

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-82

Бубела Д. В.

Керівник:

асистент

Громова В. В.

ЗМІСТ

Мета роботи	3
1 Постановка задачі	4
2 Основна частина.....	5
3 Розробка тест-кейсів	6
Висновки	9
Перелік посилань	10
Додаток А Тексти програм, які тестуються.....	11
Додаток Б Тексти модулів, що реалізують автоматичне тестування	12
Додаток В Скріншоти результатів виконання автотестів	14
Додаток Г Відповіді на контрольні запитання	15

МЕТА РОБОТИ

Ознайомитися з бібліотекою subprocess зі стандартної бібліотеки Python. Провести автоматизоване приймальне тестування (acceptance testing) програм на мові Python складеними за індивідуальним завданням вказаними в «Загальних інструкціях до лабораторних робіт» відповідно до номеру.

Завдання 1

2) Замінити закінчення (останні два символи) на 'xz' у словах, довжина яких дорівнює 5.

Завдання 2

2) Дано одновимірний масив числових значень, що нараховує n елементів. Виконати переміщення елементів масиву по колу вправо, тобто $a[1] \rightarrow a[2]$; $a[2] \rightarrow a[3]$; ... $a[n] \rightarrow a[1]$.

1 ПОСТАНОВКА ЗАДАЧІ

Програми з двох завдань повинні бути протестовані на коректність виводу на попередньо підготовлений ввід. На відміну від модульного тестування, в приймальному — програма яка тестується запускається як окрема одиниця і процес тестування проходить за тими ж потоками вводу/виводу, що й з користувачем (стандартними).

Завдання 1

Програма повинна зчитувати набір символів до початку нового рядка. Також їй необхідно рахувати довжину слова, якщо вона дорівнює 5ти — потрібно замість 2х останніх букв поставити “xz”. Зауваження: замінюватись мають саме букви, вони й підраховуються в слові. Тобто апостроф в слові не рахується, якщо він буде стояти між 2ма останніми буквами — замінити треба саме букви. Слово в лапках рахується як слово, без лапок, всі заміни робляться в слові. В стандартний потік виводу має направитись рядок зі зміненими символами.

Завдання 2

Програма повинна зчитувати набір символів до початку нового рядка. В зчитаному рядку мають знаходитись елементи масиву розділені пробілами. Пусті елементи (два пробіла підряд) не рахуються окремими елементами масиву. Усі елементи в масиві циклічно зсовуються вперед на одну одиницю. В стандартний потік виводу має вивестись введений масив з розділенням елементів через пробіл.

2 ОСНОВНА ЧАСТИНА

Програми будуть тестуватися в прийнятному режимі. Програма, що тестується, обмінюватиметься даними зі стандартним потоком вводу, виводу, помилок, якими керує програма, що тестує.

Завдання 1

Для вибору слів використовується регулярний вираз. Всі слова, що відповідають цьому виразу замінюються результатом роботи лямбди, яка приймає й видає окреме слово. В цій функції, коли кількість букв у слові дорівнює 5 здійснюється заміна 2х останніх букв регулярним виразом. Саме заміну здійснює функція `sub` з бібліотеки `re`.

Завдання 2

Зчитаний рядок розділюється на окремі елементи функцією `split` по пробілам. Для циклічного зсуву на 1 елемент конкатенуються 2 слайси списку, які є елементами з кінця та з початку до вибраних елементів з кінця. Елементи списку виводяться без додаткових обрамлень розділені пробілом.

Тестування

Програми що тестують завдання подібні та мають спільні елементи. Використовується стандартна бібліотека `unittest`. Головний клас, який відповідає за тестування є нащадком `unittest.TestCase`. Для запуску підпроцесів використовується стандартна бібліотека `subprocess`, `Popen` для відкриття програм, `PIPE` для роботи з вхідними/вихідними потоками, `TimeoutExpired` — виключення, що виникає, коли програма довго не відповідає. Кодування консолі, вводу/виводу синхронізується завдяки бібліотеці `locale`.

