Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО - КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины «Программирование на Python»

Вариант 7

Выполнил:
Зармухамбетов Булат Эльдарович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023

Тема: Работа со строками в языке Python

Цель работы: приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Пример 1. Дано предложение. Все пробелы в нем заменить символом «_».

Листинг к примеру №1:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    s = input('Введите предложение: ')
    r = s.replace(' ', '_')
    print(f'Предложение после замены: {r}')
```

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\
Введите предложение: Меня зовут Олег
Предложение после замены: Меня_зовут_Олег

Process finished with exit code 0
```

Рисунок 1. 1-ый тест к примеру №1

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\Scri
Введите предложение: Как ваше    настроение   ?
Предложение после замены: Как_ваше____настроение___?

Process finished with exit code 0
```

Рисунок 2. 2-ой тест к примеру №1

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\S
Введите предложение:        слева7пробелов
Предложение после замены: _______слева7пробелов

Process finished with exit code 0
```

Рисунок 3. 3-ий тест к примеру №1

Пример 2. Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.

Листинг к примеру №2:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if   name   == '  main  ':
```

```
word = input('Введите слово: ')

idx = len(word) // 2
if len(word) % 2 == 1:
    # Длина слова нечётная.
    r = word[:idx] + word[idx+1:]
else:
    # Длина слова чётная.
    r = word[:idx-1] + word[idx+1:]

print(r)
```

```
C:\Users\User\PycharmProjects\Python_laba_6\
Введите слово: четыре
чере

Process finished with exit code 0
```

Рисунок 4. 1-ый тест к примеру №2

```
C:\Users\User\PycharmProjects\Python_laba_6\
Введите слово: бом
бм

Process finished with exit code 0
```

Рисунок 5. 2-ой тест к примеру №2

```
C:\Users\User\PycharmProjects\Python_laba_6\
Введите слово: АбВг
Аг

Process finished with exit code 0
```

Рисунок 6. 3-ий тест к примеру №2

Пример 3. Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки стала равна заданной длине (предполагается, что требуемая длина не меньше исходной). Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными словами должно отличаться не более чем на 1.

Листинг к примеру №3:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
```

```
if __name__ == '__main__':
    s = input('Введите предложение: ')
    n = int(input('Введите длину: '))

    # Проверить требуюмую длину.
    if len(s) >= n:
        print('Заданная длина должна быть больше длины предложения',
file=sys.stderr)
        exit(1)

    # Разделить предложение на слова.
    words = s.split(' ')
    # Проверить количество слов в предложении.
    if len(words) < 2:
        print('Предложение должно содержать несколько слов', file=sys.stderr)
        exit(1)

    # Количество пробелов для добавления.
    delta = n
    for word in words:
        delta -= len(word)

    # Количество пробелов на каждое слово.
    w, r = delta // (len(words) - 1), delta % (len(words) -1)

    # Сформировать список для хранения пробелов.
    lst = []

    # Пронумеровать все слова в списке и перебрать их.
    for i, word in enumerate(words):
        lst.append(word)

        # Если слово не является последним, добавить пробелы.
        if i < len(words) - 1:
            # Определить количество пробелов.
            width = w
            if r > 0:
                width += 1
                r -= 1

            # Добавить заданное количество пробелов в список.
            if width > 0:
                lst.append(' ' * width)

    # Вывести новое предложение, объединив все элементы списка lst.
    print(''.join(lst))
```

```
C:\Users\User\PycharmProjects\Python_laba_6\
Введите предложение: Привет, Мир!
Введите длину: 15
Привет,     Мир!

Process finished with exit code 0
```

Рисунок 7. 1-ый тест к примеру №3

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\Scr
Введите предложение: Приятно познакомиться
Введите длину: 40
Приятно                    познакомиться

Process finished with exit code 0
```

Рисунок 8. 2-ой тест к примеру №3

```
C:\Users\User\PycharmProjects\Python_laba_6\ven
Введите предложение: Как Вас зовут?
Введите длину: 23
Как      Вас     зовут?

