

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 №2.3**  
**дисциплины «Программирование на Python»**

Выполнил:  
Дзуев Альберт Мухаметович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., доцент кафедры  
инфокоммуникаций

---

(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:**

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Проработал примеры лабораторной работы:

```
ython\3\prim1.py"
Введите предложение: Лабораторная работа номер 6 по дисциплине "Программирование на Python"
Предложение после замены: Лабораторная_работа_номер_6_по_дисциплине_"Программирование_на_Python"
```

Рисунок 1. Вывод программы примера 1

```
ython\3\prim2.py"
Введите слово: Привет
Прет
```

Рисунок 2. Вывод программы примера 2

```
Введите предложение: Текстовое предложение для примера 3
Введите длину: 36
Текстовое предложение для примера 3
```

Рисунок 3. Вывод программы примера 3

3. Выполнил индивидуальное задание 1 вариант 9: Дано предложение. Вывести «столбиком» его третий, шестой и т. д. символы.

### **Код программы:**

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
if __name__ == '__main__':
    sentence = input("Введите предложение на русском языке: ")
    if len(sentence) < 3:
        print("Введите предложение длиной минимум 3 символа")
    else:
        for i in range(2, len(sentence), 3):
            print(sentence[i])
```


```
ython\3\ind1.py"
Введите предложение на русском языке: ПРИВЕТ
И
Т
```

Рисунок 4. Вывод программы индивидуального задания 1

4. Выполнил индивидуальное задание 2 вариант 9: Дано предложение. Определить, есть ли в нем буквосочетания чу или щу. В случае положительного ответа найти также порядковый номер первой буквы первого из них.

**Код программы:**

```
if __name__ == '__main__':  
    sentence = input("Введите предложение: ").lower()  
    position_chu = sentence.find('чу')  
    position_shu = sentence.find('щу')  
    if position_chu != -1 and (position_chu < position_shu or position_shu == -1):  
        print(f"Буквосочетание 'чу' найдено в позиции: {position_chu + 1}")  
    elif position_shu != -1 and (position_shu < position_chu or position_chu == -1):  
        print(f"Буквосочетание 'щу' найдено в позиции: {position_shu + 1}")  
    else:  
        print("Буквосочетание 'чу' и 'щу' не найдено.")
```



```
Введите предложение: Прерасное чувство  
Буквосочетание 'чу' найдено в позиции: 11
```

Рисунок 5. Вывод программы индивидуального задания 2

5. Выполнил индивидуальное задание 3 вариант 9: Дано слово, оканчивающееся символом «.». Составить программу, которая вставляет некоторую заданную букву после буквы с заданным номером.

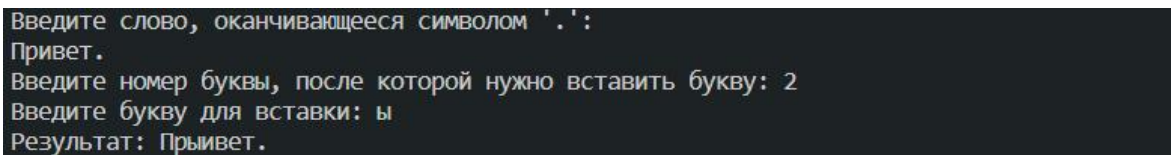
**Код программы:**

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
import sys  
if __name__ == '__main__':  
    word = input("Введите слово, оканчивающееся символом '!\n")  
    if not word.endswith('!'):  
        print("Ошибка: Входное слово не оканчивается символом '!',  
              file=sys.stderr)  
        exit(1)  
    position = int(input("Введите номер буквы, после которой нужно вставить  
букву: "))  
    if position <= 0 or position > len(word):  
        print("Некорректная позиция", file=sys.stderr)  
        exit(1)
```

```

else:
letter = input("Введите букву для вставки: ")
# Вставляем заданную букву после буквы с заданным номером
new_word = word[:position] + letter + word[position:]
print("Результат:", new_word)
exit(1)

```



```

Введите слово, оканчивающееся символом '.' :
Привет.
Введите номер буквы, после которой нужно вставить букву: 2
Введите букву для вставки: ы
Результат: Привет.ы

```

Рисунок 6. Вывод индивидуального задания 3

6. Выполнил задание повышенной сложности вариант 9: Даны два слова. Определить, можно ли из букв первого из них получить второе. Рассмотреть два варианта: повторяющиеся буквы второго слова могут в первом слове не повторяться; каждая буква второго слова должна входить в первое слово столько же раз, сколько и во второе.

