

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»

Выполнила:
Кубанова Ксения Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: работа со строками в языке Python

Цель: приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Пример 1.

Дано предложение. Все пробелы в нем заменить символом «_».

Код:

```
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(f"Предложение после замены: {r}")
```

Рисунок 1 – пример 1

Результат выражение:

```
Введите предложение: хай вау жесьть
Предложение после замены: хай_вау_жесьть
```

Рисунок 2 – вывод примера 1

Решение примера 1 предоставлено в файле *pr1.py*.

Пример 2.

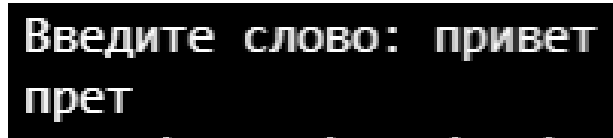
Дано слово. Если его длина нечетная, то удалить среднюю букву, в противном случае – две средние буквы.

Код:

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

Рисунок 3 – код примера 2

Результат:



```
Введите слово: привет
прет
```

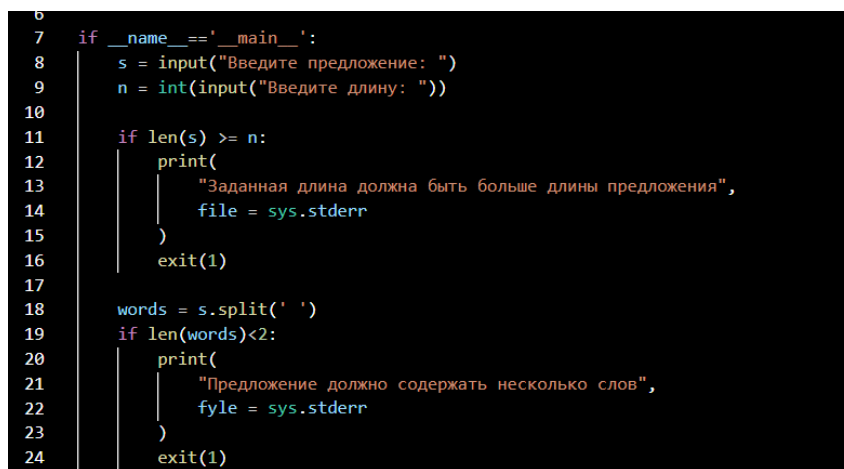
Рисунок 4 – вывод примера 2

Решение примера 2 предоставлено в файле *pr2.py*.

Пример 3.

Дана строка текста, в котором нет начальных и конечных пробелов. Необходимо изменить ее так, чтобы длина строки 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.

Код:



```
6
7 if __name__ == '__main__':
8     s = input("Введите предложение: ")
9     n = int(input("Введите длину: "))
10
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     words = s.split(' ')
19     if len(words) < 2:
20         print(
21             "Предложение должно содержать несколько слов",
22             file = sys.stderr
23         )
24         exit(1)
```

Рисунок 5 – код примера 3 часть 1

```

25     delta = n
26     for word in words:
27         delta -= len(word)
28
29     w, e = delta // (len(words) - 1), delta%(len(words)-1)
30
31     lst=[]
32
33     for i, word in enumerate(words):
34         lst.append(word)
35
36     if i < len(words)-1:
37
38         width = W
39         if r > 0:
40             width += 1
41             r -= 1
42         if width > 0:
43             lst.append(' ' * width)
44     print(''.join(lst))

```

Рисунок 6 – код примера 3 часть 2

Результат:

```

Введите предложение: аоаоыю аоуа
Введите длину: 6
Заданная длина должна быть больше длины предложения
PS C:\Users\Student\Desktop\K\Программирование на pyth\2.
hon "c:/Users/Student/Desktop/K/Программирование на pyth/
py"
Введите предложение: приургпк прук
Введите длину: 100
приургпкпрук

```

Рисунок 7 – вывод примера 3

Решение примера 3 предоставлено в файле *pr3.py*.

Индивидуальное задание 1.

Даны два слова (первое длиннее второго). Заменить во втором слове соответствующее количество символов на первое слово.

Код:

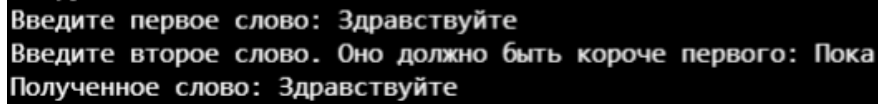
```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      s1=input("Введите первое слово: ")
8      s2=input("Введите второе слово. Оно должно быть короче первого
9      if len(s2) >= len(s1):
10         print(
11             "Заданная длина должна быть короче длины первого введё
12             file=sys.stderr
13         )
14         exit(1)
15     r=s2.replace(s2, s1)
16     print(f"Полученное слово: {r}")

```

Рисунок 8 – код индивидуального задания 1

Результат:



```
Введите первое слово: Здравствуйте
Введите второе слово. Оно должно быть короче первого: Пока
Полученное слово: Здравствуйте
```

