Тема 8 РАБОТА СО СТРОКАМИ

Практическая работа №6

8.1 Цель практической работы

Приобретение практических навыков разработки и программирования вычислительного процесса с использованием строковых переменных.

8.2 Теоретические сведения

Строка в Python - это объект класса **str**. В темах ранее, для того чтобы работать с числовыми данными, применялись функции приведения типов, например, **int**, как это продемонстрировано в следующем примере:

```
ind=int(input(" \n Введите индекс элемента "))
```

На рисунке показано, что строки индексируются с нуля, то есть, если имеется оператор **stroka="python"**, то первый символ в строке нулевой и обращение к нему будут записываться как **stroka[0]**. В то же время к элементам строки в Python может обращаться, указывая отрицательные индексы, например, оператор **print(stroka[-6])** есть ничто иное, как вывод символа **p** на экран.

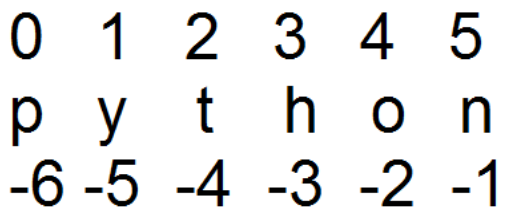

Рисунок 8.1 – Индексация строки

Известно, что в программировании, прежде чем сравнить один символ с другим, его следует преобразовать в число с помощью таблицы ASCII (American Standard Code for Information Interchange - Американский стандартный код для обмена информацией), который поддерживает кодирование 128 буквенно-цифровых символов. При работе со строками важно знать, что символ "**A**" - это совсем не то же самое, что символ "**a**". Длина строки ограничена лишь объемом оперативной памяти компьютера, и для обращения к элементу строки достаточно указать его индекс в квадратных скобках.

Строки в Python представляют собой неизменяемую последовательность и обрабатываются с использованием цикла с оператором

for. Важно помнить, что изменить символ строки, обратившись к нему по индексу, невозможно.

```
stroka="Python"
stroka[0]="p"  #недопустимый оператор
```

При обработке строки к ней можно обратиться непосредственно в цикле, как это показано в листинге ниже.

```
stroka="python"
for i in stroka:
    print(i, end=" ")
```

Результатом работы этого кода станет вывод на экран слова **p y t h o n**.

Функция len(). *При работе со строками* может оказаться полезной функция **len()**, назначение которой заключается в определении длины строки.

Функция ord(). Функция - **ord()**, синтаксис которой **ord("Символ")** возвращает код указанного символа.

Функция chr(). Действие функции, синтаксис которой **chr(Число)**, прямо противоположно функции **ord()**, а именно, по ASCII-коду символа вернуть символ.

Метод upper(). Синтаксис метода: **stroka.upper()**. Преобразует все символы строки к верхнему регистру.

Метод lower(). Синтаксис метода: **stroka.lower()**. Преобразует все символы строки к нижнему регистру.

Метод swapcase(). Синтаксис метода: **stroka. swapcase()**. Преобразует все символы строки, записанные в нижнем регистре - в верхний и наоборот.

Метод capitalize(). Синтаксис метода: **stroka.capitalize()**. Преобразует первую букву в строке в верхний регистр, а остальные в нижний регистр.

Метод title().

Синтаксис метода: **stroka.title()**. Преобразует, все первые буквы в строке в верхний регистр, а остальные в нижний регистр.

Метод startswith(). Синтаксис метода: **stroka.startswith(podstroka)**. Проверяет, начинается ли строка **stroka** с указанной подстроки (фрагмента строки) **podstroka**.

Метод endswith(). Синтаксис метода: **stroka.endswith(podstroka)**. Проверяет, заканчивается ли строка **stroka** указанной подстрокой **podstroka**.

Метод replace(). Синтаксис метода: **stroka.replace(old, new)**, где **old** - подстрока для замены, **new** - новая подстрока. Метод находит и заменяет в строке **stroka** подстроку **old** подстрокой **new**.

Метод rfind(). Синтаксис метода: **stroka.rfind(podstr)**, где **podstr** - подстрока. Метод возвращает позицию последнего вхождения подстроки в строку. Если **подстрока не обнаружена**, то возвращается значение **-1**. После параметра **podstr** могут быть указаны начальная позиция и конечная позиции

в строке, где следует искать подстроку. Эти параметры необязательны, и если отсутствует начальная позиция, то поиск будет осуществлен с начала строки.

Метод find(). Синтаксис метода: **stroka.flnd(podstr)**, где **podstr** - подстрока. В отличие от предыдущего метода, метод **find()** возвращает позицию первого вхождения подстроки в строку. Если **подстрока не обнаружена**, то возвращается значение **-1**.

Метод count(). Синтаксис метода: **stroka.count(podstr)**, где **podstr** - подстрока. Метод возвращает количество вхождений подстроки **podstr** в строку **stroka**.

