Степанович *–д.т.н., академік УТА, професор кафедри автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця. e-mail: vasanat@i.ua.*

Богач Ілона Віталіївна *– кандидат технічних наук, доцент кафедри автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця. e-mail: ilona.bogach@gmail.com .*

Вернік Ірина Володимирівна *– студентка групи І-15б, факультет комп’ютерних систем та автоматики, Вінницький національний технічний університет, м. Вінниця, e-mail: vernik.iryna@gmail.com.*

Vasyura Anatoly *– Professor, academician of Ukrainian Technological Academy, Professor of automation and intelligent information technologies department, Vinnytsia National Technical University*, Vinnytsia, e-mail: vasanat@i.ua.

Ilona Bogach *– PhD, Associate Professor of automation and intelligent information technologies department, Vinnytsia National Technical University, Vinnytsia, e-mail: ilona.bogach@gmail.com.*

Vernik Iryna – Faculty of computer systems and automatics, Vinnytsia National Technical University, Vinnytsia, email: vernik.iryna@gmail.com.