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 «Основы программной инженерии»

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

_____
(подпись)

Преподаватель:
Воронкин Р.А., канд. тех. наук, доцент,
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.

Результаты выполнения

1. Создали репозиторий с лицензией MIT, добавили в .gitignore необходимые правила для работы с IDE PyCharm, клонировали репозиторий, организовали репозиторий в соответствии с моделью git-flow.

```
Last login: Tue Oct 24 20:29:02 on ttys000
itssyoma@MacBook-Air-Sema Основы программной инженерии % git clone https://
github.com/itssyoma/megarepo_21.git
Cloning into 'megarepo_21'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
itssyoma@MacBook-Air-Sema Основы программной инженерии % cd megarepo_21
itssyoma@MacBook-Air-Sema megarepo_21 % git checkout -b develop
Switched to a new branch 'develop'
itssyoma@MacBook-Air-Sema megarepo_21 % git branch release
itssyoma@MacBook-Air-Sema megarepo_21 % git branch develop
fatal: a branch named 'develop' already exists
itssyoma@MacBook-Air-Sema megarepo_21 % git branch hotfix
itssyoma@MacBook-Air-Sema megarepo_21 % git branch feature
itssyoma@MacBook-Air-Sema megarepo_21 %
```

Рисунок 1 – Работа с репозиторием в командной строке.

2. Проработали пример №1.

example1.py > ...

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


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

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bi
ed/Учеба/Основы программной инженерии/megar
epo_23/example1.py"
Введите предложение: Привет! Меня зовут Сёма.
Предложение после замены: Привет!_Меня_зовут_Сёма.
```

Рисунок 1 – Пример №1. Код и результат выполнения программы.

3. Проработали пример №2.

example2.py > ...

```python
#!/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)

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

```
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "
ed/Учеба/Основы программной инженерии/megarepo_23/example2.py"
Введите слово: СЛОВО
СЛВО
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "
ed/Учеба/Основы программной инженерии/megarepo_23/example2.py"
Введите слово: СЛОВЕЧКО
СЛОЧКО
```

Рисунок 2 – Пример №2. Код и результат выполнения программы.

4. Проработали пример №3.

example3.py > ...

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

import sys


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

    # Проверить требуюемую 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))

```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной инженерии/megarepo_23/example3.py"
Введите предложение: Привет! Как твои дела?
Введите длину: 28
Привет!   Как   твои   дела?
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной инженерии/megarepo_23/example3.py"
Введите предложение: Как пройти до библиотеки?
Введите длину: 38
Как      пройти     до     библиотеки?
```

Рисунок 3 – Пример №3. Код и результат выполнения программы.

5. Выполнили индивидуальное задание №1 (вариант 1). Дано название футбольного клуба. Напечатать его на экране «столбиком».