Метод strip(). Синтаксис метода: **stroka.strip()**. Метод удаляет начальные и конечные пробелы в строке **stroka**.

Методы lstrip() и rstrip(). Их синтаксис и действие аналогичны рассмотренному методу **strip()**, с той разницей, что метод **lstrip()** удаляет символы пробела слева от начала исходной строки, а метод **rstrip()** - в конце исходной строки. С помощью методов **strip()**, **rstrip()** и **lstrip()** можно удалять не только пробелы. Так, если в качестве параметра одного из методов указать символ или последовательность символов, то произойдет его (их) удаление.

Метод split(). Синтаксис метода: stroka. **split().** Метод разделяет строку на подстроки и добавляет их в список. Если в качестве параметра присутствует разделитель, то строка будет разбита на подстроки в соответствии с указанным разделителем. В случае его отсутствия разделителем считается пробел.

Метод join(). Синтаксис метода: **разделитель.join(spisok)**, где **spisok** - список или кортеж. Еще одним способом преобразования строки в список является использование функции **list()**, а обратное преобразование осуществляется методом]**join()**. Такие преобразования необходимы для того, чтобы изменить элемент строки по индексу, поскольку изначально строки неизменяемы.

8.3 Пример выполнения задания на практическую работу

Определение количества символов.

Задача 8.3.1. Пользователь вводит исходную строку **stroka** и символ **simvol**. Подсчитайте, сколько раз заданный символ встречается в исходной строке.

Решение. Первоначально после обнуления ячейки **k**, которая будет играть роль счетчика символов, мы преобразуем все символы исходной строки в строчные буквы методом **lower()**. Затем в цикле происходит сравнение текущего символа строки **spisok[i]** с искомым символом **simvol**. Параметр цикла **i** будет изменяться от **0** (начало строки) до конца строки (за это отвечает ячейка **n**, в которой уже находится значение функции

len(spisok)). Подсчет количества найденных символов в строке обеспечивает оператор **k+=1**.

Ниже приведен код программы, отвечающий за решение задачи.

```
k=0
stroka=input("\n Введите строку: ")
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\n Введите символ для поиска его повторений в списке: ")
n=len(spisok)
for i in range(0, n):
    if spisok[i]==simvol:
        k+=1
print("\n", k)
```

Результат работы программы представлен на рисунке 8.2.

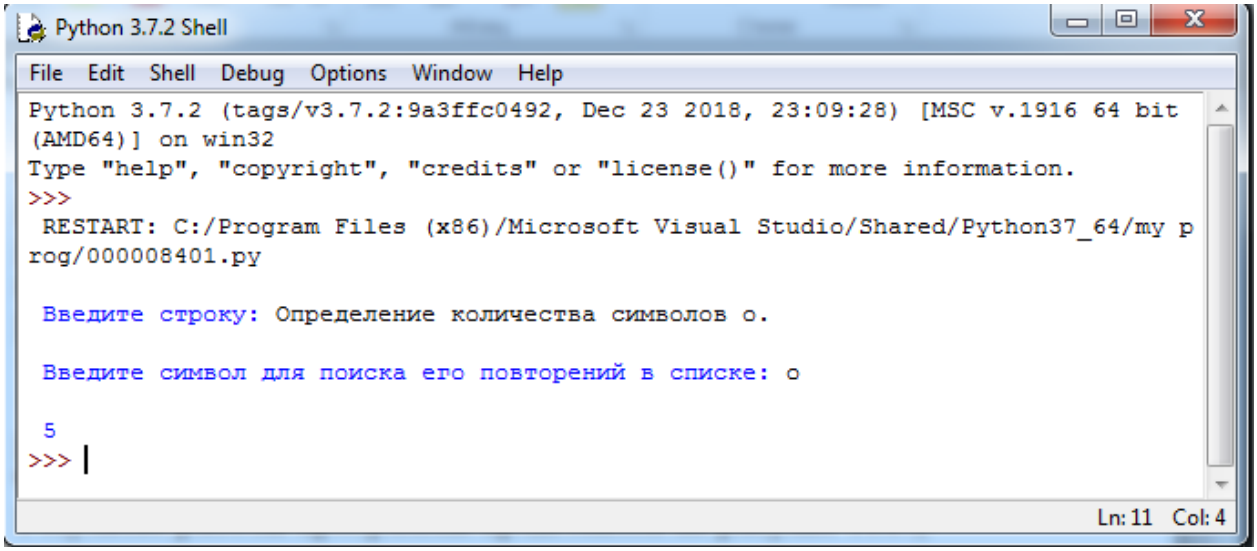

Рисунок 8.2 – Результат работы программы по определению количества символов в строке

Замена символов в строке.

