

**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**

Выполнила:

Емельянова Яна

Александровна, 2 курс,

группа ПИЖ-б-о-20-1,

Проверил:

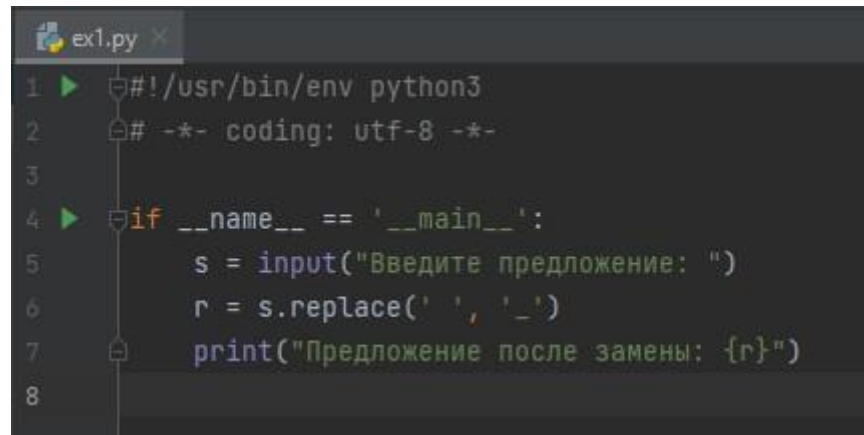
Доцент кафедры инфокоммуникаций,

Воронкин Р.А.

Ставрополь, 2021 г

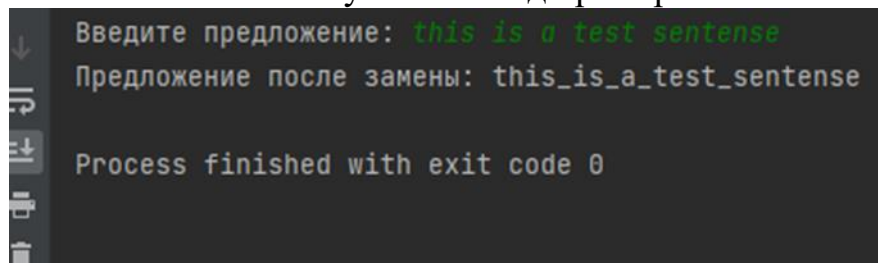
## 1. Работа со строками в языке Python

### 1.1 Пример 1 (рис. 1, 2).



```
ex1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      s = input("Введите предложение: ")
6      r = s.replace(' ', '_')
7      print("Предложение после замены: {r}")
8
```

Рисунок 1 – Код примера

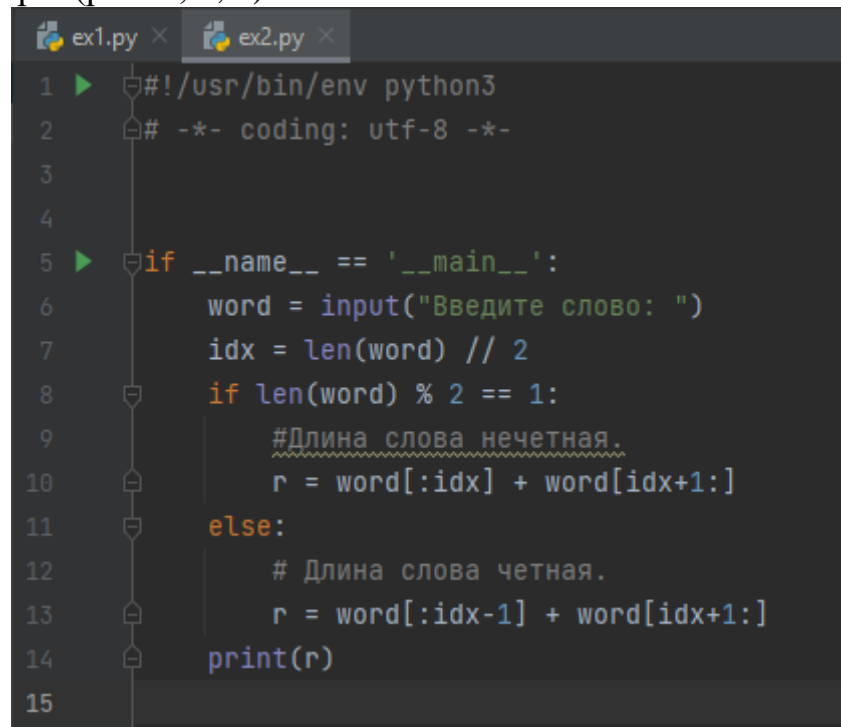


```
Введите предложение: this is a test sentence
Предложение после замены: this_is_a_test_sentence

Process finished with exit code 0
```

