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

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

if __name__ == '__main__':
    s = input("Enter the sentence: ")
    r = s.replace(' ', '_')
    print("The sentence after replacement: {0}".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

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

if __name__ == '__main__':
    word = input("Enter the word: ")

    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)
```

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.

```python
#!/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))
```

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: cqt 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.

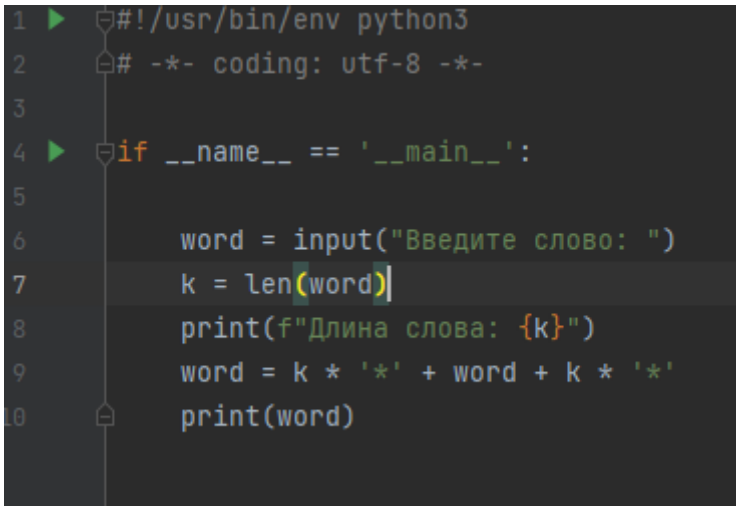

Рисунок 11 – код программы

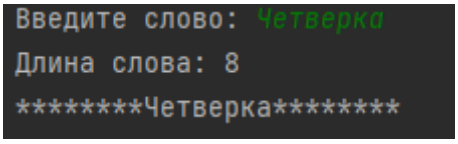

Рисунок 12 – результат

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.

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

if __name__ == '__main__':

    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).

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

if __name__ == '__main__':

    sentence = input("Введите предложение: ")
    sentence = sentence.replace('c', '')  # Удаление английских букв "c"
    sentence = sentence.replace('с', '')  # Удаление русских букв "с"
    sentence = sentence.replace('C', '') #Удаление английских букв "C"
    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.

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

if __name__ == '__main__':

    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.tittle()
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.