

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 №6
дисциплины «Программирование на Python»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокоммуникаций, доцент
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

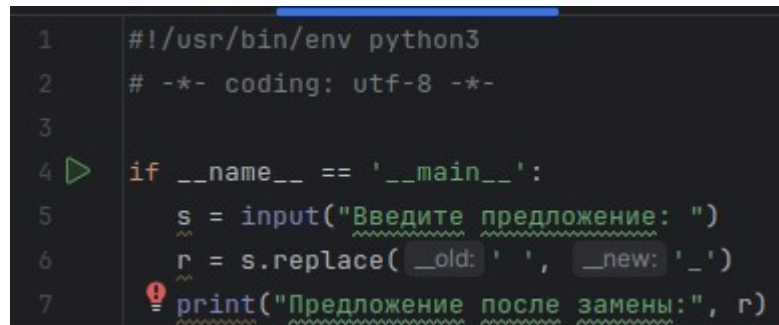
Ставрополь, 2023 г.

Тема: Работа со строками в языке Python

Цель: Приобретение навыков работы со строками при написании программ с помощью языка программирования Python версии 3.x

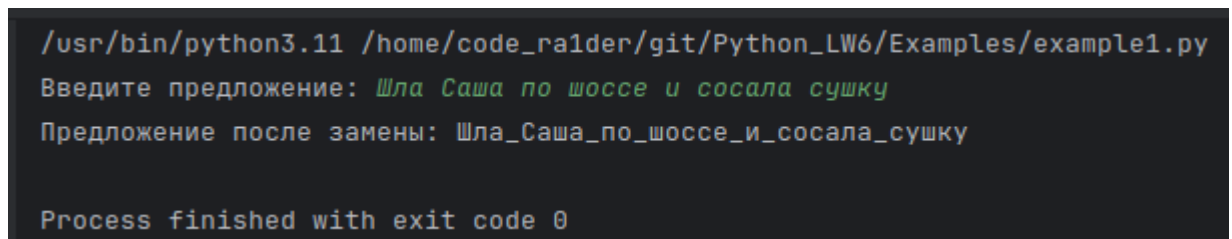
Порядок выполнения работы

1. Проработал пример 1: Дано предложение. Все пробелы в нем заменить символом «_»



```
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)
```

Рисунок 1. Код программы примера 1



```
/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Examples/example1.py
Введите предложение: Шла Саша по шоссе и сосала сушку
Предложение после замены: Шла_Саша_по_шоссе_и_сосала_сушку

Process finished with exit code 0
```

Рисунок 2. Результат работы программы

2. Проработал пример 2: Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае — две средние буквы

```

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

```

Рисунок 3. Код программы примера 2

```

/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Examples/example2.py
Введите слово: Программа
Программа

Process finished with exit code 0

```

Рисунок 4. Результат работы программы

3. Проработал пример 3. Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина стала равна заданной длине (предполагается, что требуемая длина не меньше исходной). Это следует сделать путем вставки между словами дополнительных пробелов. Количество пробелов между отдельными словами должно отличаться не более чем на 1

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  if __name__ == '__main__':
6      s = input("Введите предложение: ")
7      n = int(input("Введите длину: "))
8      # Проверить требуемую длину.
9      if len(s) >= n:
10         print(
11             "Заданная длина должна быть больше длины предложения",
12             file=sys.stderr
13         )
14         exit(1)
15     # Разделить предложение на слова.
16     words = s.split(' ')
17     # Проверить количество слов в предложении.
18     if len(words) < 2:
19         print(
20             "Предложение должно содержать несколько слов",
21             file=sys.stderr
22         )
23         exit(1)
24     # Количество пробелов для добавления.
25     delta = n
26     for word in words:
27         delta -= len(word)
28     # Количество пробелов на каждое слово.
29     w, r = delta // (len(words) - 1), delta % (len(words) - 1)
30     # Сформировать список для хранения слов и пробелов.
31     lst = []
32     # Пронумеровать все слова в списке и перебрать их.
33     for i, word in enumerate(words):
34         lst.append(word)

```

Рисунок 5. Код программы примера 3 (часть 1)

```

35     # Если слово не является последним, добавить пробелы.
36     if i < len(words) - 1:
37         # Определить количество пробелов.
38         width = w
39         if r > 0:
40             width += 1
41             r -= 1
42         # Добавить заданное количество пробелов в список.
43         if width > 0:
44             lst.append(' ' * width)
45     # Вывести новое предложение, объединив все элементы списка lst.
46     print(''.join(lst))

```

Рисунок 6. Код программы примера 3 (часть 2)

```

/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Examples/example3.py
Введите предложение: Привет Мир
Введите длину: 20
Привет          Мир

