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

Звіт
із лабораторної роботи №2
із дисципліни «Автоматизоване тестування програмного забезпечення»
на тему
Приймальне тестування(acceptance testing) програм на мові Python

Виконав:
студент групи КМ-81
Донченко Б. М.

Керівник:
асистент Громова В. В.

Київ — 2020

ЗМІСТ

МЕТА РОБОТИ

Узагальнене формулювання завдань

1. Ознайомитись з основними функціями бібліотеки Python unittest.

2. Написати специфікацію до програми, що буде тестуватися (відповідно до варіанту), тобто сформулювати повний перелік вимог до розроблюваного програмного забезпечення;

3. Провести аналіз задачі та методів її розв’язання з метою виявлення виключних ситуацій, які можуть виникати під час роботи програми при застосуванні того чи іншого методу;

4. Розробити програму, відповідно до специфікації.

5. Розробити тестові випадки (test-cases), необхідні для:

- 100% покриття вимог до розроблюваного ПЗ;

- 100% покриття гілок рішень (100% decision coverage) – на основі графу потоку керування;

6. Створити тестовий модуль, що реалізує розроблені тест-кейси.

7. Перевірити роботу тестового модуля.

Індивідуальні завдання згідно з варіантом 28

Завдання 1. Визначити, чи задовольняє рядок наступній властивості: рядок містить (крім букв) тільки одну цифру, причому її числове значення дорівнює довжині рядка.

Завдання 2. У упорядкованому за зростанням масиві з'ясувати, більше елементів зі значенням >p або елементів зі значенням <p.

1.ПОСТАНОВКА ЗАВДАННЯ

Програма 1. Вхідними даними програми є рядок, який вводить користувач. За умовою, користувач повинен ввести рядок в якому буде лише 1 число, яке дорівнює кількості символів в рядку. Можна вводити будь-які символи. У випадку введення порожнього рядка (користувач натисне Enter) буде виводитися повідомлення *"Рядок порожній"*. Коли дані будуть введені коректно, з'явиться повідомлення *"Рядок задовольняє умові."*. Якщо рядок не містить чисел, то з'явиться повідомлення *"Рядок не задовольняє умові. У рядку немає чисел."*. Якщо користувач введе рядок, у якому не буде слів, а тільки пробіли, то з'явиться повідомлення *"Рядок не задовольняє умові. У рядку немає чисел."*. Якщо користувач введе рядок, у якому число не дорівнює кількості символів в рядку, то з'явиться повідомлення *"Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку."*. Якщо користувач введе рядок, у якому кількість чисел більше ніж 1, то з'явиться повідомлення *"Рядок не задовольняє умові. У рядку більше ніж одне число."*.

Програма 2. Вхідними даними функції є число (P), та кількість елементів масиву. Користувач вводить число та кількість елементів. Число може бути як цілим, так і дробовим, також від'ємним або додатнім. Кількість елементів повинно бути цілим, але якщо користувач введе число з плаваючою точкою, то кількість елементів дорівнює числу до коми. Можна вводити лише додатні числа, інакше з'явиться повідомлення *"Введено некоректні дані."* . Якщо ввести кількість елементів 0, то буде список з 0 елементів. Після цього починається введення елементів масиву, що приймають тип *float*. Якщо дані будуть введено некоректно, то з'явиться повідомлення *"Ви ввели неправильне число."*. Якщо все ввести правильно, то виводиться список, та повідомлення в залежності від списку: "У списку чисел, які більше ніж P, більше.", "У списку чисел, які менше ніж P, більше.", "У списку чисел, які більше та менше ніж P, однакова к-сть.".

2.ОСНОВНА ЧАСТИНА

Функціонал програми 1

З самого початку програма просить користувача ввести рядок. Рядок може будь-який. Коли користувач виконав це завдання, програма перекидає його рядок до функції ***task_1()***, в якій реалізоване саме завдання лабораторної роботи. Функція робить перевірку, якщо рядок порожній, то вона повертає повідомлення: "*Рядок не задовольняє умові. Рядок порожній.*". Якщо ж він не порожній, а навіть введені потрібні дані, то виконується перевірка кожного елементу рядка, а саме програма шукає число. Пошук числа виконаний за допомогою стандартної функції Python ***isdigit().*** Вона перевіряє чи елемент є натуральним числом чи ні, якщо так то вона переносить це число в список.

