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

|  |  |
| --- | --- |
| Виконала: | Керівник: |
| студентка групи КМ-81 | *асистент Громова В. В.* |
| *Приставська М.М* |  |

Київ — 2020

**Зміст**

[Мета роботи 2](#_Toc51858891)

[1. Постановка завдання 3](#_Toc51858892)

[2. Основна частина 4](#_Toc51858893)

[3. Розробка тест-кейсів 5](#_Toc51858894)

[Висновки 6](#_Toc51858895)

[Список літератури 7](#_Toc51858896)

[Додаток 1. Відповіді на контрольні питання 8](#_Toc51858897)

[Додаток 2. Текст програми 1, що тестується 9](#_Toc51858898)

[Додаток 3. Текст модуля, що реалізує автоматичне тестування програми 1 10](#_Toc51858899)

[Додаток 4. Текст програми 2, яка тсстується 11](#_Toc51858900)

[Додаток 5. Текст модуля, що реалізовує автоматичне тестування програми 2 12](#_Toc51858901)

[Додаток 6. Скріншоти результатів виконання тестів 13](#_Toc51858902)

# Мета роботи

Завдання 1

Доповнити символом '\*' слова, що мають довжину менше заданої (максимальної) до максимальної.

Завдання 2

Дано одновимірний масив числових значень, що нараховує n елементів. Суму елементів масиву і кількість додатних елементів поставити на перше і друге місце.

# Постановка завдання

Завдання 1

Программа повинна приймати з консолі рядок слів та числове значення максимальної довжини. Аналізувати довжину кожного слова, якщо воно коротше заданого, доповнювати символами зірочки. Повертати нове речення.

Завдання 2

Програма повинна приймати з консолі масив. Далі шукати суму елементів масиву та кількість додатних. Ставити значення на перше та друге місце в масиві відповідно. Повертати новий масив.

# Основна частина

Функціонал розроблюваних програм буде відповідати вимогам, що визначені при постановці завдання.

Для виконання приймального тестування скористуємося модулями unittest та subprocess. Метод run\_subprocess викликатиме цільову програму, передаватиме їй вхідні дані через потік введення, та отримуватиме результат і можливі помилки через потоки виведення та помилок відповідно.Розробка тест-кейсів

Тест-кейси для першої програми:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Мета | Вхідні дані | Очікуваний результат |
| 1 | Перевірити роботу програми при коректних значеннях. | testing is wonderful\n5\n | testing is\*\*\* wonderful |
| 2 | Перевірити роботу програми при відсутності введення першого значення | \n4\n | Повідомлення про помилку |
| 3 | Перевірити роботу програми при коректних значеннях у випадку введення літери замість числа | testing is not wonderful\nt\n | Повідомлення про помилку |
| 4 | Перевірити роботу програми при коректних значеннях у випадку введення нецілого числа | testing is not wonderful\n1.3\n | Повідомлення про помилку |
| 5 | Перевірити роботу програми при відсутності введення другого значення | testing is not wonderful\n\n | Повідомлення про помилку |

Тест-кейси для другої програми:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Мета | Вхідні дані | Очікуваний результат |
| 1 | Перевірити роботу програми при коректних значеннях | 2 6 7 -4 3 -1 2.2 | [15.2, 5, 2.0, 6.0, 7.0, -4.0, 3.0, -1.0, 2.2] |
| 2 | Перевірити роботу програми при відсутності значень | [\n] | Повідомлення про помилку |
| 3 | Перевірити роботу програми при некоректних значеннях | [3.2, 2, 's'] | Повідомлення про помику |

# Висновки

В ході виконання лабораторної роботи було розроблено специфікацію до двох програм, було створено програми, розроблено тест-кейси для тестування даних програм. В результаті тестування виявлено, що розроблені програми повністю відповідають специфікаціям та працюють коректно.

# Список літератури

1. Майерс Г. Искусство тестирования программ / Пер. с англ. под ред. Б. А. Позина – М.: Финансы и статистика, 1982. – 176 с.
2. Дідковська М.В. Технологія розробки та тестування програм: Навчальний посібник / М.В. Дідковська. – К.: «Вік-принт», 2014. – 344 с.
3. Lewis W. E. Software testing and continuous quality improvement / W. E. Lewis – 3rd ed. – CRC Press, 2009. – 640 p.
4. Patton R. Software testing / R. Patton. – Indianapolis: Sams, 2005. – 408 p.