Process finished with exit code 0
```

Рисунок 9. 3-ий тест к примеру №3

Индивидуальное задание 1. Дано предложение. Напечатать все его буквы “и”.

Листинг к индивидуальному заданию №1:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    counter = 0
    sentence = input('Enter a sentence: ')

    for letter in sentence:

        if letter.lower() == 'и':
            # Добавил счётчик кол-ва букв "и" в строке: считаю это полезным.
            counter += 1
            print(letter)

    print(f'Number of letters "и": {counter}')
```

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\
Enter a sentence: Параллелепипед
и
Number of letters "и": 1
```

Рисунок 10. 1-ый тест к индивидуальному заданию №1

Рисунок 11. 2-ой тест к индивидуальному заданию №1

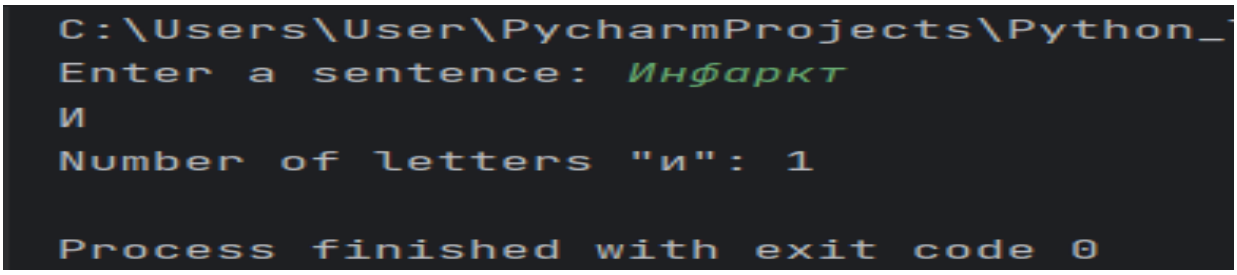

Рисунок 12. 3-ий тест к индивидуальному заданию №1

Индивидуальное задание 2. Дано предложение. Определить количество букв н, предшествующих первой запятой предложения. Рассмотреть два случая:

известно, что запятые в предложении есть;

запятых в предложении может не быть.

Листинг к индивидуальному заданию №2:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    sentence = input("Enter a sentence: ")
    counter = 0
    comma_index = sentence.find(",")

    if comma_index != -1:

        for letter in sentence[:comma_index]:
            if letter.lower() == "н":
                counter += 1
        print(f"The number of letters 'н' preceding the first comma:
{counter}")
    else:

        for letter in sentence:
            if letter.lower() == "н":
                counter += 1

        print('There are no commas!')

        print(f"Number of letters 'н': {counter}")
```

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\Scripts
Enter a sentence: Доброго, времени суток, пользователь!
The number of letters 'н' preceding the first comma: 0

Process finished with exit code 0
```

Рисунок 13. 1-ый тест к индивидуальному заданию №2

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\Scripts
Enter a sentence: Ннн, н - прекрасная буква!
The number of letters 'н' preceding the first comma: 3

Process finished with exit code 0
```

Рисунок 14. 2-ой тест к индивидуальному заданию №2

```
C:\Users\User\PycharmProjects\Python_laba_6\venv\S
Enter a sentence: Номинация года
There are no commas!
Number of letters 'н': 2

Process finished with exit code 0
```

Рисунок 15. 3-ий тест к индивидуальному заданию №2

Индивидуальное задание 3. Дано слово. Удалить из него все повторяющиеся буквы, оставив их первые вхождения, т. е. в слове должны остаться только различные буквы.

