

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 №2.3**  
**дисциплины «Программирование на Python»**  
**Вариант \_\_\_\_**

Выполнил:  
Иващенко Олег Андреевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.02 «Информационные и  
вычислительные машины»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем»

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** «Работа со строками в языке Python»

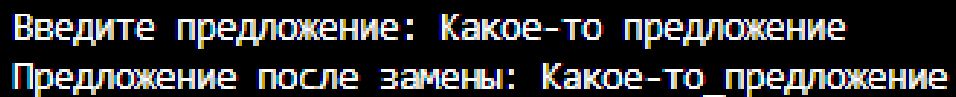
**Цель:** Приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

### Порядок выполнения работы

Таблица 1 – Код программы example\_1.py

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

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



Введите предложение: Какое-то предложение  
Предложение после замены: Какое-то\_предложение

Рисунок 1 – Вывод программы example\_1.py

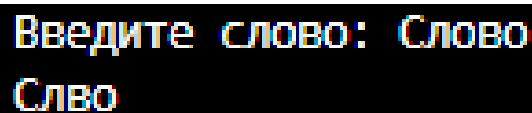
Таблица 2 – Код программы example\_2.py

```
#!/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 нечетная.
        r = word[:idx] + word[idx+1:]
    else:
        # Длина слова четная.
        r = word[:idx-1] + word[idx+1:]

    print(r)
```



Введите слово: Слово  
Слво

Рисунок 2.1 – Вывод программы example\_2.py при нечётном количестве  
СИМВОЛОВ

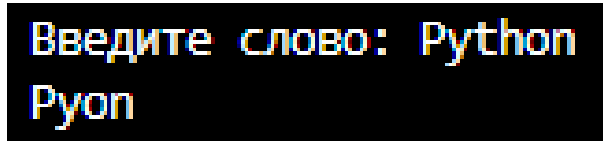


Рисунок 2.2 – Вывод программы example\_2.py при чётком количестве  
СИМВОЛОВ

Таблица 3 – Код программы example\_3.py

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

import sys

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

    # Проверить требуемую 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 слов и пробелов.
```

```

lst = []

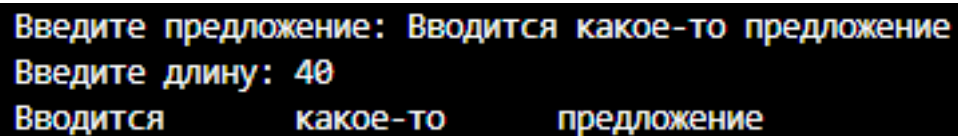
# Пронумеровать все слова в списке и перебрать их.
for i, word in enumerate(words):
    lst.append(word)

# 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))

```



```

Введите предложение: Вводится какое-то предложение
Введите длину: 40
Вводится      какое-то      предложение

```

Рисунок 3 – Вывод программы example\_3.py

Индивидуальное задание 1. Вывести «столбиком» все его буквы «и», стоящие на чётных местах.

Таблица 4 – Код программы individual\_1.py

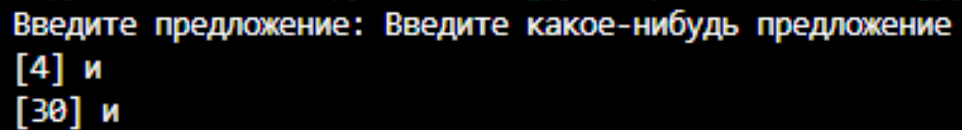
```

if __name__ == "__main__":
    sentence = input("Введите предложение: ")
    index = 0

    for i in sentence:
        if i == 'и' and index % 2 == 0:
            print(f"[{index}] {i}")

        index += 1

```



```
Введите предложение: Введите какое-нибудь предложение
[4] и
[30] и
```

Рисунок 4 – Вывод программы individual\_1.py

Индивидуальное задание 2. Дано предложение. Определить, какая из букв – «н» или «к» - встречается в нём раньше при просмотре слева направо (принять, что указанные буквы в строке есть).

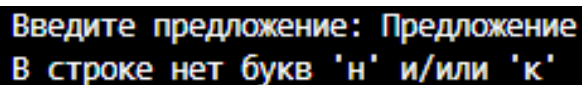
Таблица 5 – Код программы individual\_2.py