# Додаток 1. Відповіді на контрольні питання

1. Що таке приймальне тестування (acceptance testing)?

Це фінальний етап тестування програми перед публічним запуском, коли тестуються всі модулі одночасно, у вигляді робочої програми. Приймальне тестування здійснюється за допомогою реального сприйняття додатку кінцевими користувачами.

1. Які цілі й переваги приймального тестування?

Цілі:

* + Приймальне тестування допомагає тестувати програму на рівні, близькому кінцевому користувачу;
  + Даний вид тестування є фінальним етапом перед запуском програми;
  + Приймальне тестування здійснюється за допомогою реального

сприйняття додатку кінцевими користувачами.

Переваги тестування цього виду тестування:

* + Перевірка роботи усіх модулів одночасно, у вигляді готового до релізу

продукту

* + Виконання тестів незалежними від розробки спеціалістами, це дозволяє

знаходити проблеми, які не розглядалися розробниками на ранніх етапах

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

Даний вид тестування є фінальним етапом розробки ПЗ, він так називається тому, що після нього продукт приймається замовниками.

1. До якого типу тестування відноситься приймальне тестування?

До функціонального тестування, методами “black box” або “white box”.

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

На основі процедур використання, що моделюють бізнес-процеси на повністю інтегрованій системі, на основі досвіду експлуатації та обслуговування.

1. Які типові об’єкти приймального тестування?
   * Процедури використання
   * Бізнес-процеси на повністю інтегрованій системі
   * Процеси експлуатації та обслуговування
   * Звіти
2. Які види приймального тестування вам відомі?
   * Тестування прийнятності для користувача
   * Оперативні приймальні випробування
   * Контракт і регулювання приймального тестування
   * Альфа- та бета-тестування
3. Які є методи приймального тестування?
   * Тестування замовником самостійно
   * Тестування третьою стороною
   * Спільне тестування з замовником
4. Що таке регресійне тестування?

Це загальна назва для всіх видів приймального тестування, спрямованих на виявлення помилок у вже протестованих ділянках початкового коду, що можуть виникнути при внесенні нових змін (розширенні функціоналу або виправленні інших помилок тощо). Одна з головних цілей регресійного тестування - це визначити, чи впливає зміна в одній частині програмного забезпечення на його інші частини.

10.Які критерії завершення приймального тестування?

• Час  
• Бюджет  
• Проходження тестів, виправлення знайдених помилок то повторна їх

перевірка

# Додаток 2. Текст програми 1, що тестується

def prog\_1(str, max\_l):

mas = str.split()

max\_len = int(max\_l)

res = []

for el in mas:

el\_len = len(el)

if el\_len < max\_len:

el = el + '\*' \* (max\_len - el\_len)

res.append(el)

res = ' '.join(res)

return res

if \_\_name\_\_ == '\_\_main\_\_':

sent = input('Enter data:')

if len(sent) == 0:

print("No data.")

max\_l = input('Enter max length:')

print(prog\_1(sent, max\_l))

input()

# Додаток 3. Текст модуля, що реалізує автоматичне

# тестування програми 1

from subprocess import Popen, PIPE, TimeoutExpired

import unittest

class Test\_prog\_1(unittest.TestCase):

ENCODING = 'utf-8'

PROCESS\_WITH\_ARGS = ('python', 'prog\_1.py')

PROCESS\_TIMEOUT = 5 # seconds

INPUT\_SIGNATURE = 'Enter data:Enter max length:'

EXCEPTION\_SIGNATURE = 'ValueError:'

MESSAGE\_SIGNATURE\_LACK\_OF\_WORDS = 'No data.'

def run\_subprocess(self, input\_value):

global proc

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 = [

('testing is wonderful\n5\n', 'testing is\*\*\* wonderful'),

('testing is my favorite pastime\n9\n',

'testing\*\* is\*\*\*\*\*\*\* my\*\*\*\*\*\*\* favorite\* pastime\*\*')

]

