

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №6 по дисциплине «Основы  
программной инженерии»

Выполнил:  
Мамонтов Д.В.,  
2 курс, группа ПИЖ-б-о-20-1,  
Проверил:  
Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2022 г

## ХОД РАБОТЫ

```
C:\Users\TEPLOX0\Desktop\donda\УЧЕБА\ОПИ\lab2>git checkout -b develop  
Switched to a new branch 'develop'
```

Рисунок 1 – создание ветки “develop”

```
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: ich weiss nicht was soll es bedeuten  
The sentence after replacement: ich_weiss_nicht_was_soll_es_bedeuten  
  
Process finished with exit code 0
```

Рисунок 3 – работа программы

```
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 программы

```
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

```
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
```

Рисунок 7 – код программы

```

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))

```

Рисунок 8 – продолжение кода

```

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

# ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 1

Задание:

Дано предложение. Составить программу, которая выводит все вхождения в предложение двух заданных символов.

Код:

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

if __name__ == '__main__':
    word = str(input("Введите строку:"))
    part1 = str(input("Введите первый символ:"))
    part2 = str(input("Введите второй символ:"))
    print("\n", "Координаты вхождений первого символа:", end='')
    for i in range(len(word) - 1):
        if word[i] == part1:
            print(' ', i, end='')
    print("\n", "Координаты вхождений второго символа:", end='')
    for i in range(len(word) - 1):
        if word[i] == part2:
            print(' ', i, end='')
```

Введите строку: *sosiskasplesenyu*

Введите первый символ: *s*

Введите второй символ: *e*

Координаты вхождений первого символа: 0 2 4 7 11

Координаты вхождений второго символа: 10 12

Рисунок 11 – вывод программы

## ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 2

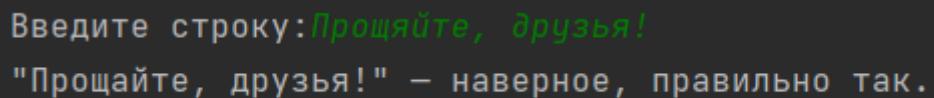
Задание:

Дана последовательность слов. Проверить, правильно ли в ней записаны буквосочетания ча и ща. Исправить ошибки.

Код:

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

if __name__ == '__main__':
    message = str(input("Введите строку:")) + "О"
    rule1 = "чя"
    rule2 = "щя"
    mistake = 0
    right = 1
    print("\n", end='')
    for i in range(len(message) - 1):
        if mistake == 0:
            print(message[i], end='')
            if ((message[i] + message[i + 1]) == rule1) or ((message[i] +
message[i + 1]) == rule2):
                mistake = 1
                right = 0
        else:
            print('a', end='')
            mistake = 0
    if right:
        print("\n – это верное предложение!")
    else:
        print("\n – наверное, правильно так.")
```



```
Введите строку:Просят, друзья!
"Прощайте, друзья!" – наверное, правильно так.
```

Рисунок 12 – вывод программы

## ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 3

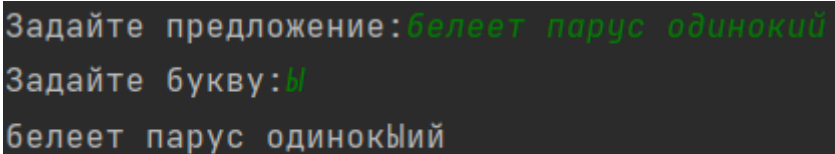
Задание:

Дано предложение, оканчивающее символом «.». Вставить заданную букву перед последней буквой и.

Код:

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

if __name__ == '__main__':
    message = str(input("Задайте предложение:"))
    letter = str(input("Задайте букву:"))
    result = ""
    last = 0
    for i in range(len(message) - 1, -1, -1):
        result += message[i]
        if (message[i] == ".") and (not last):
            result += letter
            last = 1
    print(result[::-1])
```



```
Задайте предложение:белеет парус одинокий
Задайте букву:ы
белеет парус одинокийы
```

Рисунок 13 – вывод программы

## ЗАДАНИЕ ПОВЫШЕННОЙ СЛОЖНОСТИ

Задание:

Даны три слова. Напечатать их общие буквы. Повторяющиеся буквы каждого слова не рассматривать.

Код:

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

if __name__ == '__main__':
    word1 = str(input("Введите первое слово:"))
    word2 = str(input("Введите второе слово:"))
    word3 = str(input("Введите третье слово:"))
    repeat = ""

    for i in range(len(word1)):
        for j in range(len(word2)):
            for k in range(len(word3)):
                if (word1[i] == word2[j]) and (word2[j] == word3[k]):
                    repeat += word1[i]

    sorted_repeat = sorted(repeat)
    last = '0'
    print("Общие буквы: ", end='')
    if len(sorted_repeat) == 0:
        print("отсутствуют")
    else:
        for i in range(len(sorted_repeat)):
            if sorted_repeat[i] != last:
                last = sorted_repeat[i]
                print(sorted_repeat[i], " ", end='')

```

```
Введите первое слово:картофель
Введите второе слово:простофиля
Введите третье слово:остров
ртоооо
['о', 'о', 'о', 'о', 'р', 'т']
Общие буквы: о р т

```

Рисунок 14 – результат выполнения программы



## 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 `[]`. Индексации 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.isitle()`

8) `if s1 in s2`

9) `s.find(<sub>)`.

10) `len(s)`

11) `s.count(<char>)`.

12) f-строки упрощают форматирование строк. Пример: `print(f' This is {name}, he is {age} years old')`

13) `string.find(<sub>[, <start>[, <end>]])`

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(<iterable>)`

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(<suffix>[, <start>[, <end>]])`, `str.startswith(prefix[, start[, end]])`
- 28) `s.isspace()`
- 29) Будет получена копия исходной строки в трёхкратном размере.
- 30) `s.title()`
- 31) `s.partition(<sep>)` отделяет от `s` подстроку длиной от начала до первого вхождения `<sep>` .  
Возвращаемое значение представляет собой кортеж из трех частей:  
Часть `s` до `<sep>`  
Разделитель `<sep>`  
Часть `s` после `<sep>`
- 32) Когда нужен индекс последнего вхождения подстроки в строку.