Задача 8.3.2. Пользователь вводит исходную строку **stroka** и символ **simvol**. Замените пробелы в исходной строке указанным символом.

Решение. Ранее при объяснении работы метода **join()** была решена подобная задача. В листинге приведен код, в котором обрабатываемая строка не фиксирована, а вводится с клавиатуры. Методы ее обработки остались прежними. Заметим, что подобную задачу можно решить и с применением метода **replace().**

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
```

```
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\n Введите символ для замены им пробелов: ")
n=len(stroka1)
for i in range(0, n):
    if spisok[i]==" ":
        spisok[i]=simvol
stroka1="".join(spisok)
print("\n", stroka1)
```

Результат работы программы представлен на рисунке 8.3.

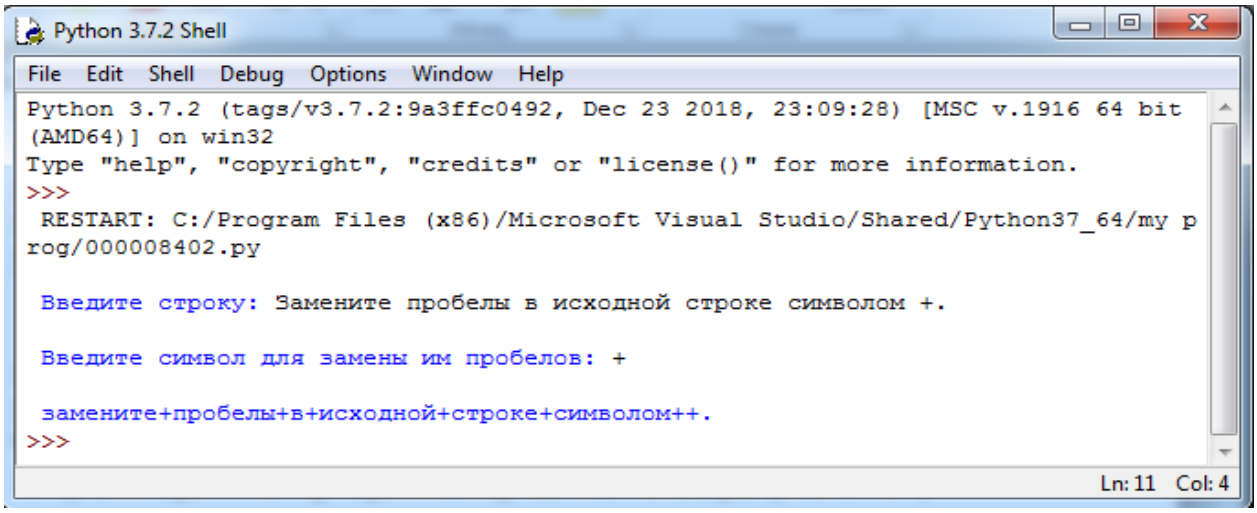

Рисунок 8.3 – Результат работы программы по замене пробелов в исходной строке символом +

Удаление символов в строке.

Задача 8.3.3. Пользователь вводит исходную строку **stroka** и символ **simvol**. Удалите в исходной строке указанный символ.

Решение. Способ основан на применении метода **replace()**. Осуществляем ввод символа, который хотим удалить, а затем используем его в качестве параметра в методе **replace()**. Второй параметр в методе делаем равным пустым кавычкам.

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
simvol=input("\n Введите символ для удаления его из строки: ")
stroka=stroka.replace(simvol, "")
print("\n", stroka)
```

Результат работы программы представлен на рисунке 8.4.

