НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №1
із дисципліни «Автоматизоване тестування програмного забезпечення»
на тему
Тестування веб-застосунків (web-application) за допомогою Selenium IDE.

Виконав:
студент групи КМ-82
Бубела Д. В.

Керівник:
асистент
Громова В. В.

Київ — 2020

ЗМІСТ

МЕТА РОБОТИ

Ознайомитися з «Selenium IDE». Навчитися використовувати це ПЗ для автоматизованого тестування сайтів, автоматизовувати роботу з веб застосунками в браузері.

Завдання 1

Створити скрипт (в термінології Selenium IDE – Test1), що виконує наступні дії:

1.1. Зайти на сайт НТУУ КПІ ім. Ігоря Сікорського https://kpi.ua

1.2. У відповідності до варіанту перейти на сторінку «Інститути» або «Факультети», користуючись існуючими посиланнями блоку СТРУКТУРА внизу головної сторінки https://kpi.ua

1.3. Користуючись існуючими посиланнями з поточної сторінки, у відповідності до варіанту перейти на сторінку заданого факультету / інституту.

1.4. Вивести в консоль назву факультету / інституту, одержану зі сторінки даного факультету / інституту.

1.5. Вивести в консоль посилання на офіційний сайт, розташоване одразу під назвою факультету / інституту.

1.6. Вивести в консоль повний url зображення, розташованого одразу під назвою факультету / інституту. Якщо зображення відсутнє або не завантажилось, вивести в консоль повідомлення «no image».

1.7. Вивести в консоль повний список кафедр заданого факультету / інституту.

1.8. Закрити браузер.

Завдання 2

1 ПОСТАНОВКА ЗАДАЧІ

Завдання 1

Програма повинна приймати ввід користувача для 2х точок по дві координати кожна. Потрібно створити функцію, що реалізовуватиме алгоритм який визначає, яка з точок знаходиться ближче до початку координат та повертає результат.

Завдання 2

Програма повинна реалізовувати функцію, що визначає мінімальний та максимальний елемент в масиві представленому списком та рахує суму між ними (включно).

Модульні тести повинні бути в окремому файлі та тестувати програму, що знаходиться в іншому файлі. Тести не повинні залежати один від одного.

2 ОСНОВНА ЧАСТИНА

Програми будуть створені таким чином, що дозволяло б їх тестування фреймворком «unittest», тобто основна логічна частина має бути оформлена в функцію, що повертає значення, за яким можна перевірити коректність роботи програми або алгоритму.

Завдання 1

Згідно умові завдання введення даних повинне бути користувачем, тобто з ’stdin’. Для забезпечення передачі корректних даних функції, що буде їх опрацьовувати програма 1, має бути створена додаткова функція fl_input, що не пропускає далі некоректного вводу від користувача. Також повинна бути реалізація інформування користувача про необхідні дії для забезпечення роботи програми. Основну логічну задачі програми буде виконувати функція closer(A, B). Вона приймає дві точки, на яких власне і буде застосовуватись мета програми. Коли відстань від одної з них до початку координат буде більшою за відстань іншої точки до початку координат — вона буде додаватись до спеціального списку c[] з точками з найменшою відстанню до початку координат. Якщо найменшої точки не буде існувати — список c[] буде пустим. Функція повертає список найменших точок.

Для роботи з тестами буде створено клас Prog_1Test що буде дочірнім від класу unittest.TestCase. Всередині класу будуть використовуватись методи assertEqual та assertRaises. Тестування запускається за допомогою unittest.main() коли програма тестування виконується як основна програма виконання.

Завдання 2

Буде створена функція sum_min_max(lst), яка буде визначати мінімальний

елемент з масиву, максимальний, зберігатиме їх індекси та рахуватиме суму слайсу списку з першим і останнім елементом, мінімуму та максимуму початкового списку відповідно. В завданні чітко не вказані дії при існуванні кількох максимумів та/або мінімумів, тому програма перевірятиметься на знаходження суми в будь-яких варіантах, тобто мінімум може вибиратися з мінімумів довільно, так само і з максимумами. Конкретна реалізація буде шукати перше входження відповідних елементів. Пошук індексів елементів здійснюватиметься функціями min та max з порядкових номерів елементів з додатковим аргументом функції що порівнює елементи потрібного списку.

Для роботи з тестами буде створено клас Prog_2Test що буде дочірнім від класу unittest.TestCase. Всередині класу будуть використовуватись методи assertEqual, assertIn та assertRaises. Тестування запускається за допомогою unittest.main() коли програма тестування виконується як основна програма виконання.

3 РОЗРОБКА ТЕСТ-КЕЙСІВ

а) перший елемент нумерованого списку;

б) другий елемент нумерованого списку;

в) тощо.

ВИСНОВКИ

Цей шаблон потрібно використовувати під час підготовки документації на кафедрі ПМА. У ньому враховано вимоги кафедри ПМА, які уточнюють положення ДСТУ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».

Текст висновків.

Текст висновків.

ПЕРЕЛІК ПОСИЛАНЬ

1. Selenium IDE Documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.selenium.dev/selenium-ide/docs/en/introduction/getting-started.