Листинг к индивидуальному заданию №3:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    word = input("Введите слово: ")

    result = ""

    for letter in word:
        if letter not in result:
            result += letter

    print(f"Result: {result}")
```

```
C:\Users\User\PycharmProjects\Python
Введите слово: параллелепипед
Result: парлеид

Process finished with exit code 0
```

Рисунок 16. 1-ый тест к индивидуальному заданию №3

```
C:\Users\User\PycharmProjects\Python_l
Введите слово: гиппопотам
Result: гипотам

Process finished with exit code 0
```

Рисунок 17. 2-ой тест к индивидуальному заданию №3

```
C:\Users\User\PycharmProjects\Python_laba_6\
Введите слово: Каддилак
Result: Кадилк

Process finished with exit code 0
```

Рисунок 18. 3-ий тест к индивидуальному заданию №3

Задание повышенной сложности. Даны два слова. Напечатать только те буквы слов, которые есть лишь в одном из них (в том числе повторяющиеся). Например, если заданные слова процессор и информация, то ответом должно быть: п е с с и н ф м а я.

Листинг к программе повышенной сложности:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == '__main__':
    word_1 = input('Enter the first sentence: ')
    word_2 = input('Enter the second sentence: ')

    unique_letters = ""
    result = ""

    for letter in word_1:
        if letter not in word_2:
            unique_letters += letter
            result += letter + ' '

    for letter in word_2:
        if letter not in word_1 and letter not in unique_letters:
            unique_letters += letter
            result += letter + ' '

    print(result)
```