Після перевірки кожного елементу і вдалого заповнення списку програма перевіряє сам список. Якщо довжина списку дорівнює 0, то програма видає повідомлення "*Рядок не задовольняє умові. У рядку немає чисел.*", якщо більше, то "*Рядок не задовольняє умові. У рядку більше ніж одне число.*". Коли довжина дорівнює 1, то програма перевіряє чи воно дорівнює довжині рядку, який ввів користувач. Якщо воно дорівнює, то з'являється повідомлення: "*Рядок не задовольняє умові. У рядку більше ніж одне число.*", якщо ні, то: "*Рядок не задовольняє умові. У рядку більше ніж одне число.*".

Функціонал програми 2

Для написання цієї програми використовувалась бібліотека ***re*** для перевірки вхідних даних. Це потрібно з самого початку, коли користувач вводить число P, яке нам потрібно буде перевірити за умовою задачі. Для цього числа не використовуються сильні заборони, але користувач має ввести саме число, а не літеру або символ. Далі користувач повинен ввести кількість чисел в списку. Тут користувач має ввести тільки число, яке більше, або дорівнює нулю. Якщо користувач введе число дробове, то буде взято число перед комою, як головне. Якщо ж користувач ввів дані не коректно, то з'явиться повідомлення "*Введено некоректні дані.*". Далі програма передає дані в функцію ***task_2(),*** в якій реалізоване саме завдання лабораторної роботи. Далі програма просить ввести користувача числа, якими він хоче наповнити список. Якщо він введе не число, то з'явиться повідомлення: "*Ви ввели неправильне число.*". Після заповнення списку програма за допомогою циклу перевіряє кожне число списку. Якщо воно більше ніж задане, то програма додає до змінної **b_p** одиницю, якщо менше, то додає одиницю до змінної **l_p.** В кінці програма просто порівнює ці два числа, якщо перше більше, то програма повідомляє "*У списку чисел, які більше ніж P, більше.*", якщо навпаки "*У списку чисел, які менше ніж P, більше.*", а якщо кількість однакова: "*У списку чисел, які більше та менше ніж P, однакова к-сть.*".

3.РОЗРОБКА ТЕСТ-КЕЙСІВ

Програма 1

У таблиці 1 наведено назви тестових функцій, тестові дані та очікуваний результат програми 1.

Таблиця 1

№	*Тестова функція*	*Тестові дані*	*Результат*
1	*test_normal_input(self)*	*“flwee6”, “ffrtger8”, “dd dd dd9”, “ffa8 fff”*	*Рядок задовольняє умові.*
2	*test_empty_space(self*	*' '*	*Рядок не задовольняє умові. У рядку немає чисел.*
3	*test_empty_string(self)*	*'\n'*	*Рядок не задовольняє умові. Рядок порожній*
4	*test_many_numbers(self)*	*“flwe6 df4”, “fe1 ff2”*	*Рядок не задовольняє умові. У рядку більше ніж одне число.*
5	*test_quantity_sym(self)*	*“flwe8”, “fe1wegwgweg”*	*Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку.*
6	*test_no_number(self)*	*“flwewefweg”, “fffffffff”*	*Рядок не задовольняє умові. У рядку немає чисел.*

- Тест-кейси ***test_normal_input(self)*** використовується для тестування правильної роботи програми із коректними даними.

- Тест-кейс ***test_empty_space(self)*** перевіряє роботу програми, коли користувач введе рядок, що містить тільки пробіли.
- Тест-кейс ***test_empty_string(self)*** призначений для перевірки роботи програми, коли користувач введе порожній рядок.
- Тест-кейс ***test_many_numbers(self)*** призначений для перевірки роботи програми, коли користувач введе рядок з декількома числами.
- Тест-кейс ***test_quantity_sym(self)*** призначений для перевірки роботи програми, коли користувач введе рядок з числом, яке не дорівнює кількості символів в рядку.
- Тест-кейс ***test_no_number(self)*** призначений для перевірки роботи програми, коли користувач введе рядок без чисел.

.

Програма 2

У таблиці 2 наведено назви тестових функцій, тестові дані та очікуваний результат програми 2.