```
if __name__ == "__main__":
    sentence = input("Введите предложение: ")
    n_count = 0
    k_count = 0

    for i in sentence:
        if i == 'н' or i == 'Н':
            n_count += 1
        elif i == 'к' or i == 'К':
            k_count += 1

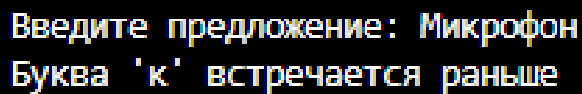
    if n_count == 0 or k_count == 0:
        print("В строке нет букв 'н' и/или 'к'")
        exit(1)

    for i in sentence:
        if i == 'н' or i == 'Н':
            print("Буква 'н' встречается раньше")
            break
        elif i == 'к' or i == 'К':
            print("Буква 'к' встречается раньше")
            break
```



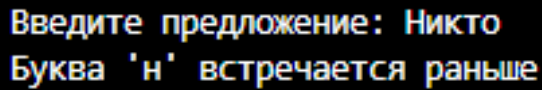
```
Введите предложение: Предложение
В строке нет букв 'н' и/или 'к'
```

Рисунок 5.1 – Вывод программы individual\_2.py в случае, если одной из букв «н» или «к» нет в слове



Введите предложение: Микрофон  
Буква 'к' встречается раньше

Рисунок 5.2 – Вывод программы individual\_2.py в случае, если буква «к» встречается раньше буквы «н»



Введите предложение: Никто  
Буква 'н' встречается раньше

Рисунок 5.3 – Вывод программы individual\_3.py в случае, если буква «н» встречается раньше буквы «к»

Индивидуальное задание 3. Дано ошибочно написанное слово «рпроцессо». Путём перемещения его букв получить слово процессор.

Таблица 6 – Код программы individual\_3.py

```
if __name__ == "__main__":  
    word = "рпроцессо"  
    word = word[1:] + word[0]  
  
    print(f"Исходное слово: {word}")
```



Исходное слово: процессор

Задание повышенной сложности. Дано предложение. Напечатать его в обратном порядке слов, например предложение «мама мыла раму» должно быть напечатано в виде «раму мыла мама».

Таблица 7 – Код программы inc\_def.py

```
if __name__ == "__main__":  
    sentence = input("Введите предложение: ")  
    words = sentence.split()  
    lst = []  
  
    for i in range(len(words) - 1, -1, -1):  
        lst.append(words[i])  
  