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\_invalid\_input(self):

bad\_input\_data = [

'testing is not wonderful\nt\n',

'testing is not wonderful\n\n',

'testing is not wonderful\n1.3\n'

]

for input\_str in bad\_input\_data:

output\_str, error\_str = self.run\_subprocess(input\_str)

self.assertIn(self.EXCEPTION\_SIGNATURE, error\_str)

def test\_lack\_of\_words(self):

input\_str = '\n4\n'

output\_str, err\_str = self.run\_subprocess(input\_str)

self.assertIn(self.MESSAGE\_SIGNATURE\_LACK\_OF\_WORDS, output\_str)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

# Додаток 4. Текст програми 2, яка тсстується

def prog\_2(l):

len\_list = len(numeric\_array)

if len\_list == 0:

return 0

sum\_of\_el = 0

number\_of\_positive\_el = 0

for el in numeric\_array:

sum\_of\_el += el

if el>0:

number\_of\_positive\_el += 1

numeric\_array.insert(0, sum\_of\_el)

numeric\_array.insert(1, number\_of\_positive\_el)

return numeric\_array

if \_\_name\_\_ == '\_\_main\_\_':

num\_l = input('Enter data:').split()

if len(num\_l) == 0:

print("No data.")

else:

try:

numeric\_array = list(map(float, num\_l))

except:

print("Invalid data.")

else: print(prog\_2(num\_l))

input()

# Додаток 5. Текст модуля, що реалізовує автоматичне

# тестування програми 2

from subprocess import Popen, PIPE, TimeoutExpired

import unittest

class Test\_prog\_2(unittest.TestCase):

ENCODING = 'utf-8'

PROCESS\_WITH\_ARGS = ('python', 'prog\_2.py')

PROCESS\_TIMEOUT = 5 # seconds

INPUT\_SIGNATURE = 'Enter data:'

# EXCEPTION\_SIGNATURE = 'ValueError:'

MESSAGE\_SIGNATURE\_LACK\_OF\_NUM = 'No data.'

MESSAGE\_SIGNATURE\_INCORRECT\_CHARACTERS = 'Invalid data.'

def run\_subprocess(self, input\_value):

global proc

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 = [

('2 6 7 -4 3 -1 2.2', '[15.2, 5, 2.0, 6.0, 7.0, -4.0, 3.0, -1.0, 2.2]')

]

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\_lack\_of\_num(self):

input\_str = '\n'

output\_str, err\_str = self.run\_subprocess(input\_str)

self.assertIn(self.MESSAGE\_SIGNATURE\_LACK\_OF\_NUM, output\_str)

def test\_incorrect\_characters(self):

bad\_input\_data = [

'1 е 3\nt\n',

]

for input\_str in bad\_input\_data:

output\_str, error\_str = self.run\_subprocess(input\_str)

self.assertIn(self.MESSAGE\_SIGNATURE\_INCORRECT\_CHARACTERS, output\_str)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

# Додаток 6. Скріншоти результатів виконання тестів

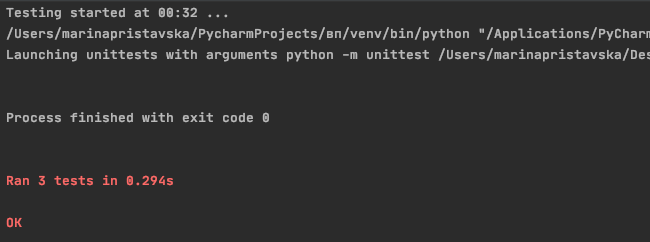


Рис. 1 – Результат виконання тестів першої програми

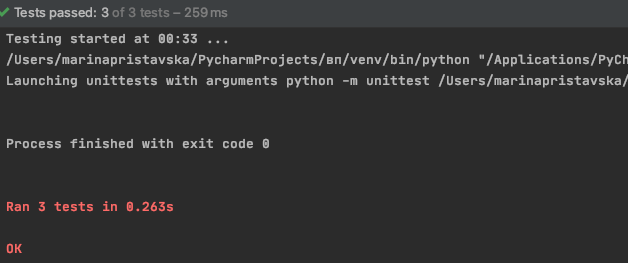


Рис. 2 – Результат виконання тестів другої програми