№	*Тестова функція*	*Тестові дані*	*Результат*
1	*test_normal_input(self)*	*3,5,1,2,3,4,5*	*У списку чисел, які більше та менше ніж P, однакова к-сть.*
		1,5,1,2,3,4,5	*У списку чисел, які більше ніж P, більше.*
		5,5,1,2,3,4,5	*У списку чисел, які менше ніж P, більше.*
2	*test_bad_input_arg(self)*	*'\n\n', '3\nf\n', ' \n \n', '\n2\n'*	*Введено некоректні дані.*
3	*test_bad_input_numbers(self)*	*2,5,1,f 2,5,a*	*Ви ввели неправильне число.*

Таблиця 2

- Тест-кейси ***test_normal_input(self)*** використовується для тестування правильної роботи програми із коректними даними.
- Тест-кейс ***test_bad_input_arg(self)*** перевіряє роботу програми, коли аргументи для списку введено некоректно.
- Тест-кейс ***test_bad_input_numbers(self)*** перевіряє роботу програми, коли користувач введе букву або символ замість числа в елемент списку .

ВИСНОВКИ

На лабораторній роботі 2 ми ознайомилися з приймальним тестуванням програм на мові Python. Було написано специфікації для двох програм відповідно до завдань варіанту 28. Перша програма перевіряє рядок, якщо в ньому є лише одне число, яке дорівнює кількості символів в самому рядку, а друга – перевіряє яких чисел в списку більше, або яких більше ніж задане, або менше.

Перед розробкою програми було проведено аналіз та виявлено ситуації, які можуть завадити коректній роботі програми.

Після аналізу відбулася розробка програми згідно із специфікацією. Для кожної з них було створено тестовий модуль, щоб реалізувати тест-кейси. Відповідно до тестових даних, що були підготовлені заздалегідь, відбулася перевірка модуля.

ДОДАТКИ

Додаток 1

Текст програми 1

```
def task_1(string):
    num_list = []
    num=''
    for i in a:
        if i.isdigit():
            num = num + i
        else:
            if num != '':
                num_list.append(int(num))
                num = ''
    if num != '':
        num_list.append(int(num))
    if len(string) == 0:
        return "Рядок порожній."
    if len(num_list) == 1:
        if num_list[0] == len(a):
            return "Рядок задовольняє умові."
        else:
            return "Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку."
    elif len(num_list) == 0:
        return "Рядок не задовольняє умові. У рядку немає чисел."
    else:
        return "Рядок не задовольняє умові. У рядку більше ніж одне число."
if __name__ == '__main__':
    a = str(input("Введіть Ваш рядок: "))
    print(task_1(a))
```

Текст програми 2

```
import re
def task_2(p, n):
    arr = []
    p = float(p)
    n = float(n)
    n = int(n)
    print("Задавайте числа через Enter!")
    while len(arr) < n:
        elem = input("")
        res = re.match(r'-\d|\d', elem)
        if res == None:
            return "Ви ввели неправильне число."
        else:
            arr.append(float(elem))

    new_list = sorted(arr)
    b_p = 0
    l_p = 0
    for i in new_list:
        if i > p:
            b_p += 1
        elif i < p:
            l_p += 1
    if b_p > l_p:
        return "У списку чисел, які більше ніж P, більше."
    elif b_p < l_p:
        return "У списку чисел, які менше ніж P, більше."
    elif b_p == l_p:
        return "У списку чисел, які більше та менше ніж P, однакова к-сть."

if __name__ == "__main__":
    p = input("P: ")
    res1 = re.match(r'-\d|\d', p)
    n = input("N: ")
    res2 = re.match(r'\d|\d.0', n)
    if res1 == None or res2 == None:
        print("Введено некоректні дані.")
    else:
        print(task_2(p, n))
```