    sentence = ' '.join(lst)
```

```
print(sentence)
```

```
Введите предложение: Какое-то предложение  
предложение Какое-то
```

Рисунок 7 – Вывод программы inc\_def.py

### Контрольные вопросы

1. Что такое строки в языке Python?

В языке программирования Python строка – это последовательность символов, заключённых в кавычки. Они не могут быть изменены после создания, вместо них создаются новые строки.

2. Какие существуют способы задания строковых литералов в языке Python?

В языке Python существует несколько способов задания строковых литералов:

- Одинарные кавычки ('string');
- Двойные кавычки ("string");
- Тройные кавычки ('''string''' или """"string""").

3. Какие операции и функции существуют для строк?

Операции со строками:

- Конкатенация (+) – объединение строк;
- Умножение (\*) – повторение строки несколько раз;
- Индексация – обращение к символу строки по его индексу;
- Срезы – получение подстроки строки по заданным индексам.

Функции для строк:

- len() – возвращает длину строки;

- `lower()` – преобразует все символы строки в нижний регистр;
- `upper()` – преобразует все символы строки в верхний регистр;
- `strip()` – удаляет пробельные символы в начале и конце строки;
- `split()` – разбивает строку на список подстрок по заданному разделителю.

#### 4. Как осуществляется индексирование строк?

Индексирование строк в Python осуществляется с использованием квадратных скобок. Индексы начинаются с 0 для первого символа строки. Отрицательные индексы также поддерживаются, где -1 обозначает последний символ строки.

#### 5. Как осуществляется работа со срезами для строк?

Работа со срезами строк осуществляется с использованием квадратных скобок и двоеточия для указания диапазона индексов. Синтаксис выглядит так: `строка[начало:конец]`. Важно отметить, что конец диапазона не включается в срез.

#### 6. Почему строки Python относятся к неизменяемому типу данных?

Строки в Python относятся к неизменяемым типам данных по нескольким причинам:

- Безопасность – неизменяемость строк делает их более безопасными в использовании в различных контекстах, так как предотвращает случайные или нежелательные изменения внутри строки;
- Хеширование – неизменяемые объекты, такие как строки, могут быть использованы в качестве ключей в хеш-таблицах, таких как словари. Это обеспечивает эффективный поиск и доступ к данным;



- Оптимизация памяти – использование неизменяемых строк позволяет более эффективно управлять памятью. Например, если создать новую строку из существующей, Python может ссылаться на тот же участок памяти, что и исходная строка, если она не изменялась;
- Кэширование – неизменяемые объекты могут быть кэшированы, что улучшает производительность в некоторых сценариях, так как одни и те же строки с одинаковым содержимым будут использовать один и тот же объект в памяти;
- Однократное создание – когда строка создана, её значение не может быть изменено. Это способствует предсказуемости и упрощению работы с такими данными в программе.

7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

Для проверки того, что каждое слово в строке начинается с заглавной буквы, можно использовать метод строки `istitle()`. Этот метод возвращает `True`, если каждое слово в строке начинается с заглавной буквы, и `False` в противном случае.

8. Как проверить строку на вхождение в неё другой строки?

Для проверки вхождения одной строки в другую в Python, можно использовать оператор `in`. Этот оператор возвращает `True`, если подстрока присутствует в строке, и `False` в противном случае.

9. Как найти индекс первого вхождения подстроки в строку?

Для нахождения индекса первого вхождения подстроки в строку в Python, можно использовать метод строки `find()` или `index()`.

10. Как подсчитать количество символов в строке?

Для подсчёта количества символов в строке используется метод `len()`.

11. Как подсчитать то, сколько раз определённый символ встречается в строке?

Для подсчёта количества вхождений определённого символа в строке используется метод `count()`.

12. Что такое f-строки и как ими пользоваться?

f-строки (formatted string literals) представляют собой специальный синтаксис для форматирования строк, введенный в версии Python 3.6 и более поздних. Они позволяют встраивать значения переменных и выражений прямо в строку, делая код более читаемым и удобным.

Основные особенности f-строк:

- Префикс «f» - строки, начинающиеся с этого префикса, считаются f-строками;
- Выражения внутри фигурных скобок – значения переменных и выражений встраиваются внутрь фигурных скобок внутри строки;
- Форматирование значений – можно использовать дополнительные спецификации формата для управления стилем отображения значений;
- Многострочные f-строки – можно создавать многострочные f-строки с использованием тройных кавычек.

13. Как найти подстроку в заданной части строки?

Для поиска подстроки в заданной части строки в Python используются метод строки `find()` или оператор `in`.

