

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

Отчет по лабораторной работе №6

по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-20-1

Ваньянц И.М. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2021

ХОД РАБОТЫ

```
C:\Users\Илья\lab-6>git checkout -b develop  
Switched to a new branch 'develop'
```

Рисунок 1 – создание ветки “develop”

1. Пример 1

```
1 ▶ #!/usr/bin/env python3  
2 # -*- coding: utf-8 -*-  
3  
4 ▶ if __name__ == '__main__':  
5     s = input("Enter the sentence: ")  
6     r = s.replace(' ', '_')  
7     print("The sentence after replacement: {}".format(r))
```

Рисунок 2 – код примера

```
Enter the sentence: поставьте пожалуйста оценку  
The sentence after replacement: поставьте_пожалуйста_оценку
```

Рисунок 3 – результат

2. Пример 2

```
1 ▶ #!/usr/bin/env python3  
2 # -*- coding: utf-8 -*-  
3  
4 ▶ if __name__ == '__main__':  
5     word = input("Enter the word: ")  
6  
7     idx = len(word) // 2  
8     if len(word) % 2 == 1:  
9         # Длина слова нечетная.  
10        r = word[:idx] + word[idx+1:]  
11    else:  
12        # Длина слова четная.  
13        r = word[:idx-1] + word[idx+1:]  
14  
15    print(r)
```

Рисунок 4 – код примера

```
Enter the word: loop
lp
Process finished with exit code 0
```

Рисунок 5 - вывод программы при четной длине слова

```
Enter the word: cat
ct
Process finished with exit code 0
```

Рисунок 6 - вывод программы при нечетной длине слова

3. Пример 3.

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     s = input("Введите предложение: ")
8     n = int(input("Введите длину: "))
9
10    # Проверить требуемую длину.
11    if len(s) >= n:
12        print(
13            "Заданная длина должна быть больше длины предложения",
14            file=sys.stderr
15        )
16        exit(1)
17
18    # Разделить предложение на слова.
19    words = s.split(' ')
20    # Проверить количество слов в предложении.
21    if len(words) < 2:
22        print(
23            "Предложение должно содержать несколько слов",
24            file=sys.stderr
25        )
26        exit(1)
27
28    # Количество пробелов для добавления.
29    delta = n
```

```

30     for word in words:
31         delta -= len(word)
32
33     # Количество пробелов на каждое слово.
34     w, r = delta // (len(words) - 1), delta % (len(words) - 1)
35
36     # Сформировать список для хранения слов и пробелов.
37     lst = []
38
39     # Пронумеровать все слова в списке и перебрать их.
40     for i, word in enumerate(words):
41         lst.append(word)
42
43         # Если слово не является последним, добавить пробелы.
44         if i < len(words) - 1:
45             # Определить количество пробелов.
46             width = w
47             if r > 0:
48                 width += 1
49                 r -= 1
50
51             # Добавить заданное количество пробелов в список.
52             if width > 0:
53                 lst.append(' ' * width)
54
55     # Вывести новое предложение, объединив все элементы списка lst.
56     print(''.join(lst))

```

Рисунки 7 и 8 – код программы

```

Введите предложение: cat is sleeping
Введите длину: 25
cat      is      sleeping

Process finished with exit code 0

```

Рисунок 9 – вывод при верном вводе

```

Введите предложение: cat is sleeping
Введите длину: 10
Заданная длина должна быть больше длины предложения

Process finished with exit code 1

```

Рисунок 10 – вывод при неправильном вводе

4. Индивидуальное задание 1

Дано слово. Добавить к нему в начале и конце столько звездочек, сколько букв в этом слове.

```
word = input("Введите слово: ")
k = len(word)
print(f"Длина слова: {k}")
word = k * '*' + word + k * '*'
print(word)
```

Рисунок 11 – код программы

```
Введите слово: Четверка
Длина слова: 8
*****Четверка*****
```

Рисунок 12 – результат

5. Индивидуальное задание 2

Даны два слова. Определить, сколько начальных букв первого слова совпадает с начальными буквами второго слова. Рассмотреть два случая:

известно, что слова разные;

слова могут быть одинаковыми.