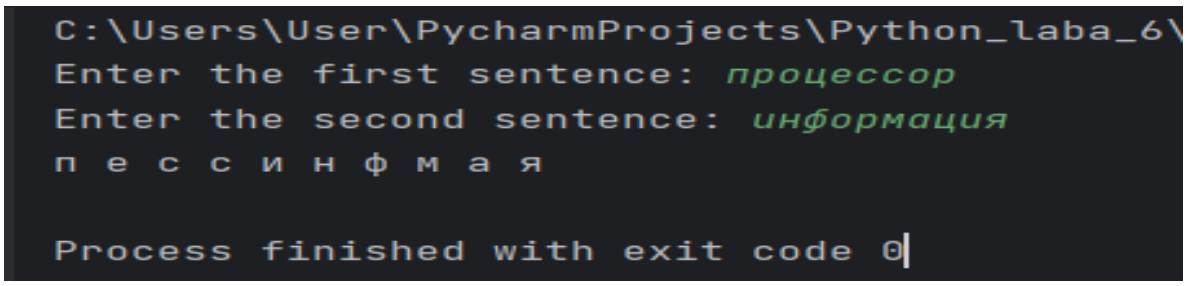

Рисунок 19. 1-ый тест к программе повышенной сложности

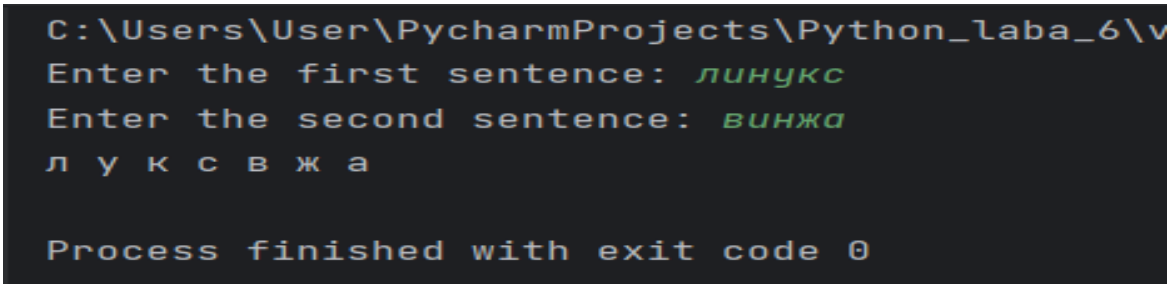

Рисунок 20. 2-ой тест к программе повышенной сложности

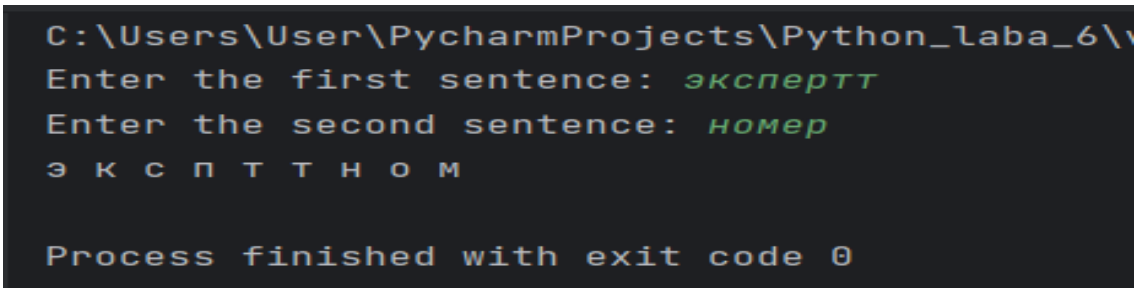

Рисунок 21. 3-ий тест к программе повышенной сложности

Вывод: в ходе выполнения данной лабораторной работы были приобретены навыки по взаимодействию со строками при написании программ с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы

1. Строки в языке Python - это упорядоченная последовательность символов, используемая для хранения и представления текстовой информации.

2. Существуют три способа задания строковых литералов в языке Python: с помощью одинарных кавычек ('), двойных кавычек (") и тройных кавычек (''' или """).

3. Для строк существует множество операций и функций, таких как конкатенация, повторение, извлечение символов по индексу, замена подстроки, разделение на подстроки и другие.

4. Индексирование строк осуществляется с помощью квадратных скобок и номера индекса символа, начиная с 0 для первого символа.

5. Работа со срезами для строк позволяет извлекать подстроки по заданным индексам или с использованием шага.

6. Строки Python относятся к неизменяемому типу данных, потому что после создания строки её содержимое нельзя изменить.

7. Для проверки того, что каждое слово в строке начинается с заглавной буквы, можно воспользоваться методом title() или написать функцию для этой проверки.

8. Для проверки строки на вхождение в неё другой строки можно воспользоваться оператором in или методом find().

9. Индекс первого вхождения подстроки в строку можно найти с помощью метода find() или index().

10. Количество символов в строке можно подсчитать с помощью функции len().

11. Количество раз, которое определённый символ встречается в строке, можно подсчитать с помощью метода count().

12. F-строки (форматированные строки) позволяют встраивать значения переменных и выражений в строки с помощью фигурных скобок и префикса f.

13. Подстроку в заданной части строки можно найти с помощью метода find() или index().

14. Содержимое переменной можно вставить в строку, воспользовавшись методом format().

15. Для проверки того, что в строке содержатся только цифры, можно воспользоваться методом isnumeric().

16. Строку можно разделить по заданному символу с помощью метода split().

17. Для проверки строки на то, что она составлена только из строчных букв, можно воспользоваться методом islower().

18. Для проверки того, что строка начинается со строчной буквы, можно воспользоваться методом islower() и индексированием.

19. В Python нельзя прибавить целое число к строке напрямую, не преобразовав его в строку.

20. Строку можно "перевернуть" с помощью срезов и шага -1.

21. Список строк можно объединить в одну строку с помощью метода join(), передав ему список строк.

22. Верхний и нижний регистр строки можно получить с помощью методов upper() и lower() соответственно.

23. Первый и последний символы строки можно преобразовать к верхнему регистру с помощью методов capitalize() и upper().

24. Для проверки строки на то, что она составлена только из прописных букв, можно воспользоваться методом isupper().

25. Метод splitlines() используется для разделения строки на подстроки по символам новой строки.

26. В заданной строке все вхождения некой подстроки можно заменить на что-либо с помощью метода replace().

27. Для проверки того, что строка начинается или заканчивается заданной последовательностью символов, можно использовать методы startswith() и endswith().

28. Для проверки того, что строка включает только пробелы, можно воспользоваться методом isspace().

29. Если умножить некую строку на 3, она будет повторена три раза.

30. Первый символ каждого слова в строке можно привести к верхнему регистру с помощью метода title().

31. Метод partition() используется для разделения строки по первому вхождению заданного разделителя.

32. Метод rfind() используется для поиска последнего вхождения подстроки в строку.