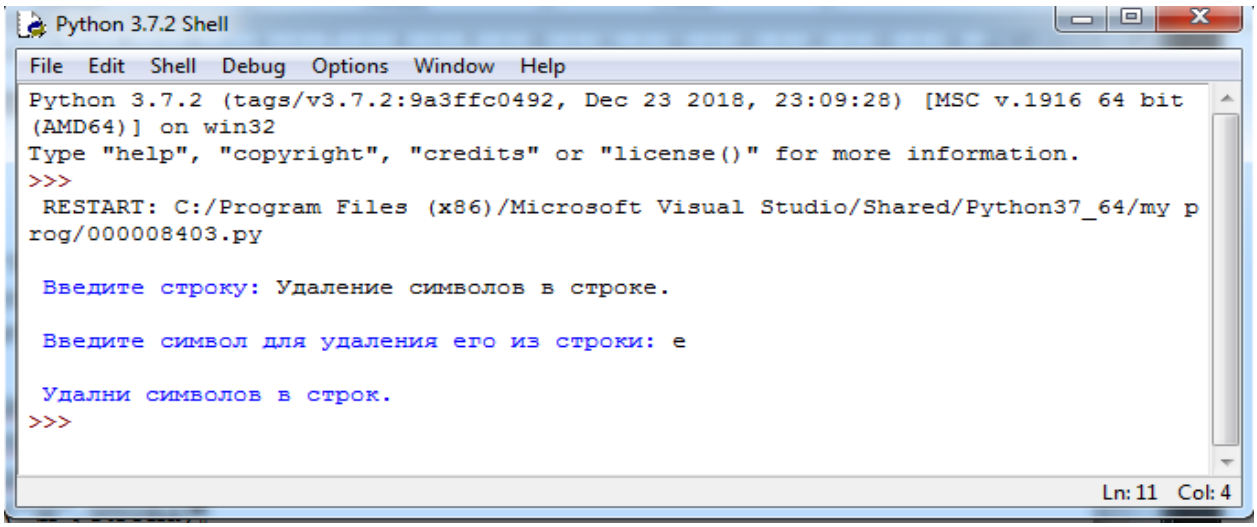

Рисунок 8.4 – Результат работы программы по удалению введенного символа в исходной строке

Вставка символа в строку.

Задача 8.3.4. Пользователь вводит исходную строку **stroka.** В исходную строку необходимо добавить символ **simvol1** после символа **simvol**, которые пользователь вводит с клавиатуры.

Решение. Вставка подстроки в заданную позицию осуществляется с помощью оператора **spisok.insert(i+l, simvol1)**, предварительно мы преобразовали строку в список с помощью функции **list**. Изменение параметра **i** в цикле с оператором **for** обеспечивает продвижение по строке. Если очередной элемент списка равен найденному элементу (**if spisok[i]== simvol**), то применяется метод **insert**. В конце программы, используя метод **join**, делаем обратное преобразование: список превращается в строку, которая и выводится на экран. Важно помнить, что при работе со строками надо учитывать, в каком состоянии находится символ - в верхнем или нижнем регистре.

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
print(stroka)
spisok=list(stroka)
simvol=input("\n Введите символ, после которого вставим введенный ниже символ: ")
simvol1=input("\n Введите символ для вставки его в строку: ")
n=len(stroka)
k=0
for i in range(0, n):
    if spisok[i]==simvol:
        k+=1
for i in range(0, n+k):
    if spisok[i]==simvol:
```

```
        spisok.insert(i+1, simvol1)
    stroka="".join(spisok)
    print("\n", stroka)
```

Результат работы программы представлен на рисунке 8.5.

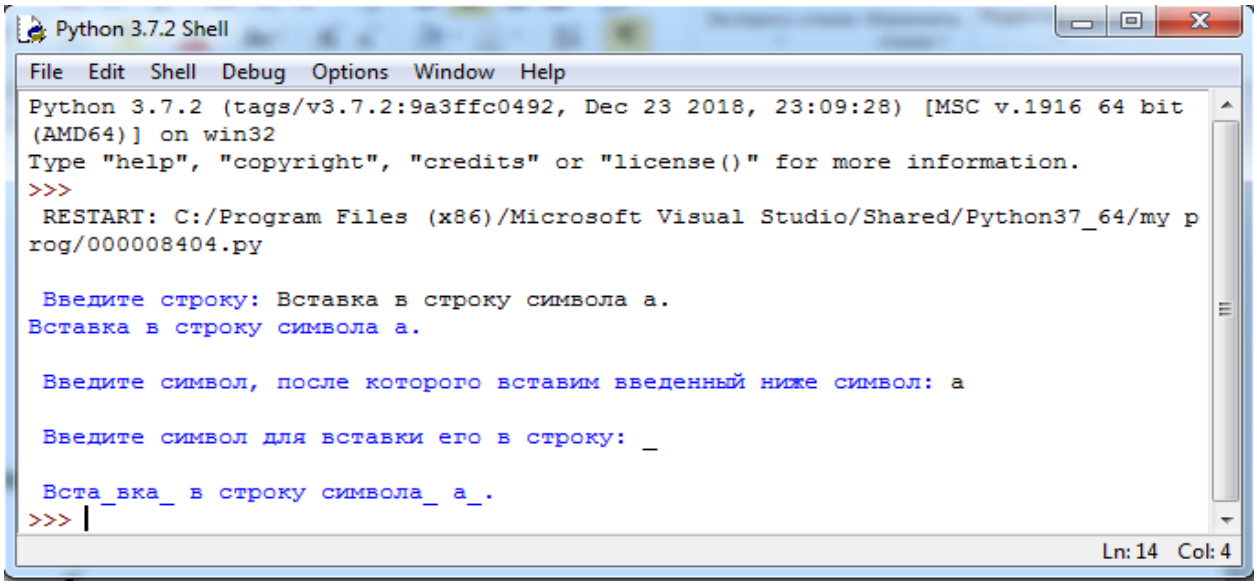

Рисунок 8.5 – Результат работы программы по вставке введенного символа "_" после символа "a" в исходную строку

Анализ символа на принадлежность к группе.

Задача 8.3.5. Пользователь вводит исходную строку **stroka.** Определить количество символов **simvol** и **simvol1** в исходной строке. Символы **simvol1** и **simvol** пользователь вводит с клавиатуры.

Решение. В данной программе, преобразовав строку символов в список, вводим два символа, количество которых мы хотим обнаружить в исходной строке. В цикле с оператором **for** сравниваем каждый элемент списка с введенными пользователем символами, и если условие истинно, счетчик увеличивается на единицу.

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
print(stroka)
spisok=list(stroka)
simvol=input("\n Введите первый символ, количество которого необходимо найти: ")
simvol1=input("\n Введите второй символ, количество которого необходимо найти: ")
k1=0
k2=0
k=0
```

```
n=len(stroka)
for i in range(0, n):
    if spisok[i]==simvol:
        k+=1
        k1+=1
    if spisok[i]==simvol1:
        k+=1
        k2+=1
print("\n Всего найденных символов", simvol," = ", k1)
print("\n Всего найденных символов", simvol1," = ", k2)
print("\n Всего найденных символов", simvol, " и ", simvol1, " = ", k)
```