Додаток А

Текст програм, які тестуються

Лістинг файлу prog_1.py

```
def closer(A, B):
    c = []
    if A[0]**2 + A[1]**2 < B[0]**2 + B[1]**2:
        c.append(A)
    if A[0]**2 + A[1]**2 > B[0]**2 + B[1]**2:
        c.append(B)
    return c


def fl_input(message):
    isCorrect = False
    while not isCorrect:
        try:
            fl = float(input(message))
        except ValueError:
            print("Невірний формат вводу")
        else:
            isCorrect = True
    return fl


if __name__ == '__main__':
    print("Визначення точки з ")
    print("Введіть дані точки A (x1, y1)")
    x1 = fl_input("x1 = ")
    y1 = fl_input("y1 = ")
    print("Введіть дані точки B (x2, y2)")
    x2 = fl_input("x2 = ")
    y2 = fl_input("y2 = ")
    c = closer((x1, y1), (x2, y2))
    if len(c) == 0:
        print("Точки знаходяться на однаковій відстані")
    else:
        print(f"Точка з координатами {c[0]} знадиться ближче до початку координат")
```

Лістинг файлу prog_2.py

```
def sum_min_max(lst):
    if not lst:
        return 0
    minmax_idx = [min(range(len(lst)), key=lst.__getitem__),
                  max(range(len(lst)), key=lst.__getitem__)]
    minmax_idx.sort()
    return sum(lst[minmax_idx[0]:minmax_idx[1] + 1])
```

Додаток Б

Текст модулів, що реалізують автоматичне тестування

Лістинг файлу test_prog_1.py

```
import prog_1
import unittest


class Prog_1Test(unittest.TestCase):

    def test_simple_equality(self):
        self.assertEqual(prog_1.closer((2.2, 3.3), (-3.3, -2.2)), [])

    def test_int_compatibility(self):
        self.assertEqual(prog_1.closer((22, 33), (-31, -24)), [(-31, -24)])

    def test_big_numbers(self):
        self.assertEqual(prog_1.closer(
            (10000000000000000000.2, -90000000000000000000.4), (1221221, 32312321)),
            [(1221221, 32312321)])

    def test_negative_test(self):
        self.assertEqual(prog_1.closer((-9.0, 9.0), (1.1, 1.1)), [(1.1, 1.1)])

    def test_bad_data(self):
        self.assertRaises(TypeError, prog_1.closer, (-9.0, 9.0), (1.1, 'o'))


if __name__ == '__main__':
    unittest.main()
```

Лістинг файлу test_prog_2.py

```
import prog_2
import unittest


class Prog_2Test(unittest.TestCase):

    def test_straight_sum(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [1, -6, 0, 34, 434]
            ),
            462
        )

    def test_empty(self):
        self.assertEqual(
            prog_2.sum_min_max(
                []
            ),
            0
        )

    def test_one(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [41]
            ),
            41
        )

    def test_two(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [1, 2]
            ),
            3
        )

    def test_reverse_sum(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [1, 434, 0, 34, -6]
            ),
            462
        )

    def test_float_compatibility(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [1.32, -6.23, 0, 34, 434.001]
```

```
            ),
            461.77099999999996
        )

    def test_negative_test(self):
        self.assertEqual(
            prog_2.sum_min_max(
                [-794, -43, -8989, -11, -89]
            ),
            -9000
        )

    def test_multiple_minmax(self):
        self.assertIn(
            prog_2.sum_min_max(
                [-89, 1, 90, -89, 2, 90]
            ),
            [2, 5, 1, 3]
        )

    def test_nested_lst(self):
        self.assertRaises(TypeError, prog_2.sum_min_max, [12, [21, 32]])

    def test_bad_type_sum(self):
        self.assertRaises(TypeError, prog_2.sum_min_max, [1, 'o', 2])

    def test_bad_type_cmp(self):
        self.assertRaises(TypeError, prog_2.sum_min_max, [1, 2, 'o'])


if __name__ == '__main__':
    unittest.main()
```

Додаток В

Скріншоти результатів виконання автотестів

```
~/K/A/L/№/source $ python -m unittest -v test_prog_1.py
test_bad_data (test_prog_1.Prog_1Test) ... ok
test_big_numbers (test_prog_1.Prog_1Test) ... ok
test_int_compatibility (test_prog_1.Prog_1Test) ... ok
test_negative_test (test_prog_1.Prog_1Test) ... ok
test_simple_equality (test_prog_1.Prog_1Test) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
```

Рисунок В.1 – Тестування програми 1

```
~/K/A/L/№/source $ python -m unittest -v test_prog_2.py
test_bad_type_cmp (test_prog_2.Prog_2Test) ... ok
test_bad_type_sum (test_prog_2.Prog_2Test) ... ok
test_empty (test_prog_2.Prog_2Test) ... ok
test_float_compatibility (test_prog_2.Prog_2Test) ... ok
test_multiple_minmax (test_prog_2.Prog_2Test) ... ok
test_negative_test (test_prog_2.Prog_2Test) ... ok
test_nested_lst (test_prog_2.Prog_2Test) ... ok
test_one (test_prog_2.Prog_2Test) ... ok
test_reverse_sum (test_prog_2.Prog_2Test) ... ok
test_straight_sum (test_prog_2.Prog_2Test) ... ok
test_two (test_prog_2.Prog_2Test) ... ok

----------------------------------------------------------------------
Ran 11 tests in 0.000s

OK
```

Рисунок В.2 – Тестування програми 2