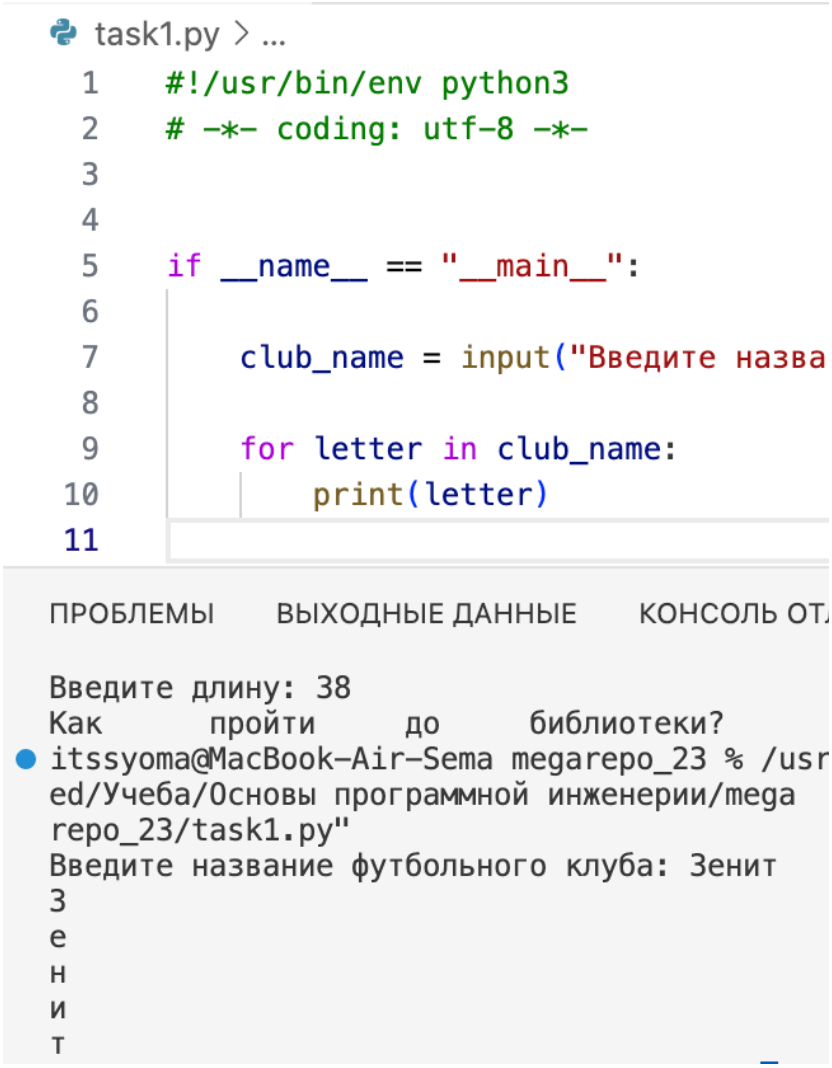

Рисунок 4 – Задание №1. Код и результат выполнения программы.

6. Выполнили индивидуальное задание №2 (вариант 1). Дано предложение. Определить, есть ли буква а в нем. В случае положительного ответа найти также порядковый номер первой из них.

task2.py > ...

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

import sys

if __name__ == "__main__":

    sentence = input("Введите предложение: ")

    if sentence.isdigit():
        print("Введенное предложение состоит из цифр", file=sys.stderr)
        exit(1)

    if "a" in sentence:
        print(f"Буква а найдена в предложении на позиции {sentence.find('a')+1}")
    else:
        print("Буква а в предложении не найдена")

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Yandex.Disk
ed/Учеба/Основы программной инженерии/megarepo_23/task2.py"
Введите предложение: Я люблю Дуа Липу
Буква а найдена в предложении на позиции 11
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Yandex.Disk
ed/Учеба/Основы программной инженерии/megarepo_23/task2.py"
Введите предложение: Я люблю Тейлор Свифт
Буква а в предложении не найдена
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Yandex.Disk
ed/Учеба/Основы программной инженерии/megarepo_23/task2.py"
Введите предложение: 9428594829
Введенное предложение состоит из цифр
```

Рисунок 5 – Задание №2. Код и результат выполнения программы.

7. Выполнили индивидуальное задание №3 (вариант 1). 1. Дано слово. Удалить из него третью букву. Удалить из него k-ю букву.

task3.py > ...

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

import sys


if __name__ == "__main__":

    word = input("Введите слово: ")

    if len(word) < 3:
        print("Слово содержит менее трех букв", file=sys.stderr)
        exit(1)

    k = int(input("Какую букву, помимо 3-ей, нужно удалить? "))

    if k == 3 or k > len(word) or k == 0:
        print("Неверное значение k", file=sys.stderr)
        exit(1)

    word_result = ""
    if k < 3:
        if k == 1:
            word_result += word[1]
        elif k == 2:
            word_result += word[0]
        if len(word) > 3:
            word_result += word[3:]
    else:
        word_result += word[:2]
        if k >= 5:
            word_result += word[3:k-1]
        if k != len(word):
            word_result += word[k:]

    print(word_result)