Результат работы программы представлен на рисунке 8.6.

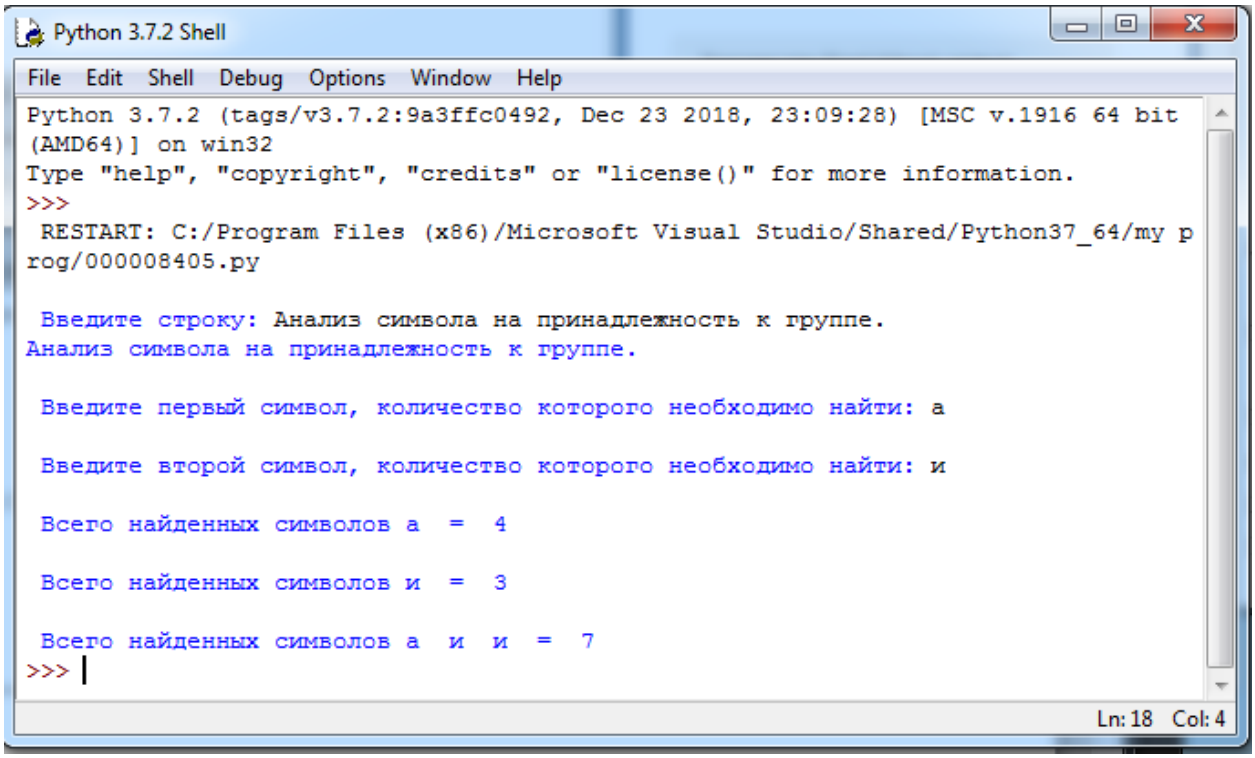

Рисунок 8.6 – Результат работы программы по анализу символов на принадлежность к группе

Обращение строки.

Задача 8.3.6. Пользователь вводит исходную строку **stroka.** Необходимо перевернуть строку, то есть записать символы в обратном порядке (последние становятся первыми и наоборот).