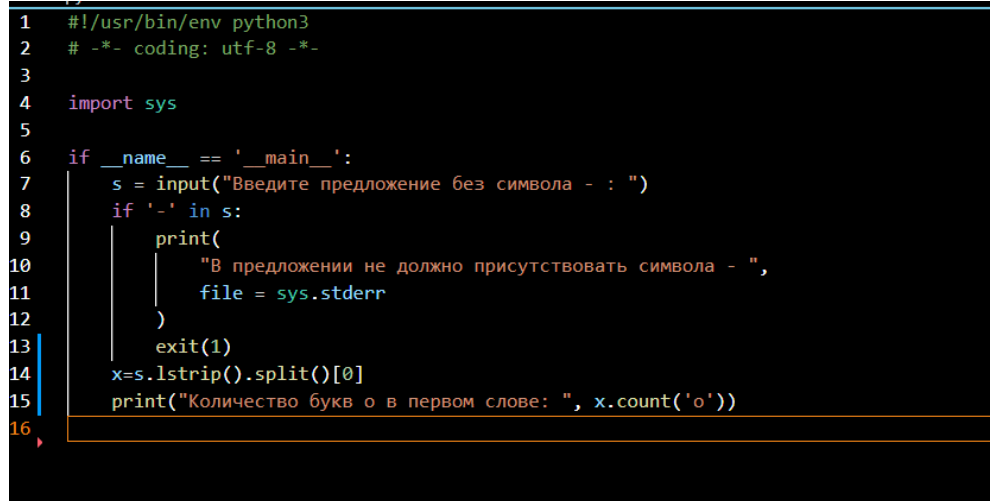
Рисунок 9 – вывод индивидуального задания 1

Решение индивидуального задания 1 предоставлено в файле *ind1.py*.

Индивидуальное задание 2.

Дано предложение, в котором нет символа «-». Определить количество букв о в первом слове. Учесть, что в начале предложения могут быть пробелы.

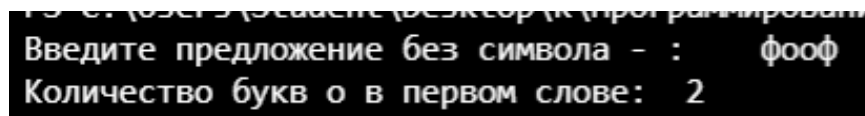
Код:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      s = input("Введите предложение без символа - : ")
8      if '-' in s:
9          print(
10             "В предложении не должно присутствовать символа - ",
11             file = sys.stderr
12         )
13         exit(1)
14     x=s.lstrip().split()[0]
15     print("Количество букв о в первом слове: ", x.count('o'))
16
```

Рисунок 10 – код индивидуального задания 2

Результат:



```
Введите предложение без символа - :   фООФ
Количество букв о в первом слове:  2
```

Рисунок 11 – вывод индивидуального задания 2

Решение индивидуального задания 2 предоставлено в файле *ind2.py*.

Индивидуальное задание 3.

Дана строка. Удалить из нее все пробелы.

Код:

```

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(f"Предложение с удалёнными пробелами: {r}")

```

Рисунок 12 – код индивидуального задания 3

Результат:

```

PS C:\Users\Student\Desktop\К\Программирование\1\
Введите предложение: пр в ф пмвырш
Предложение с удалёнными пробелами: првфпмвырш
PS C:\Users\Student\Desktop\К\Программирование\1\

```

Рисунок 13 – вывод индивидуального задания 3

Решение индивидуального задания 3 предоставлено в файле *ind3.py*.

Задание повышенной сложности:

Даны два слова. Для каждой буквы первого слова определить, входит ли она во второе слово. Повторяющиеся буквы первого слова не рассматривать. Например, если заданные слова процессор и информация, то для букв первого из них ответом должно быть: нет да да да нет нет.

Код:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      s1=input("Введите первое слово: ")
6      s2=input("Введите второе слово: ")
7
8      for _, item in enumerate(s1):
9          if item in s2:
10             print("да")
11          else:
12             print("нет")
13

```

Рисунок 14 – код усложнённого задания

Результат:

```

Введите первое слово: вов
Введите второе слово: вав
да
нет
да
PS C:\Users\Student\Desktop\К\

```

Рисунок 15 – вывод усложнённого задания

Решение задания повышенной сложности предоставлено в файле *hardmode.py*.

Контрольные вопросы

1. Что такое строки в языке Python?

Строка на Python представляет собой последовательность символов, заключенных в одинарные, двойные или тройные кавычки. Она может содержать буквы, цифры, символы и пробелы.

2. Какие существуют способы задания строковых литералов в языке Python?

Одинарные кавычки, двойные кавычки, форматирование строки, многострочные строки, экранированные символы.

3. Какие операции и функции существуют для строк?

+ — оператор конкатенации строк.

* — оператор создает несколько копий строки.

Оператор `in` возвращает `true` , если подстрока входит в строку, и `false` , если нет.

