

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 «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
1 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

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.

Порядок выполнения работы

1. Создание репозитория GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

BigLofanbl4 / lab6

lab6 is available.

Great repository names are short and memorable. Need inspiration? How about [effective-fortnight](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

2. Проработал примеры из лабораторной работы.

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

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

Рисунок 2.1 – Код из примера 1

```
Введите предложение: Hello World!
Предложение после замены: Hello_World!
```

Рисунок 2.2 – Вывод программы из примера 1

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

if __name__ == "__main__":
    word = input("Введите слово: ")
    idx = len(word) // 2

    if len(word) % 2 == 1:
        r = word[:idx] + word[idx + 1 :]
    else:
        r = word[:idx - 1] + word[idx + 1 :]
    print(r)
```

Рисунок 2.3 – Код из примера 2

```
Введите слово: hello
helo
```

Рисунок 2.4 – Вывод программы из примера 2

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

import sys

if __name__ == "__main__":
    s = input("Введите предложение: ")
    n = int(input("Введите длину: "))
    if len(s) >= n:
        print("Заданная длина должна быть больше длины предложения", file=sys.stderr)
        exit(1)

    words = s.split(" ")

    if len(words) < 2:
        print("Предложение должно содержать несколько слов", file=sys.stderr)
        exit(1)

    delta = n

    for word in words:
        delta -= len(word)

    w, r = delta // (len(words) - 1), delta % (len(words) - 1)

    lst = []

    for i, word in enumerate(words):
        lst.append(word)

        if i < len(words) - 1:
            width = w

            if r > 0:
                width += 1
                r -= 1

            if width > 0:
                lst.append(" " * width)

    print("".join(lst))
```

Рисунок 2.5 – Код из примера 3

```
Введите предложение: hello world
Введите длину: 20
hello          world
```

Рисунок 2.6 – Вывод программы из примера 3

3. Дано предложение. Вывести «столбиком» его первый, второй, пятый, шестой, девятый, десятый и т. д. символы.

```

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

if __name__ == "__main__":
    sentence = input("Enter sentente: ")
    index = 0
    while index < len(sentence):
        print(sentence[index])
        if index + 1 < len(sentence):
            print(sentence[index + 1])
        index += 4

```

Рисунок 3.1 – Код программы

```

Enter sentente: 1234567891
1
2
5
6
9
1

```

Рисунок 3.2 – Вывод программы

4. Дана строка, в которой есть слово или. Определить, сколько раз оно встречается.

```

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

if __name__ == "__main__":
    sentence = "или были да кабили или да"
    count = sentence.split(" ").count("или")
    print(f"Число слов или: {count}")

```

Рисунок 4.1 – Код программы

```

Число слов или: 2

```

Рисунок 4.2 – Вывод программы

5. Дано слово. Переставить его первую букву на место последней. При этом вторую, третью, ..., последнюю буквы сдвинуть влево на одну позицию.

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

if __name__ == "__main__":
    word = input("Enter word: ")
    word = word[1:] + word[0]
    print(word)
```

Рисунок 5.1 – Код программы

```
Enter word: pog
ogp
```

Рисунок 5.2 – Вывод программы

6. Дано предложение. Поменять местами его первое и последнее слова.

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

if __name__ == "__main__":
    sentence = input("Enter sentence: ").split()
    sentence[0], sentence[-1] = sentence[-1], sentence[0]
    print(f'Result: {" ".join(sentence)}')
```

Рисунок 7 – Код программы

```
Enter sentence: Hello world
Result: world Hello
```

Рисунок 8 – Вывод программы

Ответы на контрольные вопросы

1. Строки в языке Python - это последовательность символов, заключенных в кавычки (одинарные, двойные или тройные).

2. Существуют следующие способы задания строковых литералов в языке Python:

- Одиночные кавычки: 'строка'
- Двойные кавычки: "строка"
- Тройные кавычки: """строка""" или """"строка""""

3. Операции и функции для строк:

- Конкатенация (+): объединение двух строк
- Умножение (*): повторение строки заданное количество раз
- Индексирование ([]): доступ к отдельным символам строки по их

индексу

- Срезы ([]): получение подстроки из строки
- Методы: функции, применяемые к строке с помощью точечной

нотации

4. Индексирование строк осуществляется с помощью квадратных скобок ([]). Первый символ имеет индекс 0, второй - индекс 1 и так далее. Можно использовать отрицательные индексы для доступа к символам с конца строки (-1 для последнего символа, -2 для предпоследнего и т.д.).

