**Строки**

**Строки**

Строки в языке Python являются объектами типа *str*. Переменной можно присвоить значение, являющееся строкой, с помощью двойных или одинарных кавычек:

S = "Hello"

T = 'Hello'

Строка считывается со стандартного ввода функцией input():

S = input()

Напомним, что для строк определены операции сложения (конкатенация) и умножения строки на число.

>>>S = 'Hello' +'World'

HelloWorld

>>>S = 'A' \* 50

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Узнать количество символов (длину строки) можно при помощи функции len():

>>>S = 'Hello'

>>>**print**(len(S))

5

**Обращение по индексу**

Для получения одного символа строки используется обращение по индексу. **Индексом** называется номер символа в строке.

Например, S[i] — это символ строки, который имеет индекс i, при этом нумерация начинается с числа 0. То есть, если S = 'Hello', то S[0] == 'H', S[1] == 'e', S[2] == 'l', S[3] == 'l', S[4] == 'o'.

Если мы хотим обратиться к последнему элементу некоторой строки, то нужно сначала узнать ее длину, после чего вычесть из полученного значения 1. Для нахождения длины используется функция len(S). Таким образом, последний элемент строки будет возвращаться функцией S[len(S) - 1]. Однако данная конструкция достаточно громоздка, вместо неё можно использовать отрицательные значения индексов.

Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с -1. То есть S = 'Hello', то S[-1] == 'o', S[-2] == 'l', S[-3] == 'l', S[-4] == 'e', S[-5] == 'H'.

Проиллюстрируем индексы строки с помощью таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Строка S | H | e | l | l | o |
| Индекс | S[0] | S[1] | S[2] | S[3] | S[4] |
| Индекс | S[-5] | S[-4] | S[-3] | S[-2] | S[-1] |

Если же номер символа строки S больше либо равен len(S), или меньше чем -len(S), то при обращении к этому символу строки произойдет ошибка **IndexError: string index out of range**.

**Задача. Количество букв 'a' в строке**

Дана строка, необходимо посчитать количество букв 'a' в ней.

**Решение**

count = 0

S = input()

**for** i **in** range(len(S)):

 **if** S[i] == 'a':

 count += 1

**print**(count)

Это решение можно немного упростить:

count = 0

S = input()

**for** c **in** S:

 **if** c == 'a':

 count += 1

**print**(count)

**Срезы**

**Срезы**

**Срез** (*slice*) — извлечение из данной строки некоторого фрагмента подстроки или подпоследовательности символов.

Срез можно создать таким образом: S[a : b]. Такой срез возвращает подстроку из b - a символов, начиная с символа c индексом a до символа с индексом b, не включая его. В этом случае a должно быть меньше b, иначе срез вернёт пустую строку. Будем рассматривать в качестве примера строку S = 'Abracadabra'. Для нее срез S[2 : 7] == 'racad'.

Можно использовать как положительные, так и отрицательные индексы в одном срезе. Например, S[1 : -1] — это строка без первого и последнего символа (срез начинается с символа с индексом 1 и заканчивается индексом -1, не включая его). Для рассмотренной выше строки срез S[1 : -1] == 'bracadabr', а срез S[-6 : -2] == 'adab'.

При использовании срезов ошибка **IndexError** никогда не возникает. Например, срез S[1 : 100000] вернёт строку 'bracadabra'.

Можно не указывать второй параметр (но поставить двоеточие), тогда срез берётся до конца строки. Например, чтобы удалить из строки первый символ, можно взять срез S[1:]. Аналогично можно опустить первый параметр, тогда срез S[:5] берётся от начала строки до указанного параметра 5, не включая его. Срез S[:] совпадает с самой строкой S.

**Срезы с тремя параметрами**

Пусть нам необходимо взять срез, в котором символы будут идти не подряд, а с некоторым шагом d. Тогда срез будет иметь три параметра: S[a : b : d]. Третий параметр задаёт шаг, как и в случае с функцией range(), то есть будут взяты символы с индексами a, a + d, a + 2 \* d и так далее. Так, для рассмотренной выше строки S[2 : 9 : 2] вернёт строку 'rcdb'.

Кроме того, параметры можно опускать. Например, срез S[ : : 2] вернёт символы с чётными индексами, а срез S[1 : : 2] — с нечётными. Кроме того, можно взять значение шага среза, равным отрицательному числу, тогда символы будут идти в обратном порядке. Например, срез S[9 : 4 : -1] == 'rbada', а S[ : : -1] == 'arbadacarbA'.

**Методы find и rfind**

**Метод** — это функция, применяемая к объекту, например, к строке или числу. Метод вызывается в виде *ИмяОбъекта.ИмяМетода(параметры)*.

**Метод find и rfind**

Рассмотрим методы find и rfind для работы со строками.

Метод find находит в данной строке подстроку, которая передаётся в качестве параметра. Функция возвращает индекс первого (cамого левого) вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение -1. С помощью этого метода можно проверять содержится ли подстрока в строке. Например:

>>>S = 'Abracadabra'

>>>**print**(S.find('abr'))

7

>>>**print**(S.find('Abr'))

0

>>>**print**(S.find('aa'))

-1

Аналогично, метод rfind возвращает индекс последнего вхождения данной строки («поиск справа»).

>>>S = 'Abracadabra'

>>>**print**(S.rfind('a'))

10

>>>**print**(S.rfind('br'))

8

Если вызвать метод find с тремя параметрами S.find(T, a, b), то поиск будет осуществляться в срезе S[a : b]. Если указать только два параметра S.find(T, a), то поиск будет осуществляться в срезе S[a : ], то есть начиная с символа с индексом a и до конца строки. Отметим, что метод S.find(T, a, b) возвращает индекс в строке S, а не индекс относительно начала среза. Например:

>>>S = ’Abracadabra’

>>>**print**(S.find(’a’, 2, 8))

3

>>>**print**(S.find(’b’, 2))

8

**Методы count и replace**

**Метод count**

Подсчитывает количество вхождений одной строки в другую строку. Простейшая форма вызова S.count(T) возвращает число вхождений строки T внутри строки S. При этом подсчитываются только непересекающиеся вхождения, например:

>>>S = 'Abracadabra'

>>>**print**(S.count('a'))

4

>>>**print**(S.count('ra'))

2

>>>S = 'aaaaaaa'

>>>**print**(S.count('aa'))

3

При указании трёх параметров S.count(T, a, b), будет выполнен подсчёт числа вхождений строки T в срез S[a : b].

**Метод replace**

Метод replace заменяет все вхождения одной строки на другую и возвращает полученную. При этом исходная строка не изменяется. Формат: S.replace(old, new) — заменить в строке S все вхождения подстроки old на подстроку new. Пример:

>>>S = 'Abracadabra'

>>>t = S.replace('ra', 'e')

>>>**print**(t)

'Abecadabe'

>>>**print**(S.replace('a', 'A'))

'AbeAcAdAbe'

C помощью метода replace можно изменить саму строку и удалить в ней все символы, равные некоторому, следующим образом:

>>>S = 'Abracadabra'

>>>S = S.replace('a', '')

>>>**print**(S)

'Abrcdbr'

Если методу replace задать третий параметр num, то заменены будут не все вхождения, а только не больше чем первые num из них.

S.replace(old, new, num)

Например:

>>>S ='Abracadabra'

>>> S.replace('a', '', 2)

'Abrcdabra'

Точно так же как метод count, метод replace работает с непересекающимися шаблонами.

>>>S = 'aa'

>>>S.replace('a', 'aa')

'aaaa'

>>>S = 'aaaaa'

>>>S.replace('aa', 'a')

'aaa'