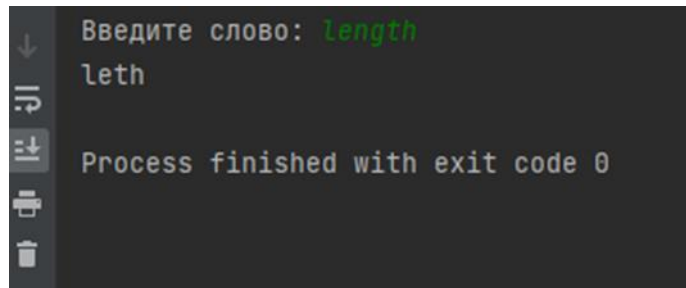
Рисунок 2 – Работа программы

### 1.2 Пример 2 (рис. 3, 4, 5).



```
ex1.py x  ex2.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  ▶  if __name__ == '__main__':
6      word = input("Введите слово: ")
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     print(r)
15
```

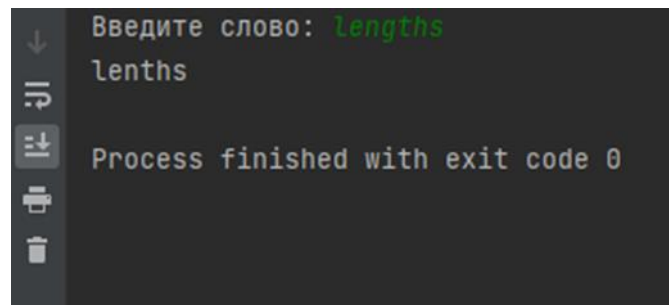
Рисунок 3 – Код программы



```
Введите слово: length
leth

Process finished with exit code 0
```

Рисунок 4 – Вывод программы при чётной длине слова



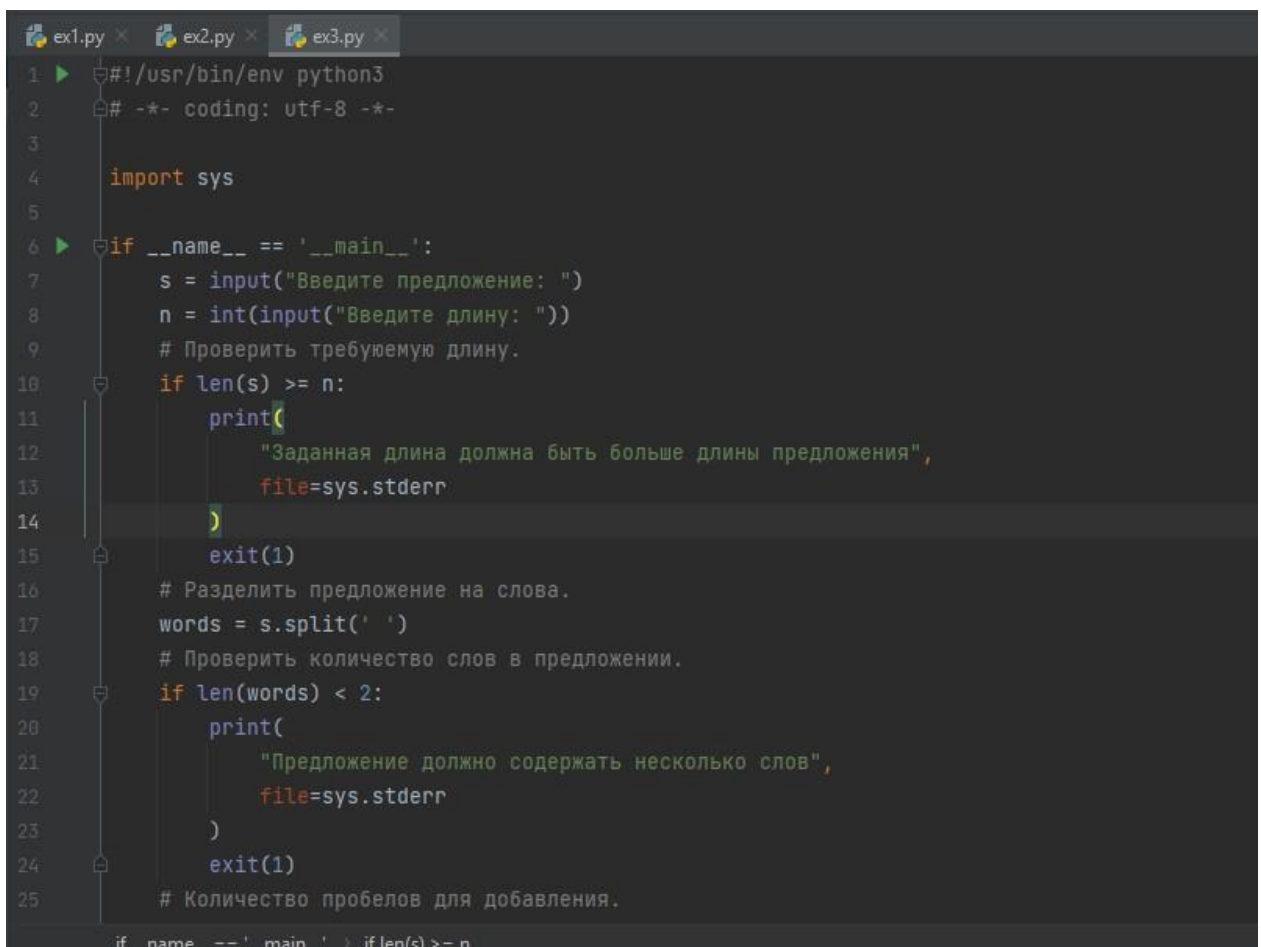
```
Введите слово: lengths
lenth

Process finished with exit code 0
```