`chr()` - Преобразует целое число в символ

`ord()` - Преобразует символ в целое число

`len()` - Возвращает длину строки

`str()` – Изменяет тип объекта на `string`

4. Как осуществляется индексирование строк?

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. Как осуществляется работа со срезами для строк?

Python также допускает возможность извлечения подстроки из строки, известную как “string slice”. Если `s` это строка, выражение формы `s[m:n]` возвращает часть `s`, начинающуюся с позиции `m`, и до позиции `n`, но не включая позицию.

6. Почему строки Python относятся к неизменяемому типу данных?

Каждый объект в Python имеет уникальный идентификационный номер. Строки, как неизменяемый тип данных, имеют постоянный идентификационный номер, что означает, что их значение не может быть изменено после создания.

Строки в Python относятся к неизменяемым типам данных. Это означает, что после создания строки нельзя изменить её содержимое. Вместо этого создается новая строка с измененным значением.

7. Как проверить то, что каждое слово в строке начинается с заглавной буквы?

`istitle()`.

8. Как проверить строку на вхождение в неё другой строки?

`Find()` или `in`.

9. Как найти индекс первого вхождения подстроки в строку?

Для поиска индекса первого вхождения подстроки в строку на Python можно воспользоваться методом `find()`.

10. Как подсчитать количество символов в строке?

`Len()`.

11. Как подсчитать то, сколько раз определённый символ встречается в строке?

`Count()`.

12. Что такое f-строки и как ими пользоваться?

F-строки (f-strings) - это способ форматирования строк в Python, который позволяет встраивать значения переменных и выражений непосредственно в строку.

13. Как найти подстроку в заданной части строки?

`string.index(<sub>[, <start>[, <end>]])` ищет в строке заданную подстроку.

14. Как вставить содержимое переменной в строку, воспользовавшись методом `format()`?

Имеет вид:

```
'That {} was {}'.format(t, d)
```

t и d - переменные

15. Как узнать о том, что в строке содержатся только цифры?

`string.isdigit()` определяет, состоит ли строка из цифр (проверка на число).

16. Как разделить строку по заданному символу?

`split()` - метод разбивает строку на подстроки с использованием разделителя и возвращает список подстрок.

17. Как проверить строку на то, что она составлена только из строчных букв?

`islower()`. Этот метод возвращает True, если все буквы в строке являются строчными.

18. Как проверить то, что строка начинается со строчной буквы?

`islower()`. Этот метод возвращает True, если первый символ строки является строчной буквой.

19. Можно ли в Python прибавить целое число к строке?

Да, в Python можно прибавить целое число к строке. Однако, для этого необходимо явно преобразовать целое число в строку с помощью функции `str()`.

20. Как «перевернуть» строку?

Для "переворачивания" строки в Python можно воспользоваться срезами.

21. Как объединить список строк в одну строку, элементы которой разделены дефисами?

Чтобы объединить список строк в одну строку, элементы которой разделены дефисами, можно использовать метод `join()` в Python.

22. Как привести всю строку к верхнему или нижнему регистру?

Чтобы привести всю строку к верхнему регистру, можно использовать метод `upper()`, а для приведения к нижнему регистру — метод `lower()`.

23. Как преобразовать первый и последний символы строки к верхнему регистру?

Для этого можно воспользоваться срезами строк.

24. Как проверить строку на то, что она составлена только из прописных букв?

Для проверки того, что строка состоит только из прописных букв, можно воспользоваться методом `islower()`.

25. В какой ситуации вы воспользовались бы методом `splitlines()` ?

Метод `splitlines()` используется для разделения строки на подстроки по символам новой строки. Это может быть полезно, например, при чтении текстовых файлов, где строки разделены символами новой строки.

26. Как в заданной строке заменить на что-либо все вхождения некоей подстроки?

Для замены всех вхождений некоей подстроки на другую подстроку в заданной строке на Python, можно воспользоваться методом `replace()`.

27. Как проверить то, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов?

Для проверки того, что строка начинается с заданной последовательности символов, или заканчивается заданной последовательностью символов, можно использовать методы `startswith()` и `endswith()` соответственно.

28. Как узнать о том, что строка включает в себя только пробелы?

Для проверки того, что строка включает в себя только пробелы, можно воспользоваться методом `isspace()`.

29. Что случится, если умножить некую строку на 3?

Если умножить некую строку на 3, то строка будет повторена три раза и результат будет строкой, содержащей три копии исходной строки.

30. Как привести к верхнему регистру первый символ каждого слова в строке?

Для приведения к верхнему регистру первого символа каждого слова в строке на Python, можно воспользоваться методом `title()`

31. Как пользоваться методом `partition()` ?

Метод `partition()` используется для разделения строки на три части по первому вхождению указанной подстроки.

32. В каких ситуациях пользуются методом `rfind()` ?

Метод `rfind()` используется для поиска последнего вхождения указанной подстроки в строке.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе со строками при написании программ языка Python версии 3.x.