3 РОЗРОБКА ТЕСТ-КЕЙСІВ

Тест кейси для виняткових ситуацій зробити неможливо, адже програма приймає будь-який текстовий ввід і видає результат.

Таблиця 3.1 – Тест-кейси завдання 1

Тест	Мета	Вхідні данні	Очікуваний результат
1	Перевірити заміну в простому слові не на початку і не в кінці речення	Practice makes perfect	Practice makxz perfect
2	Перевірити заміну на >1 слові	A sound mind in a sound body	A souxz mind in a souxz body
3	Чи програма рахує розділення слів правильно? Апостроф має бути частиною слова	o'reill contains of six words and an apostrophe	o'reill contains of six worxz and an apostrophe
4	Заміна не в латинських символах	П'ятий човен йшов на дно	П'ятххз човххз йшов на дно

Продовження таблиці 3.1

Тест	Мета	Вхідні данні	Очікуваний результат
5	Перевірити чи рахує 5 апострофів за слово з 5ма символами	"""" "" "" "" ""	"""" "" "" "" ""
6	Як реагує на пробіли?	9 пробілів	9 пробілів?
7	Чи працює на пустих рядках?		
8	Перевірити заміну в реченнях, слів після яких іде кома	here are many ways to sort them - by suits (diamonds, clubs, hearts, and spades) or by numbers	here are many ways to sort them - by suixz (diamonds, cluxz, hearts, and spades) or by numbers

Таблиця 3.2 – Тест-кейси завдання 2

Тест	Мета	Вхідні данні	Очікуваний результат
1	Перевірити зсув в простому реченні	Practice makes perfect	perfect Practice makes
2	Перевірити заміну на цифрах	1 2 3 4 5	5 1 2 3 4

Продовження таблиці 3.2

Тест	Мета	Вхідні данні	Очікуваний результат
3	Чи програма рахує розділення слів правильно? Апостроф має бути частиною слова	П'ятий човен на дно йшов	йшов П'ятий човен на дно
4	Перевірити чи рахує апострофи	"""" "" "" ""	"" """" "" ""
5	Як реагує на пробіли?	9 пробілів	Пустий рядок
6	Чи працює на пустих рядках?		
7	Перевірити зсув в реченнях, слів після яких іде кома	here ar3e many wa“ys to sort them - by suits (diamonds, clubs, hearts, and spades) or by numbers	numbers here ar3e many wa“ys to sort them - by suits (diamonds, clubs, hearts, and spades) or by

ВИСНОВКИ

Ми ознайомилися з процесом приймального тестування з самого початку і до кінця. Воно відбувається на фінальній стадії розробки програмного продукту, адже воно найбільше схоже з діями користувача. Для приймального тестування теж можна використовувати бібліотеку `unittest`, бо вона дає інтерфейс самого тестування. Необхідна бібліотека для виклику окремої підпрограми, так як перевіряється її робота в звичних для неї умовах, тести не вбудовуються в структуру самого програмного засобу, що тестується. `Subprocess` забезпечує необхідні вимоги вказані вище та також має функції для роботи з потоками вводу, виводу, помилок програми, може ловити виняткові ситуації. Програми були успішно протестовані на коректність роботи.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Програмування на мові PYTHON: інструкції до виконання лабораторних робіт з дисципліни «Програмування-1. Основи програмування» / КПІ ім. Ігоря Сікорського; укладач В. В. Громова.
2. Python Software Foundation. subprocess — Subprocess management [Електронний ресурс] / Python Software Foundation.. — 2020. — Режим доступу до ресурсу: <https://docs.python.org/3/library/subprocess.html>.
3. Python Software Foundation. unittest — Unit testing framework [Електронний ресурс] / Python Software Foundation.. — 2020. — Режим доступу до ресурсу: <https://docs.python.org/3/library/unittest.html>.