Process finished with exit code 0

```

Рисунок 7. Результат работы программы

4. Выполнил индивидуальное задание №1: Дано предложение. Вывести «столбиком» его третий, шестой и т. д. Символы

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  sentence = input("Введите предложение: ")
5  count = 1
6
7  for char in sentence:
8      if count % 3 == 0:
9          print(char)
10     count += 1

```

Рисунок 8. Код программы индивидуального задания 1

```
/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Individual/individ1.py
Введите предложение: Привет Мир сегодня я родился
и
т
и
с
о
я

д
с

Process finished with exit code 0
```

Рисунок 9. Результат работы программы

5. Выполнил индивидуальное задание №2: Дано предложение. Определить, есть ли в нем буквосочетания чу или шу. В случае положительного ответа найти также порядковый номер первой буквы первого из них

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  sentence = input("Введите предложение: ")
5  count = 1
6
7  for i in range(len(sentence)):
8      if sentence[i] == 'щ' or sentence[i] == 'ч' or sentence[i] == 'ш' or sentence[i] == 'ц':
9          if sentence[i + 1] == 'у':
10             print('Есть буквосочетание чу или шу')
11             print('Порядковый номер первого буквосочетания -', count)
12             break
13     count += 1
```

Рисунок 10. Код программы индивидуального задания 2

```
/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Individual/individ2.py
Введите предложение: Мы поймали щук
Есть буквосочетание чу или шу
Порядковый номер первого буквосочетания - 12

Process finished with exit code 0
```

Рисунок 11. Результат работы программы

6. Выполнил индивидуальное задание №3: Дано слово, оканчивающее символом «.». Составить программу, которая вставляет некоторую заданную букву после буквы с заданным номером

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  word = input("Введите слово с точкой в конце: ")
5  replacer = 'a'
6  for i in range(len(word)):
7      if i == 5:
8          print('Новое слово:', word[:5] + replacer + word[5:])
```

Рисунок 12. Код программы индивидуального задания 3

```
/usr/bin/python3.11 /home/code_raider/git/Python_LW6/Individual/individ3.py
Введите слово с точкой в конце: Россия.
Новое слово: Россияa.
Process finished with exit code 0
```

Рисунок 13. Результат работы программы

Ответы на вопросы

1. Строка в Python — это последовательность символов с неизменяемым типом данных
2. Существуют три способа задания строк:
 - Одинаковые кавычки: string = 'Пример строки'
 - Двойные кавычки: string = „Пример строки“
 - Тройные кавычки (для многострочных строк): string = ```Пример строки```
3. Существуют такие операции как + (конкатенация), * (повторение), len() (длина строки), str() (преобразование в строку)
4. Индексация строк схожа с индексацией массивов — первый элемент строки будет иметь индекс 0
5. Срезы задаются с использованием start:stop:step. Например, string[1:5] вернет подстроку с индексами от 1 до 4

6. Строки не могут быть изменены после создания, что обеспечивает их неизменяемость и сохранение данных

7. Для этого можно использовать метод `istitle()`. Пример — `string.istitle()`

8. Для этого есть оператор `in`. Пример — `substring in string`

9. Для нахождения индекса первого вхождения подстроки в строку можно использовать `find()`. Пример — `index = string.find(substring)`

10. Для счета кол-ва символов в строке используется функция `len()`. Пример — `length = len(string)`

11. Для этого можно использовать `count()`. Пример — `string.count(char)`, где `char` — символ

12. F-строки — способ форматирования строк, включая значения переменных. Пример — `f'Привет, {name}'`

13. Чтобы найти подстроку к заданной части строки можно использовать `find()`, указав диапазон индексов: `index = string[start:end].find(substring)`

14. `„Привет, {}!“ .format(name)`

15. Можно использовать функцию `isdigit()`. Пример — `string.isdigit()`

16. Для этого можно использовать метод `split()`

17. Можно воспользоваться методом `islower()`

18. Для этого снова может помочь метод `islower()`

19. Нет, строки не являются численными переменными. Чтобы это сделать нужно для начала преобразовать строку

20. Для этого можно использовать отрицательный срез. Пример — `string2 = string1[::-1]`

21. Можно использовать метод `join()`

22. Для этого есть методы `upper()` и `lower()`

23. Можно использовать одновременно срез и метода `upper()`

24. Можно использовать метод `isupper()`

25. Метод `splitlines()` предназначен для разделения строк по символам новой строки

- 26. Можно использовать метод `replace()`
- 27. Для этого есть методы `startswith()` и `endwith()`
- 28. Можно использовать метод `isspace()`
- 29. Строка буде повторена три раза
- 30. Можно использоваться метод `title()`
- 31. Метод `partition()` разбивает строку на 3 части по первому вхождению заданного разделителя, возвращая кортеж: (before, separator, after)
- 32. Данный метод используют, когда необходимо найти подстроки в строке с возвратом их индекса вхождения

Вывод: В данной работе были изучены методы работы со строками и получены навыки на их основе с помощью языка программирования Python версии 3.x