Додаток 2

Текст модуля, що реалізує автоматичне тестування програми 1

```
import locale
from subprocess import Popen, PIPE, TimeoutExpired
import unittest

class TestCalcAcceptance(unittest.TestCase):
    ENCODING = locale.getpreferredencoding()
    PROCESS_WITH_ARGS = ('python3', 'prog_1.py')
    PROCESS_TIMEOUT = 5
    INPUT_SIGNATURE = 'Введіть Ваш рядок: '
    MESSAGE_SIGNATURE_EMPTY_STRING = 'Рядок не задовольняє умові. Рядок порожній.'
    MESSAGE_SIGNATURE_QUANTITY_SYM = 'Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку.'
    MESSAGE_SIGNATURE_NO_NUM = 'Рядок не задовольняє умові. У рядку немає чисел.'
    MESSAGE_SIGNATURE_MANY_NUM = 'Рядок не задовольняє умові. У рядку більше ніж одне число.'
    def run_subprocess(self, input_value):
        try:
            proc = Popen(self.PROCESS_WITH_ARGS,
                         stdin=PIPE,
                         stdout=PIPE,
                         stderr=PIPE)
            out_value, err_value = proc.communicate(input_value.encode(self.ENCODING),
                                                    timeout=self.PROCESS_TIMEOUT)
        except TimeoutExpired:
            proc.kill()
            out_value, err_value = proc.communicate()
        return out_value.decode(self.ENCODING), err_value.decode(self.ENCODING)


    def test_normal_input(self):

        input_data = [
            ('flwee6', 'Рядок задовольняє умові.'),
            ('ffrtger8', 'Рядок задовольняє умові.'),
            ('dd dd dd9', 'Рядок задовольняє умові.'),
            ('ffa8 fff', 'Рядок задовольняє умові.'),
        ]
        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str)
            actual_result = output_str.strip().split(self.INPUT_SIGNATURE)[-1]
            self.assertEqual(actual_result, expect_str)

    def test_empty_space(self):
        input_str = '   '
        output_str, err_str = self.run_subprocess(input_str)
        self.assertIn(self.MESSAGE_SIGNATURE_NO_NUM, output_str)
```

```
    def test_empty_string(self):
        input_str = '\n'
        output_str, err_str = self.run_subprocess(input_str)
        self.assertIn(self.MESSAGE_SIGNATURE_EMPTY_STRING, output_str)

    def test_many_numbers(self):
        input_data = [
            ('flwe6 df4', 'Рядок не задовольняє умові. У рядку більше ніж одне число.'),
            ('fe1 ff2', 'Рядок не задовольняє умові. У рядку більше ніж одне число.'),
        ]
        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str)
            self.assertIn(self.MESSAGE_SIGNATURE_MANY_NUM, output_str)
    def test_quantity_sym(self):
        input_data = [
            ('flwe8', 'Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку.'),
            ('fe1wegwgweg', 'Рядок не задовольняє умові. Число не дорівнює к-сті символів в рядку.'),
        ]
        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str)
            self.assertIn(self.MESSAGE_SIGNATURE_QUANTITY_SYM, output_str)
    def test_no_number(self):
        input_data = [
            ('flwewefweg', 'Рядок не задовольняє умові. У рядку немає чисел.'),
            ('fffffffff', 'Рядок не задовольняє умові. У рядку немає чисел.'),
        ]
        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str)
            self.assertIn(self.MESSAGE_SIGNATURE_NO_NUM, output_str)

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

Текст модуля, що реалізує автоматичне тестування програми 2

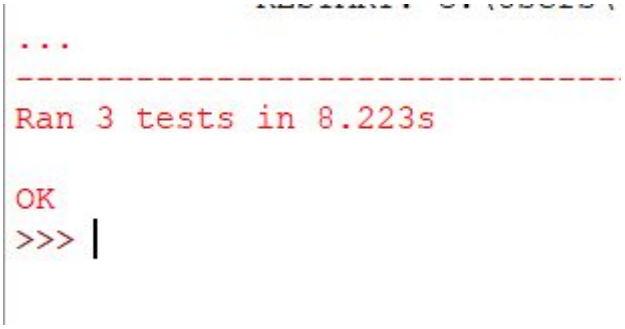
```
import locale
from subprocess import Popen, PIPE, TimeoutExpired
import unittest


class TestRideSideExchange(unittest.TestCase):
    ENCODING = locale.getpreferredencoding()
    PROCESS_WITH_ARGS = ('python3', 'prog_2.py')
    PROCESS_TIMEOUT = 5
    INPUT_SIGNATURE = 'P: N: Задавайте числа через Enter!'
    EXCEPTION_SIGNATURE = 'Введено некоректні дані.'
    MESSAGE = 'Ви ввели неправильне число.'