Решение. При решении данной задачи можно было бы воспользоваться методом **reverse** для обработки списков, но алгоритм работы данной программы основан на поочередной выборке символов из заданной строки и перемещение их в начало новой строки. Вспомогательная строка **tmp** изначально пуста. С помощью организации цикла просматриваем символы

исходной строки от первого к последнему. Каждый из них присоединяется к началу уже накопленной строки, оператором **tmp=stroka[i]+tmp**.

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
print("\n", stroka)
tmp=""
n=len(stroka)
for i in range(0, n):
    tmp=stroka[i]+tmp
print("\n Строка наоборот: ", tmp)
```

Результат работы программы представлен на рисунке 8.7.

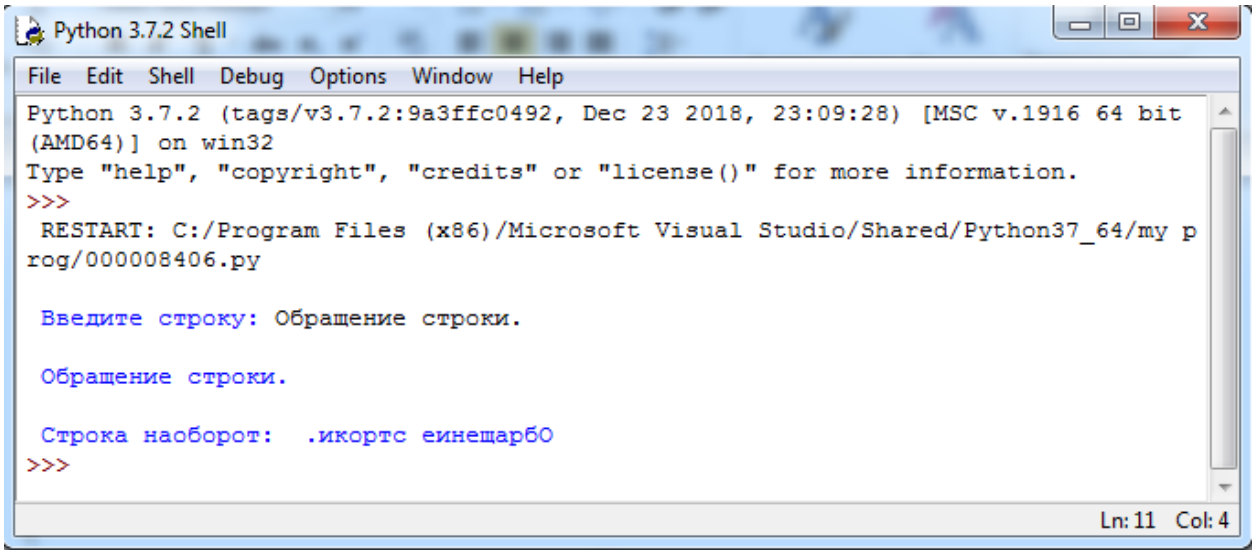

Рисунок 8.7 – Результат работы программы по обращение строки

Алфавитная выборка.

Задача 8.3.7. Имеется заранее заданный алфавит. Пользователь вводит исходную строку **stroka.** Выберите из строки символы, используемые в алфавите, и выведите их на экран.

Решение. При решении данной задачи мы познакомимся с таким понятием, как «константа» в Python. Переменная, набранная прописными буквами, в изучаемом языке программирования называется **константой**. Существует особенность применения констант в Python: в отличие от многих языков программирования константы можно изменять. Поэтому, создав константу, пользователь должен контролировать ее неизменность.

Итак, имеем некий набор символов, условно называемый алфавитом, который хранится в константе **ALFAVIT**. Пользователь вводит строку, и для каждого символа строки (**for** letter **in** stroka:) мы осуществляем проверку условия. Если символ есть в алфавите (**if** letter **in** ALFAVIT:), то в переменной **tmp** накапливаем результирующие символы оператором **tmp+=letter**.

Ниже приведен код программы, отвечающий за решение задачи.