#### Код программы:

```

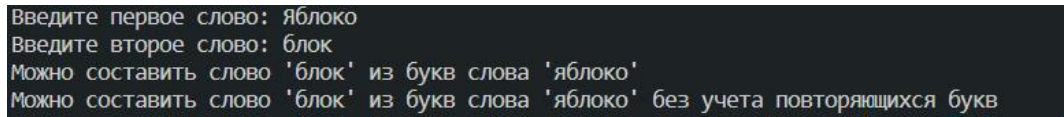
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
def can_spell(word1, word2):
# Проверка, что каждая буква из второго слова встречается в первом слове не
более раза
counts1 = {char: word1.count(char) for char in set(word1)}
counts2 = {char: word2.count(char) for char in set(word2)}
for char in counts2:
if char not in counts1 or counts1[char] < counts2[char]:
return False
return True
def letter_repeat(word1, word2):
# Проверка, что можно составить второе слово из первого без учета
повторений букв
for char in word2:
if char not in word1:
return False
return True
word1 = input("Введите первое слово: ").lower()
word2 = input("Введите второе слово: ").lower()
if can_spell(word1, word2):
print(f"Можно составить слово '{word2}' из букв слова '{word1}'")
else:

```

```

print(f'Нельзя составить слово '{word2}' из букв слова '{word1}''')
if letter_repeat(word1, word2):
    print(f'Можно составить слово '{word2}' из букв слова '{word1}' без учета
    повторяющихся букв")
else:
    print(f'Нельзя составить слово '{word2}' из букв слова '{word1}' без учета
    повторяющихся букв")

```



```

Введите первое слово: Яблоко
Введите второе слово: блок
Можно составить слово 'блок' из букв слова 'яблоко'
Можно составить слово 'блок' из букв слова 'яблоко' без учета повторяющихся букв

```

Рисунок 7. Вывод программы задания повышенной сложности

### Ответы на контрольные вопросы:

1. Что такое строки в языке Python?

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. Какие существуют способы задания строковых литералов в языке Python?

Строки в апострофах и в кавычках, экранированные последовательности - служебные символы, «сырые» строки - подавляют экранирование, строки в тройных апострофах или кавычках.

3. Какие операции и функции существуют для строк?

Оператор сложения (+), умножения (\*), принадлежности подстроки (in).

Функции:

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.

Существует еще один вариант синтаксиса среза, о котором стоит упомянуть. Добавление дополнительного «:» и третьего индекса означает шаг, который указывает, сколько символов следует пропустить после извлечения каждого символа в срезе.

#### 6. Почему строки Python относятся к неизменяемому типу данных?

Строки – один из типов данных, которые Python считает неизменяемыми, что означает невозможность их изменять. На самом деле нет особой необходимости изменять строки. Обычно можно легко сгенерировать копию исходной строки с необходимыми изменениями.

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

string.istitle() определяет, начинаются ли слова строки с заглавной буквы.

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

s.find(<sub>) возвращает первый индекс в s который соответствует

началу строки <sub> , сели же в s нет <sub>, то функция выдаст -1

9. Как найти индекс первого вхождения подстроки в строку?  
s.find(<sub>) возвращает первый индекс в s который соответствует началу строки <sub> , сели же в s нет <sub>, то функция выдаст -1

10. Как подсчитать количество символов в строке?

len(s) возвращает количество символов в строке s.

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

s.count(<sub>) возвращает количество точных вхождений подстроки <sub> в s.

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

В Python версии 3.6 был представлен новый способ форматирования строк. Эта функция официально названа литералом отформатированной строки, но обычно упоминается как f- строки (f-string).

Возможности форматирования строк огромны и не будут подробно описана здесь.

Одной простой особенностью f-строк, которые вы можете начать использовать сразу, является интерполяция переменной. Вы можете указать имя переменной непосредственно в f-строковом литерале (f'string'), и python заменит имя соответствующим значением.

Пример: print(f"Произведение {n} на {m} равно {prod}"), где m, n, prod это переменные.

13. Как найти подстроку в заданной части строки?

s.find(подстрока, начало, конец).

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

```
print('{}'.format(s)).
```

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

`s.isdigit()` возвращает `True` когда строка `s` не пустая и все ее символы являются цифрами, а `False` если нет.

16. Как разделить строку по заданному символу?

```
str.split('заданный символ').
```

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

`s.islower()` возвращает `True`, если строка `s` не пустая, и все содержащиеся в ней буквенные символы строчные, а `False` если нет. Не алфавитные символы игнорируются.

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

`S[0].islower()` выдаст `True` если строка начинается со строчной буквы и `False` если нет.

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

Нет.

20. Как «перевернуть» строку?

```
s[::-1].
```

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

```
str.join('-', s), где s – это список строк.
```



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

`s.upper(), s.lower()`.

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

`string[0].upper() + string[1:-1] + string[-1].upper()`

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

`s.isupper()`.

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

`splitlines()` делит `s` на строки и возвращает их в список. Любой из следующих символов или последовательностей символов считается границей строки: `\n`, `\r`, `\r\n`, `\v` или же `\x0b`, `\f` или же `\x0c`, `\x1c`, `\x1d`, `\x1e`, `\x85`, `\u2028`, `\u2029`.

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

`s.replace(old, new)`.

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

`str.startswith()` и `str.endswith()`.

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

`s.isspace()`.

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

Она напечатается 3 раза.

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

`s.title()`.

31. Как пользоваться методом `partition()`?

Разбивает строку при первом появлении строки аргумента и возвращает кортеж, содержащий часть перед разделителем, строку аргумента и часть после разделителя.

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

`rfind()` и `find()` оба используются для поиска вхождения подстроки в строку, но есть различие в том, что `rfind()` ищет справа налево (с конца строки), в то время как `find()` ищет слева направо (с начала строки). То есть `rfind()` находит последнее вхождение, а `find()` первое вхождение подстроки в строку.

**Вывод:** в результате выполнения работы были приобретены навыки по работе со строками при написании программ с помощью языка программирования Python версии 3.x.