```
word1, word2 = input("Введите два слова: ").split()
if word1 == word2:
    print("Все символы в этих словах одинаковые!")
    exit(0)
else:
    k = 0
    for i in range(len(word1)):
        if word1[i] in word2[i]:
            k += 1
        else:
            break
    print(f"В данных словах содержится {k} одинаковых начальных символов")
```

Рисунок 13 – код программы

```
Введите два слова: поставьте пожалуйста
В данных словах содержится 2 одинаковых начальных символов
```

Рисунок 14 – результат

6. Индивидуальное задание 3

Дано предложение. Удалить из него все буквы с (как в кириллице так и на латинице).

```
sentence = input("Введите предложение: ")
sentence = sentence.replace('с', '') # Удаление английских букв "с"
sentence = sentence.replace('с', '') # Удаление русских букв "с"
sentence = sentence.replace('С', '') # Удаление английских букв "С"
sentence = sentence.replace('С', '') # Удаление русских букв "С"
print(sentence)
```

Рисунок 15 – код программы

```
Введите предложение: Шла Саша по шоссе и сосала сушку
Шла аша по шое и оала ушку
```

Рисунок 16 – результат программы

7. Задание повышенной сложности

Даны два слова. Для каждой буквы первого слова (в том числе для повторяющихся в этом слове букв) определить, входит ли она во второе слово. Например, если заданные слова информация и процессор, то для букв первого из них ответом должно быть: нет нет нет да да нет нет да нет нет.

```
word1, word2 = input("Введите два слова: ").split()
for ch in word1:
    if ch in word2:
        print("Да", end=" ")
    else:
        print("Нет", end=" ")
```

Рисунок 17 – код программы

```
Введите два слова: информация процессор
Нет Нет Нет Да Да Нет Нет Да Нет Нет
```

Рисунок 18 – вывод для заданных условие слов

```
Введите два слова: умоляю оценку
Да Нет Да Нет Нет Нет
```

Рисунок 19 – вывод для произвольных слов

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

2. Строки в апострофах и в кавычках, экранированные последовательности - служебные символы, "Сырые" строки, строки в тройных апострофах или кавычках.
3. Сложение, умножение, оператор принадлежности. Строковых функций в Python много, вот некоторые из них: `chr()` – Преобразует целое число в символ `ord()` – Преобразует символ в целое число `len()` – Возвращает длину строки `str()` – Изменяет тип объекта на `string`
4. В Python строки являются упорядоченными последовательностями символьных данных и могут быть проиндексированы. Доступ к отдельным символам в строке можно получить, указав имя строки, за которым следует число в квадратных скобках `[]`. Индексация строк начинается с нуля: у первого символа индекс 0, следующего 1 и так далее. Индекс последнего символа в python — “длина строки минус один”.
5. Если `s` это строка, выражение формы `s[m:n]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая позицию. Если пропустить первый индекс, срез начинается с начала строки. Аналогично, если опустить второй индекс `s[n:]`, срез длится от первого индекса до конца строки.
6. Более легкое представление в памяти.
7. `s.istitle()`
8. `if s1 in s2`
9. `s.find()`.
10. `len(s)`
11. `s.count()`.
12. f-строки упрощают форматирование строк. Пример: `print(f' This is {name}, he is {age} years old')`
13. `string.find([, [,]])`
14. `'Hello, { }!'.format('Vasya')`
15. `string.isdigit()`
16. `'foo.bar.baz.qux'.rsplit(sep='.')` – пример разделения
17. `string.islower()`
18. `s[0].isupper()`
19. С точки зрения математической операции нельзя, можно лишь только вывести из без разделения друг от друга
20. `s[::-1]` – при помощи среза.
21. `'-'.join()`
22. К верхнему – `string.upper()`, к нижнему – `string.lower()`.
23. `s[0].upper()` `s[len(s) - 1].upper()`
24. `s.isupper()`
25. Если нужно сохранить символы, обозначающие конец слов.
26. `s.replace('что заменить', 'на что заменить')`
27. `string.endswith([, [,]])`, `str.startswith(prefix[, start[, end]])`
28. `s.isspace()`
29. Будет получена копия исходной строки в трёхкратном размере.
30. `s.title()`
31. `s.partition()` отделяет от `s` подстроку длиной от начала до первого вхождения . Возвращаемое значение представляет собой кортеж из трех частей: Часть `s` до Разделитель Часть `s` после
32. Когда нужен индекс последнего вхождения подстроки в строку.