```
ALFAVIT="abcdgh"
stroka=input("\n Введите строку: ")
print("\n", stroka)
tmp=""
for letter in stroka:
    if letter in ALFAVIT:
        tmp+=letter
print(tmp)
```

Результат работы программы представлен на рисунке 8.8.

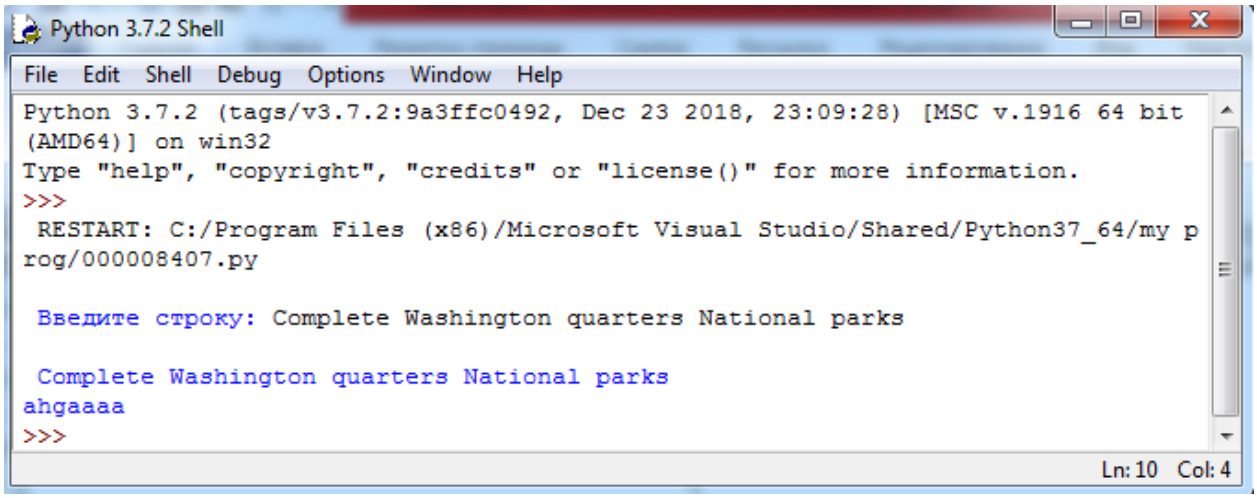

Рисунок 8.8 – Результат работы программы алфавитной выборки

Срезы строк.

Задача 8.3.8. Из исходной строки получите символы, расположенные между заданными начальным и конечным значениями.

Решение. При изучении темы по работе с кортежами, уже было знакомство с понятием «срез» и говорилось о том, что срез кортежа получается в результате вывода элементов кортежа, находящихся между заранее заданными начальной (**a**) и конечной (**b**) позициями элементов. Механизм работы со срезами строк очень похож на принципы работы со срезами кортежей. Применяя срезы к строкам, мы можем выбрать из них любой символ или последовательность расположенных подряд символов. Для этого нам понадобятся начальная и конечная позиции, которые будут обозначать границы среза.

Ниже приведен код программы, отвечающий за решение задачи.

```
stroka=input("\n Введите строку: ")
print("\n", stroka)
flag=None
while flag!="":
```