```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/i
epo_23/task3.py"
Введите слово: абвг
Какую букву, помимо 3-ей, нужно удалить? 4
аб
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/i
epo_23/task3.py"
Введите слово: абвгд
Какую букву, помимо 3-ей, нужно удалить? 4
абд
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/i
epo_23/task3.py"
Введите слово: абвгд
Какую букву, помимо 3-ей, нужно удалить? 5
абг
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/i
Введите слово: аб
Слово содержит менее трех букв
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/i
Введите слово: абвгдеж
Какую букву, помимо 3-ей, нужно удалить? 0
Неверное значение k
```

Рисунок 6 – Задание №3. Код и результат выполнения программы.

8. Выполнили задание повышенной сложности (вариант 1). Дано предложение. Найти наибольшее количество идущих подряд пробелов.

hard.py > ...

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


if __name__ == "__main__":

    sentence = input("Введите предложение: ")

    max_spaces = 0
    current_spaces = 0

    for char in sentence:
        if char == " ":
            current_spaces += 1
        else:
            max_spaces = max(max_spaces, current_spaces)
            current_spaces = 0

    print(f"Наибольшее количество идущих подряд пробелов: {max_spaces}")
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Ya
ed/Учеба/Основы программной инженерии/mega
repo_23/hard.py"
Введите предложение: Привет! Как  дела   ?
Наибольшее количество идущих подряд пробелов: 3
itssyoma@MacBook-Air-Sema megarepo_23 % /usr/local/bin/python3 "/Users/itssyoma/Ya
ed/Учеба/Основы программной инженерии/megar
epo_23/hard.py"
Введите предложение: раз         два  три
Наибольшее количество идущих подряд пробелов: 9
```

Рисунок 7 – Задание повышенной сложности.

Код и результат выполнения программы.

Ответы на контрольные вопросы

1. Что такое строки в языке Python?

Строки - это последовательность символов, заключенных в кавычки, которые могут быть использованы для хранения и обработки текстовых данных в программе на языке Python.

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

- С помощью одинарных кавычек: 'строка'

- С помощью двойных кавычек: "строка"

- С помощью тройных кавычек: '"многострочная

строка"'

3. Какие операции и функции существуют для строк?

- Конкатенация (+)
- Умножение (*)
- Индексирование ([])
- Срезы ([:])
- Длина строки (len())
- Поиск подстроки (find(), index())
- Замена подстроки (replace())
- Разделение строки на подстроки (split())
- Объединение списка строк в одну строку (join())
- Приведение к верхнему/нижнему регистру (upper(), lower())

4. Как осуществляется индексирование строк?

Индексирование строк осуществляется с помощью квадратных скобок []. Индексация начинается с 0, т.е. первый символ имеет индекс 0, второй – 1 и т.д. Отрицательные индексы начинаются с -1, т.е. последний символ имеет индекс -1, предпоследний -2 и т.д.

5. Как осуществляется работа со срезами для строк?

Работа со срезами для строк осуществляется с помощью оператора [:]. Срезы позволяют выбирать из строки подстроку, начиная с определенного индекса и заканчивая другим индексом. Например, строка[2:5] вернет подстроку, начиная с третьего символа и заканчивая пятым (не включая его).

6. Почему строки Python относятся к неизменяемому типу данных?

Строки Python относятся к неизменяемому типу данных, потому что после создания строки ее содержимое нельзя изменить. Вместо этого создается новая строка с измененным содержимым.

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 воспользоваться методом istitle(). Он возвращает True, если каждое слово начинается с заглавной буквы, и False в противном случае.

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

Для проверки того, что одна строка содержится в другой строке можно воспользоваться оператором in. Например, "abc" in "abcdefg" вернет True.

9. Как найти индекс первого вхождения подстроки в строку?

Для нахождения индекса первого вхождения подстроки в строку можно воспользоваться методом find() или index(). Они оба возвращают индекс первого символа найденной подстроки или -1, если подстрока не найдена. Разница между ними заключается в том, что метод find() возвращает -1, если подстрока не найдена, а метод index() вызывает исключение ValueError.