5. Работа со срезами строк осуществляется с помощью оператора [:]. Синтаксис: [начало:конец:шаг]. Начало и конец указывают индексы символов, включая начальный и исключая конечный. Шаг определяет, какие символы будут взяты (положительное значение - с начала строки, отрицательное - с конца).

6. Строки Python относятся к неизменяемому типу данных, что означает, что после создания строки ее нельзя изменить. Вместо этого при выполнении операций со строками создается новая строка.

7. Для проверки того, что каждое слово в строке начинается с заглавной буквы, можно воспользоваться методом `istitle()`. Он возвращает `True`, если каждое слово начинается с заглавной буквы, и `False` в противном случае.

8. Для проверки наличия подстроки в строке можно воспользоваться оператором `in`. Он возвращает `True`, если подстрока присутствует в строке, и `False` в противном случае.

9. Для поиска индекса первого вхождения подстроки в строку можно использовать метод `find()`. Он возвращает индекс первого символа подстроки, если она найдена, и `-1` в противном случае.

10. Для подсчета количества символов в строке можно воспользоваться функцией `len()`. Она возвращает количество символов в строке.

11. Для подсчета количества вхождений определенного символа в строку можно воспользоваться методом `count()`. Он возвращает количество вхождений символа в строку.

12. F-строки (форматированные строки) позволяют вставлять значения переменных и выражений в строку с помощью фигурных скобок `{}` и префикса `f` перед строкой. Пример: `name = "John"; age = 25; print(f"My name is {name} and I am {age} years old.")`

13. Для поиска подстроки в заданной части строки можно использовать метод `find()`, указав начальный и конечный индексы.

14. Для вставки содержимого переменной в строку с помощью метода `format()` нужно использовать фигурные скобки `{}` внутри строки и передать значения переменных в метод `format()` в качестве аргументов.

15. Для проверки того, что строка содержит только цифры, можно воспользоваться методом `isdigit()`. Он возвращает `True`, если все символы строки являются цифрами, и `False` в противном случае.

16. Для разделения строки по заданному символу можно использовать метод `split()`. Он разделяет строку на список подстрок по указанному символу и возвращает этот список.

17. Для проверки того, что строка состоит только из строчных букв, можно воспользоваться методом `islower()`. Он возвращает `True`, если все символы строки являются строчными буквами, и `False` в противном случае.

18. Для проверки того, что строка начинается со строчной буквы, можно воспользоваться методом `islower()` для первого символа строки.

19. Нельзя прибавить целое число к строке в Python, так как операция сложения не определена для разных типов данных.

20. Для "переворота" строки можно воспользоваться срезами с отрицательным шагом. Например: `s = "hello"; reversed_s = s[::-1]`

21. Для объединения списка строк в одну строку, элементы которой разделены дефисами, можно воспользоваться методом `join()`. Например: `words = ['hello', 'world']; sentence = '-'.join(words)`

22. Для приведения строки к верхнему регистру можно использовать метод `upper()`. Он возвращает новую строку, где все символы приведены к верхнему регистру.

23. Для преобразования первого и последнего символов строки к верхнему регистру можно использовать методы `capitalize()` и `title()`.

24. Для проверки того, что строка состоит только из прописных букв, можно воспользоваться методом `isupper()`. Он возвращает `True`, если все символы строки являются прописными буквами, и `False` в противном случае.

25. Метод `splitlines()` используется для разделения строки на список строк по символу новой строки (`'\n'`).

26. Для замены всех вхождений подстроки на что-либо другое в заданной строке можно использовать метод `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 состоит только из пробелов, можно использовать метод `isspace()`. Он возвращает `True`, если все символы строки являются пробелами, и `False` в противном случае.

29. Если умножить строку на целое число, то она будет повторена заданное количество раз. Например, "abc" * 3 вернет "abcabcabc".

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

31. Метод `partition()` разделяет строку на три части по первому вхождению указанного разделителя. Он возвращает кортеж из трех элементов: часть строки до разделителя, сам разделитель и часть строки после разделителя.

32. Метод `rfind()` используется для поиска последнего вхождения подстроки в строку. Он возвращает индекс первого символа последнего вхождения подстроки, если она найдена, и -1 в противном случае.