Додаток А

Тексти програм, які тестуються

Лістинг файлу task_1.py

```
import re

print("В усіх словах, що складаються з 5ти букв 2 останні буде замінено на \"xz\". Введіть рядок...")
s = input()
re_words = re.compile(r"([A-r']|^)([A-r]+'?[A-r]*)((\.\.,\!\\?]?)"

s = re_words.sub(
    lambda match:
        match[1] + re.sub(r"(.*)([A-r]{1})('?)([A-r]{1})('|$)", r"\1x\3z\5", match[2]) + match[3]
        if
            len(match[2]) - match[2].count("'") == 5
        else
            match[0],
        s,
)
print(s)
```

Лістинг файлу task_1.py

```
arr = input("Введіть елементи масиву через пробіл\n").split()
arr = arr[-1:] + arr[:-1]
print(*arr)
```

Додаток Б

Тексти модулів, що реалізують автоматичне тестування

Лістинг файлу task_1_acceptance_test.py

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

class TestTask1(unittest.TestCase):
    """Acceptance test task_1"""
    SCRIPT_NAME = "task_1.py"
    INPUT_SIGNATURE = "В усіх словах, що складаються з 5ти \"\
    \"букв 2 останні буде замінено на \"xz\". Введіть рядок...\n"
    PROCESS_TIMEOUT = 5
    ENCODING = locale.getpreferredencoding()

    def run_subprocess(self, input_value):
        """Run subprocess for testing"""
        try:
            proc = Popen(["python", self.SCRIPT_NAME],
                          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_valid_input(self):
        input_data = (
            # plain
            ("Practice makes perfect", "Practice makxz perfect"),
            # repeat
            ("A sound mind in a sound body", "A souxz mind in a souxz body"),
            # apostrophe
            ("o'reill contains of six words and an apostrophe", "o'reill contains of six worxz and an apostrophe"),
            # another language with '
            ("П'ятий човен йшов на дно", "П'ятхз човхз йшов на дно"),
            # just apostrophes
            ("'''''' '''''' '' ''''", "'''''' '''''' '' ''''"),
            # spaces
            ("      ", "      "),
            # blank
            (" ", " "),
            # plain sentence, comma
            ("here are many ways to sort them - by suits (diamonds, clubs, hearts, and spades) or by numbers",
             "here are many ways to sort them - by suixz (diamonds, cluxz, hearts, and spades) or by numbers"),
        )

        for input_str, expect_str in input_data:
            output_str, error_str = self.run_subprocess(input_str + '\n')
            actual_result = output_str.replace(self.INPUT_SIGNATURE, '').replace("\n", '')
            self.assertEqual(actual_result, expect_str)

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

Лістинг файлу task_2_acceptance_test.py

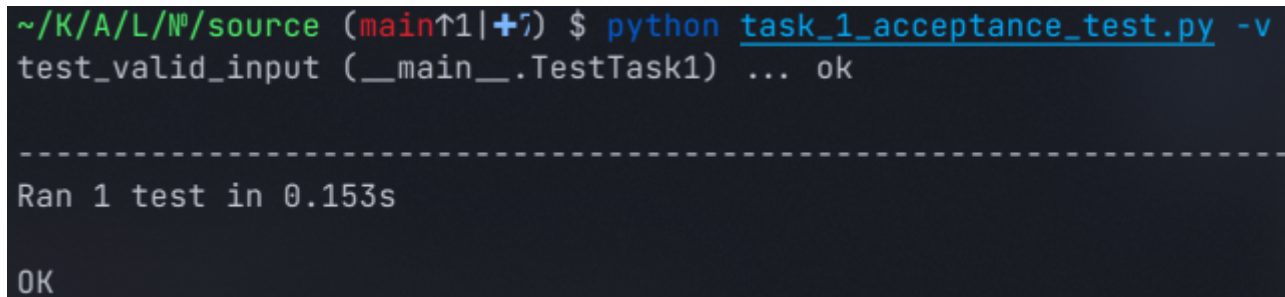
```
import locale
from subprocess import Popen, PIPE, TimeoutExpired
import unittest

class TestTask1(unittest.TestCase):
    """Acceptance test task_1"""
    SCRIPT_NAME = "task_2.py"
    INPUT_SIGNATURE = "Введіть елементи масиву через пробіл\n"
    PROCESS_TIMEOUT = 5
    ENCODING = locale.getpreferredencoding()

    def run_subprocess(self, input_value):
        """Run subprocess for testing"""
        try:
            proc = Popen(["python", self.SCRIPT_NAME],
                          stdin=PIPE,
                          stdout=PIPE,
                          stderr=PIPE)
```