10. Как подсчитать количество символов в строке?

Для подсчета количества символов в строке можно воспользоваться функцией len(). Например, len("abcdefg") вернет 7.

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

Для подсчета того, сколько раз определенный символ встречается в строке можно воспользоваться методом count(). Например, "abcdefg".count("a") вернет 1.

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

F-строки - это новый способ форматирования строк, который был добавлен в Python 3.6. Они позволяют вставлять значения переменных прямо внутрь строки, используя фигурные скобки и передавая значения переменных через f-строку.

13. Как найти подстроку в заданной части строки?

Для поиска подстроки в заданной части строки можно использовать методы find() или index() с указанием начального и конечного индексов. Например, "abcdefg".find("c", 2, 5) вернет индекс символа "c" в строке "abcdefg", начиная с третьего символа и заканчивая пятым (не включая его).

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

Для вставки содержимого переменной в строку можно воспользоваться методом format(). Например, "Меня зовут и мне лет".format(name, age).

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

Для проверки того, что в строке содержатся только цифры можно воспользоваться методом isdigit(). Он 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 можно воспользоваться методом split(). Например, "a-b-c".split("-") вернет список ["a", "b", "c"].

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

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

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

Для проверки того, что строка начинается со строчной буквы можно воспользоваться методом islower() и индексированием. Например, "abc".islower() and "abc"[0].islower() вернет True.

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

В Python нельзя прибавить целое число к строке. Это вызовет ошибку TypeError. Однако, можно преобразовать целое число в строку и затем произвести конкатенацию строк.

20. Как «перевернуть» строку?

Для "переворачивания" строки можно воспользоваться срезами с отрицательными индексами. Например, "abcdefg"[::-1] вернет строку "gfedcba".

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

Для объединения списка строк в одну строку с разделителем можно воспользоваться методом join(). Например, "-".join(["a", "b", "c"]) вернет

строку "a-b-c".

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

Для приведения всей строки к верхнему или нижнему регистру можно воспользоваться методами upper() и lower(). Например, "AbCdEfG".upper() вернет строку "ABCDEFG".

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

Можно использовать методы upper() и capitalize() для преобразования первого символа в верхний регистр, а также индексирование для преобразования последнего символа. Например:

```
s = "example"
s = s[0].upper() + s[1:-1] + s[-1].upper()
print(s)  # "ExamplE"
```

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

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

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

Метод splitlines() используется для разделения строки на подстроки по символу перевода строки (). Например, "a\nb\nc".splitlines() вернет список ["a", "b", "c"].

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

Для замены всех вхождений подстроки в заданной строке можно воспользоваться методом replace(). Например, "abcdefg".replace("c", "X") вернет строку "abXdefg".

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

Для проверки того, что строка начинается или заканчивается заданной последовательностью символов можно воспользоваться методами startswith() и endswith(). Например, "abcdefg".startswith("ab") вернет True.

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

Для проверки того, что строка состоит только из пробелов можно воспользоваться методом isspace(). Он возвращает True, если все символы в строке являются пробелами, и False в противном случае.

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

Если умножить некую строку на целое число, то получится новая строка, которая будет содержать исходную строку повторенную указанное количество раз. Например, "abc" * 3 вернет строку "abcabcabc".

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

Для приведения к верхнему регистру первого символа каждого слова в строке можно воспользоваться методом title(). Например, "this is a test".title() вернет строку "This Is A Test".

31. Как пользоваться методом partition()?

Метод partition() используется для разделения строки на три части: часть до первого вхождения заданной подстроки, саму подстроку и часть после нее. Например, "a-b-c".partition("-") вернет кортеж ("a", "-", "b-c").

32. В каких ситуациях пользуются методом rfind()?

Метод rfind() используется для поиска последнего вхождения подстроки в заданной строке. Он работает аналогично методу find(), но начинает поиск с конца строки. Если подстрока не найдена, метод возвращает -1.