    def run_subprocess(self, input_value):
        try:
            proc = Popen(self.PROCESS_WITH_ARGS,
                         stdin=PIPE,
                         stdout=PIPE,
                         stderr=PIPE)
            out_value, err_value = proc.communicate(input_value.encode(self.ENCODING),
                                                    timeout=self.PROCESS_TIMEOUT)
        except TimeoutExpired:
            proc.kill()
            out_value, err_value = proc.communicate()
        return out_value.decode(self.ENCODING), err_value.decode(self.ENCODING)

    def test_normal_input(self):
        input_data = [
            ('3\n5\n1\n2\n3\n4\n5\n',  'У списку чисел, які більше та менше ніж P, однакова к-сть.'),
            ('1\n5\n1\n2\n3\n4\n5\n',  'У списку чисел, які більше ніж P, більше.'),
            ('5\n5\n1\n2\n3\n4\n5\n',  'У списку чисел, які менше ніж P, більше.'),
        ]

        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str)
            actual_result = output_str.strip().split(self.INPUT_SIGNATURE)[-1].strip()
            self.assertEqual(actual_result, expect_str)
```

```
    def test_bad_input_arg(self):
        bad_input_data = [
            '\n\n',
            '3\nf\n',
            ' \n \n',
            '\n2\n',
        ]

        for input_str in bad_input_data:
            output_str, error_str = self.run_subprocess(input_str)
            self.assertIn(self.EXCEPTION_SIGNATURE, output_str)
    def test_bad_input_numbers(self):
        bad_input_data = [
            '2\n5\n1\nf\n',
            '2\n5\na\n',
        ]

        for input_str in bad_input_data:
            output_str, error_str = self.run_subprocess(input_str)
            self.assertIn(self.MESSAGE, output_str)
if __name__ == '__main__':
    unittest.main()
```

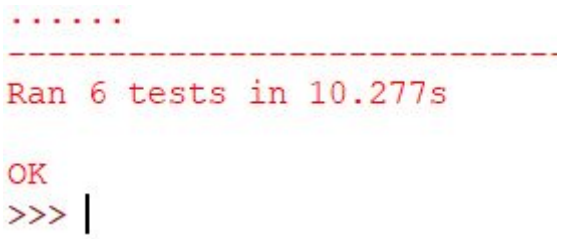
Додаток 3

Скріншоти результатів виконання автотестів

Програма 1



Програма 2



Додаток 4

Контрольні запитання

1. Що таке приймальне тестування (acceptance testing)?

Приймальне тестування (acceptance testing) – це тестування, яке проводиться, щоб перевірити, чи програмне забезпечення відповідає вимогам замовника.

2. Які цілі й переваги приймального тестування?

Серед цілей приймального тестування виділяють перевірку на відповідність вимогам замовника та пошук помилок, які пов’язані зі зручністю для користувачів, а серед переваг – виправлення недоліків, що заважають зручному користуванню даним програмним забезпеченням.

3. Яке місце приймального тестування в життєвому циклі розробки ПЗ?

Приймальне тестування виконується на фінальному етапі тестування перед публічним запуском програми.

4. До якого типу тестування відноситься приймальне тестування?

Приймальне тестування відносить до функціонального тестування.

5. На основі чого створюються тест-кейси для приймального тестування?

Тест-кейси створюються на основі специфікації програми, що тестується.

6. Які типові об’єкти приймального тестування?

До типових об’єктів приймального тестування відносять бізнес-процеси на повністю інтегрованій системі, процеси експлуатації та обслуговування, звіти.

7. Які види приймального тестування вам відомі?

До видів приймального тестування відносять альфа-тестування та бета-тестування.

8. Які є методи приймального тестування?

Існують такі методи приймального тестування: тестування замовником самостійно, спільне тестування постачальника програмного забезпечення та замовника, тестування незалежною (третьою) стороною.

9. Що таке регресійне тестування?

Регресійне тестування – вид тестування, який має на меті переконатися, що зміна коду в певній частині програми на буде мати негативного впливу на саму програму.

10. Які критерії завершення приймального тестування?

Для завершення приймального тестування потрібно, щоб програмне забезпечення, яке тестується, відповідало заданим вимогам.