

Строки

Строка считывается со стандартного ввода функцией `input()`. Напомним, что для двух строк определена операция сложения (конкатенации), также определена операция умножения строки на число.

Строка состоит из последовательности символов. Узнать количество символов (длину строки) можно при помощи функции `len`:

```
>>> s = 'Hello'
>>> print(len(s))
5
```

Срезы (slices)

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

Есть три формы срезов. Самая простая форма среза: взятие одного символа строки, а именно, `S[i]` — это срез, состоящий из одного символа, который имеет номер `i`, при этом считая, что 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'`.

Номера символов в строке (а также в других структурах данных: списках, кортежах) называются *индексом*.

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

Или в виде 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 в срезе 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`.

Срез с двумя параметрами: `S[a:b]` возвращает подстроку из `b-a` символов, начиная с символа с индексом `a`, то есть до символа с индексом `b`, не включая его. Например, `S[1:4]=='ell'`, то же самое получится если написать `S[-4:-1]`. 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).

При использовании такой формы среза ошибки `IndexError` никогда не возникает. Например, срез `S[1:5]` вернет строку `'ello'`, таким же будет результат, если сделать

второй индекс очень большим, например, `S[1:100]` (если в строке не более 100 символов).

Если опустить второй параметр (но поставить двоеточие), то срез берется до конца строки. Например, чтобы удалить из строки первый символ (его индекс равен 0, то есть взять срез, начиная с символа с индексом 1), то можно взять срез `S[1:]`, аналогично если опустить первый параметр, то срез берется от начала строки. То есть удалить из строки последний символ можно при помощи среза `S[:-1]`. Срез `S[:]` совпадает с самой строкой `S`.

Если задать срез с тремя параметрами `S[a:b:d]`, то третий параметр задает шаг, как в случае с функцией `range`, то есть будут взяты символы с индексами `a`, `a+d`, `a+2*d` и т.д. При задании значения третьего параметра, равному 2, в срез попадет каждый второй символ, а если взять значение среза, равное `-1`, то символы будут идти в обратном порядке.

Методы

Метод - это функция, применяемая к объекту, в данном случае - к строке. Метод вызывается в виде **Имя_объекта.Имя_метода(параметры)**. Например, `S.find("e")` — это применение к строке `S` метода `find` с одним параметром `"e"`.

Метод `find` и `rfind`

Метод `find` находит в данной строке (к которой применяется метод) данную подстроку (которая передается в качестве параметра). Функция возвращает индекс первого вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение `-1`. Например:

```
>>> S = 'Hello'
>>> print(S.find('e'))
1
>>> print(S.find('ll'))
2
>>> print(S.find('L'))
-1
```

Аналогично, метод `rfind` возвращает индекс последнего вхождения данной строки (“поиск справа”).

```
>>> S = 'Hello'
>>> print(S.find('l'))
2
>>> print(S.rfind('l'))
3
```

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. Метод `S.find(T, a, b)` возвращает индекс в строке `S`, а не индекс относительно

Метод `replace`

Метод `replace` заменяет все вхождения одной строки на другую. Формат: `S.replace(old, new)` — заменить в строке `S` все вхождения подстроки `old` на подстроку `new`. Пример:

```
>>> 'Hello'.replace('l', 'L')
'HeLLo'
```

Если методу `replace` задать еще один параметр: `S.replace(old, new, count)`, то заменены будут не все вхождения, а только не больше, чем первые `count` из них.

```
>>> 'AbraKadabra'.replace('a', 'A', 2)
'AbRAkAdabra'
```

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:

```
>>> 'Abracadabra'.count('a')
4
>>> ('a' * 100000).count('aa')
50000
```

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