Додаток В

Скріншоти результатів виконання автотестів

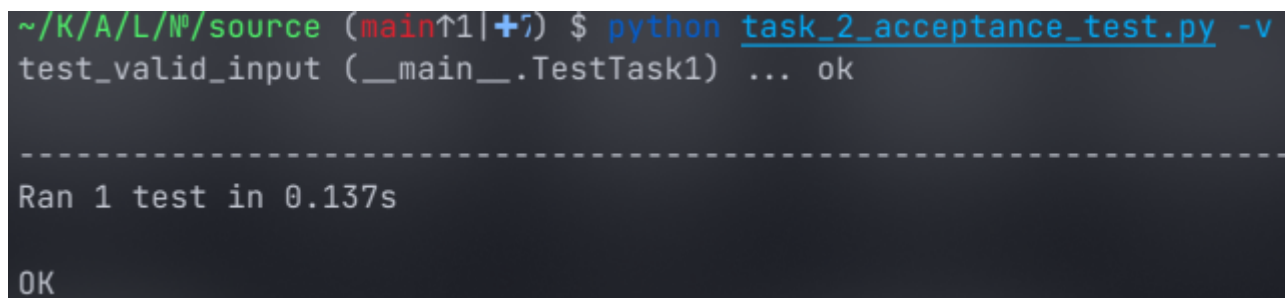


```
~/K/A/L/№/source (main↑1|+7) $ python task_1_acceptance_test.py -v
test_valid_input (__main__.TestTask1) ... ok

-----
Ran 1 test in 0.153s

OK
```

Рисунок В.1 – Тестування програми 1



```
~/K/A/L/№/source (main↑1|+7) $ python task_2_acceptance_test.py -v
test_valid_input (__main__.TestTask1) ... ok

-----
Ran 1 test in 0.137s

OK
```

Рисунок В.2 – Тестування програми 2

Додаток Г

Відповіді на контрольні запитання

1) Що таке приймальне тестування (acceptance testing)?

Приймальне тестування — це тестування ПЗ яке проводять для перевірки на відповідність до вимог замовника. Також це фінальний етап тестування програми перед публічним запуском

2) Які цілі й переваги приймального тестування?

Цілі: приймальне тестування допомагає відшукати помилки, пов'язані зі зручністю та простотою програми для користувачів; Даний вид тестування є фінальним етапом перед запуском програми; Приймальне тестування здійснюється за допомогою реального сприйняття додатку кінцевими користувачами.

Переваги: Приймальне тестування допомагає відшукати баги пов'язані зі зручністю та простотою програми для користувачів.

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

Даний вид тестування є фінальним етапом в життєвому циклі розробки ПЗ.

4) До якого типу тестування відноситься приймальне тестування?

Приймальне тестування відноситься до функціонального тестування.

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

Тест-кейси для приймального тестування створюються на основі вимог до програмного забезпечення перелічені в специфікації.

6) Які типові об'єкти приймального тестування?

- Процедури використання
- Бізнес-процеси на повністю інтегрованій системі

- Процеси експлуатації та обслуговування

- Звіти

7) Які види приймального тестування вам відомі?

- Тестування прийнятності для користувача

- Оперативні приймальні випробування (ОПВ)

- Контракт і регулювання приймального тестування

- Альфа і бета-тестування

8) Які є методи приймального тестування?

- Тестування замовником самостійно

- Тестування третьою стороною

- Спільне тестування з замовником

9) Що таке регресійне тестування?

Регресійне тестування — у більшості випадків, розглядають, як тестування, яке має на меті переконатися, що нова зміна коду в певному місці програми не вплинула негативно на інші доти працюючі частини програми. Регресійними можуть бути як функціональні, так і нефункціональні тести.

10) Які критерії завершення приймального тестування?

Приймальне тестування завершується при створенні всіх тестових випадків передбачених специфікацією та успішному їх проходженню.