```
        flag=input("\n Введите начало среза или для выхода введите ENTER:
")
        if flag:
            flag=int(flag)
            kon=input("\n Введите конец среза: ")
            kon=int(kon)
            print(stroka[flag:kon])
    input("\n Введите ENTER для выхода ")
```

Результат работы программы представлен на рисунке 8.9.

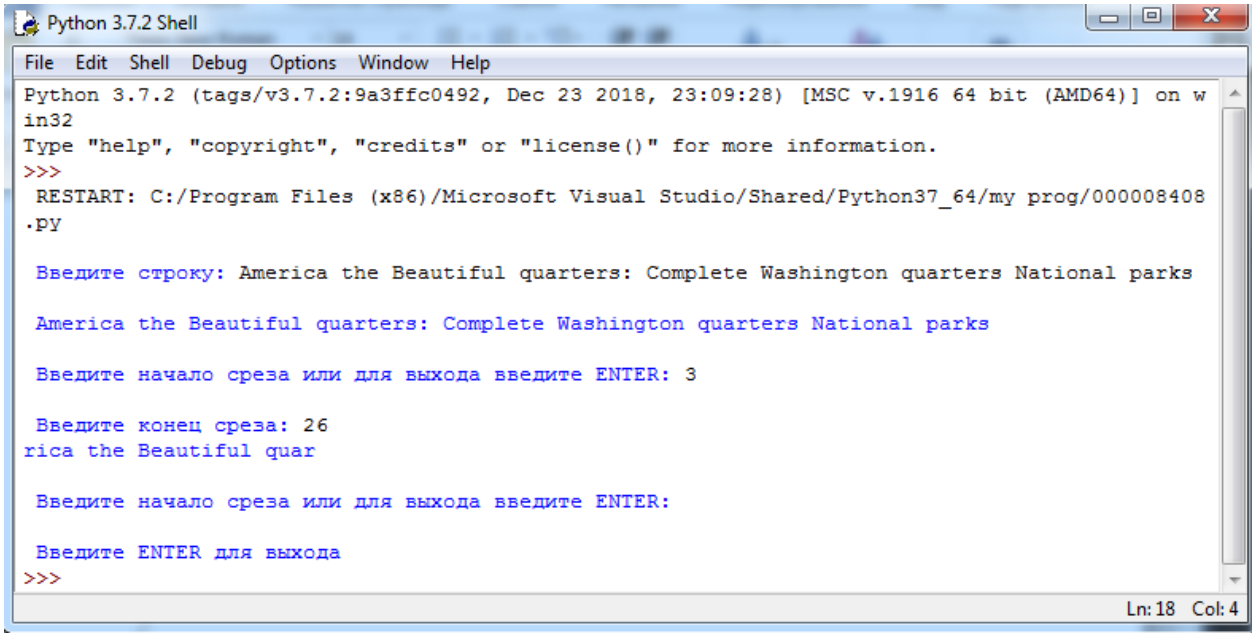

Рисунок 8.9 – Результат работы программы среза строки

8.4 Задание на практическую работу

Реализовать задачи для самостоятельного решения и одну задачу для индивидуального решения согласно варианту**.**

8.4.1 Задачи для самостоятельного решения.

8.4.1.1. Разработайте программу, которая зашифровывает введенный с клавиатуры текст. Процесс шифровки производится следующим образом: из десятичного кода каждого введенного с клавиатуры символа вычитается число десять. Получившаяся в результате вычитания величина интерпретируется как десятичный код некоторого другого символа, который и выводится на экран компьютера.

8.4.1.2. Подсчитать самую длинную последовательность подряд идущих букв «р». Преобразовать ее, заменив точками все восклицательные знаки.

8.4.2 Задачи для индивидуального решения.

№ вар.	Задание
1	Разработайте программу, которая проверяет, является ли введенная с клавиатуры последовательность символов целым числом, записанным в двоичной системе счисления.
2	Разработайте программу, которая вычисляет среднюю длину слов во введенной с клавиатуры строке.
3	Дано слово. Определите, является ли оно палиндромом (словом, которое читается одинаково в обоих направлениях, например «потоп»).
4	Дана строка символов. Определите самое длинное слово в строке и количество слов такой же длины.
5	Дана строка символов. Определите количество слов, являющихся записью десятичного числа.
6	Дана строка символов. Удалите из нее все пробелы.
7	Дана строка символов. Дано слово. Удалите из строки это слово.
8	Дана строка символов. Выделите подстроку между первой и второй точкой.
9	Дана строка символов. Определите длину самого короткого и самого длинного слова.
10	Дана строка символов. Определите, сколько слов начинается и кончается одной и той же буквой.
11	Дана строка символов. Определите, сколько слов содержат две буквы «с».
12	Дана строка символов. Определите, является ли она правильным скобочным выражением. Рассматривайте только круглые скобки.
13	Дана строка символов. Определите, сколько слов содержат ровно три буквы «е».
14	Строка содержит только цифры. Удалите все впереди стоящие нули.
15	Дана строка символов. Подсчитайте количество знаков препинания в строке.
16	Дана строка символов. Удалить из строки все запятые.

17	Дана строка символов. Приведено некоторое слово. Вставьте его после каждого пробела.
18	Дана строка символов. Найдите сумму чисел, встречающихся в строке.
19	Дана строка символов. Удалите из строки все числа.
20	Дана строка символов. Удалите из строки слово, имеющее наибольшую длину.
21	Дана строка символов. Вставьте в строку пробел после каждого знака препинания.
22	Дана строка символов. Сформируйте строку, состоящую из слов исходной строки, записанных наоборот.
23	Дана строка из цифр и латинских букв. Определите, каких букв, гласных (A, E, I, O, и) или согласных, больше в этой строке.
24	Из заданной строки удалите те ее части, которые заключены в кавычки (вместе с кавычками).
25	Задано слово. Замените в этом слове букву A на букву O. Если буквы A в этом слове нет, то выведите соответствующее сообщение.

8.5 Требования к оформлению отчета по практической работе

При оформлении отчета по практической работе рекомендуется следующая структура и последовательность элементов:

– титульный лист;
– название лабораторной работы;
– цель лабораторной работы;
– индивидуальное задание (по вариантам) на лабораторную работу;
– краткие комментарии по выполнению индивидуального задания и структурная схема алгоритма решения задачи;
– необходимый программный код индивидуального задания; – результаты работы программы; – выводы.

Индивидуальное задание на лабораторную работу содержит полный текст индивидуального задания, полученного у преподавателя, описание алгоритма выполнения индивидуального задания, структурную схему алгоритма решения задачи.

Необходимый программный код индивидуального задания содержит полный текст кода программы, разработанный студентом.

Результаты работы программы обычно содержат копии окон работы программы.

8.6 Контрольные вопросы для защиты лабораторной работы

8.6.1. Как называется кодировка, поддерживающая кодирование буквенно-цифровых символов? Расскажите о ее структуре.

8.6.2. Перечислите основные функции для работы с символами. Приведите примеры.

8.6.3. Перечислите методы работы со строками, позволяющие преобразовывать символы строки к различным регистрам клавиатуры.

8.6.4. Какой метод позволяет разбить строку на подстроки? Напишите его синтаксис.

8.6.5. Какой метод отвечает за преобразование строки в список? Напишите его синтаксис.

8.6.6. Приведите примеры базовых алгоритмов строк.

8.6.7. Каким образом можно осуществить срез строки?