Рисунок 5 – Вывод программы при нечётной длине слова

### 1.3 Пример 3 (рис. 6, 7, 8).

Код программы представлен на рисунках 6 и 7.



```
ex1.py x ex2.py x ex3.py x
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      # Проверить требуемую длину.
10     if len(s) >= n:
11         print(
12             "Заданная длина должна быть больше длины предложения",
13             file=sys.stderr
14         )
15         exit(1)
16     # Разделить предложение на слова.
17     words = s.split(' ')
18     # Проверить количество слов в предложении.
19     if len(words) < 2:
20         print(
21             "Предложение должно содержать несколько слов",
22             file=sys.stderr
23         )
24         exit(1)
25     # Количество пробелов для добавления.
```

Рисунок 6 – Код программы

```
ex1.py x ex2.py x ex3.py x
25 # Количество пробелов для добавления.
26 delta = n
27 for word in words:
28     delta -= len(word)
29 # Количество пробелов на каждое слово.
30 w, r = delta // (len(words) - 1), delta % (len(words) - 1)
31 # Сформировать список для хранения слов и пробелов.
32 lst = []
33 # Пронумеровать все слова в списке и перебрать их.
34 for i, word in enumerate(words):
35     lst.append(word)
36 # Если слово не является последним, добавить пробелы.
37 if i < len(words) - 1:
38     # Определить количество пробелов.
39     width = w
40     if r > 0:
41         width += 1
42         r -= 1
43     # Добавить заданное количество пробелов в список.
44     if width > 0:
45         lst.append(' ' * width)
46
47 # Вывести новое предложение, объединив все элементы списка lst.
48 print(''.join(lst))
49
if __name__ == '__main__': > if len(s) >= n
```

Рисунок 7 – Продолжение

```
Введите длину: 28
this  is a test sentence

Process finished with exit code 0
```

Рисунок 8 – Пример работы программы

## Решение индивидуальных заданий

### Вариант 9

#### 1.4 Индивидуальное задание №1 (рис. 9, 10).

Условие: Дано предложение. Вывести «столбиком» его третий, шестой и т. д. символы.

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

import sys

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

    for i in range(len(s)):
        if i % 3 == 0:
            print(s[i])
```

Рисунок 9 – Код программы

```
Введите предложение: Привет человек из сомалии
П
в

л
е
и
с
а
и
```

Рисунок 10 – Пример работы программы

#### 1.5 Индивидуальное задание №2 (рис. 11, 12).

Условие: Дано предложение. Определить, есть ли в нем буквосочетания чу или щу. В случае положительного ответа найти также порядковый номер первой буквы первого из них.

```
string = input("Введите предложение: ")

if 'чy' in string:
    print(string.find('чy'))
if 'щy' in string:
    print(string.find('щy'))
```

Рисунок 11 – Код программы

```
Введите предложение: Шучу с шутником над щукой
2
20
```

Рисунок 12 – Пример работы программы

### 1.6 Индивидуальное задание №3 (рис. 13, 14)

Условие: Дано слово, оканчивающееся символом «.». Составить программу, которая вставляет некоторую заданную букву после буквы с заданным номером.

```
word = input("Введите слово: ")

word = word[:3] + 'a' + word[3:]
print(word)
```

Рисунок 16 – Код программы

```
Введите слово: клубника
клубника

Process finished with exit code 0
|
```

Рисунок 17 – Работа программы

### 2. 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(<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>`