14. Как вставить содержимое переменной в строку, воспользовавшись методом `format()`?

Для вставки содержимого переменной в строку с использованием метода `format()` в Python внутри строки указываются фигурными скобками-заполнители (placeholder), который будет заменён значением переменной, указанной внутри метода `format()`.

15. Как узнать о том, что в строке содержатся только цифры?

Для проверки того, что в строке содержатся только цифры, можно использовать метод строки `isdigit()`. Этот метод возвращает `True`, если все символы в строке являются цифрами, и `False` в противном случае.

16. Как разделить строку по заданному символу?

Для разделения строки по заданному символу в Python используется метод `split()`. Этот метод разбивает строку на список подстрок на основе указанного разделителя.

17. Как проверить строку на то, что она составлена только из строчных букв?

Для проверки того, что строка составлена только из строчных букв, можно использовать метод строки `islower()`. Этот метод возвращает `True`, если все буквы в строке являются строчными, и `False` в противном случае.

18. Как проверить то, что строка начинается со строчной буквы?

Для проверки того, что строка начинается со строчной буквы в Python, можно использовать метод строки `islower()` в сочетании с методом `startswith()`.

19. Можно ли в Python прибавить целое число к строке?

В Python прибавление целого числа к строке не является автоматическим, и это приведет к ошибке типов. Однако, вы можете преобразовать целое число в строку с использованием функции `str()` и затем произвести конкатенацию.

20. Как «перевернуть» строку?

Для "переворачивания" строки в Python можно использовать срез с шагом -1. Это позволяет создать новую строку, элементы которой идут в обратном порядке. Пример: `revesed_string = my_string[::-1]`

21. Как объединить список строк в одну строку, элементы которой разделены дефисами?

Для объединения списка строк в одну строку, элементы которой разделены дефисами, вы можете использовать метод строки `join()`. Пример: `result_string = "-".join(string_list)`

22. Как привести всю строку к верхнему или нижнему регистру?

Для приведения строки к верхнему или нижнему регистру в Python, можно использовать методы строки `upper()` для верхнего регистра и `lower()` для нижнего регистра.

23. Как преобразовать первый и последний символы строки к верхнему регистру?

`result_string = my_string[0].upper() + my_string[1:-1] + my_string[-1].upper()`

24. Как проверить строку на то, что она составлена только из прописных букв?

Для проверки того, что строка составлена только из прописных букв в Python, можно использовать метод строки `islower()`.

25. В какой ситуации вы воспользовались бы методом `splitlines()`?

Метод `splitlines()` в Python используется для разделения строки на подстроки на основе символов новой строки (`\n`), возврата каретки (`\r`) или их

комбинации. Этот метод полезен, например, при чтении текстовых файлов, где строки могут быть разделены различными символами новой строки в зависимости от операционной системы.

26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?

Для замены всех вхождений подстроки в заданной строке в Python используется метод `replace()`

27. Как проверить то, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов?

Для проверки того, что строка начинается с заданной последовательности символов, используется метод строки `startswith()`. Для проверки того, что строка заканчивается заданной последовательностью символов, используется метод `endswith()`.

28. Как узнать о том, что строка включает в себя только пробелы?

Для проверки того, что строка состоит только из пробелов в Python, используется метод строки `isspace()`.

29. Что случится, если умножить некую строку на 3?

Если умножить строку на число в Python, строка будет повторена указанное количество раз.

30. Как привести к верхнему регистру первый символ каждого слова в строке?

Для приведения к верхнему регистру первого символа каждого слова в строке в Python, можно использовать метод `title()`.

31. Как пользоваться методом `partition()`?

Метод `partition()` в Python используется для разделения строки на три части с использованием указанного разделителя. Метод возвращает кортеж, содержащий часть строки до разделителя, сам разделитель и часть строки после разделителя.

32. В каких ситуациях пользуются методом `rfind()`?

Метод `rfind()` в Python используется для поиска последнего вхождения подстроки в строке. Он возвращает индекс последнего вхождения указанной подстроки или -1, если подстрока не найдена.

**Выводы:** В процессе выполнения лабораторной работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x, были написаны 7 программ, среди которых 3 программы с примерами лабораторной работы, 3 программы с индивидуальными заданиями и 1 программа с задачей повышенной сложности.