СТРОКИ

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. Чтобы определить длину строки **s** можно воспользоваться функцией **len(s)** - она возвращает целое число, равное длине строки.

Строки

Почти любой другой объект в Питоне можно привести к строке, которая ему соответствует. Для этого нужно вызвать функцию str(), передав ей в качестве параметра объект, переводимый в строку.

Строка – неизменяемый объект! Пытаясь изменить строку, вы просто создаете новый объект, отличающийся от старого.

Строки

Как вы думаете, что напечатает программа?

```
a = '234'
b = a
a = '123‘
print(b)
```

Операторы принадлежности

- **in** – считается истиной (true), если находит переменную в заданной строке, и ложью (false) в противном случае;
- **not in** – считается истиной (true), если не находит переменную в заданной строке, и ложью (false) в противном случае.

```
S = 'Hello'
if S in ‘Helloween‘: # вернёт True
print (‘YES’)
```

Срезы строк

Срез (slice) – это способ извлечь из строки отдельные символы или подстроки. При применении среза конструируется новая строка, а строка, к которой был применён срез, остается без изменений.

Срезы строк

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 считается, 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'.

Срезы строк

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

В языке Питон присутствует и нумерация символов строки отрицательными числами. Последний символ строки имеет номер -1, предпоследний -2 и так далее. При попытке обратиться к символу с номером, меньшим чем -len(s), или большим, чем len(s) – 1 возникает ошибка.

Срезы строк

Нумерация символов в строке "Python" представлена в таблице:

P	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

Получить доступ, например, к символу **o**, можно двумя способами s[4] и s[-2].

Срезы строк

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] == ‘yth', 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).

Срезы строк

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

Если первый параметр находится правее второго, то будет сгенерирована пустая строка.

Срезы строк

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 кажый второй символ, а если взять значение среза, равное -1, то символы будут идти в обратном порядке. Например, можно перевернуть строку срезом **S[::-1]**.

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"**.

Методы find и rfind

Метод **find** находит в 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.

Методы find и rfind

```
S = 'Hello‘
print(S.find('e')) # вернёт 1
print(S.find('lo')) # вернёт 3
print(S.find('L')) # вернёт -1
```

Методы find и rfind

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

```
S = 'Hello‘
print(S.find('l')) # вернёт 2
print(S.rfind('l')) # вернёт 3
```

Методы find и rfind

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 среза.

Методы find и rfind

Часто возникает задача найти и вывести все вхождения подстроки в строку, включая накладывающиеся. Например, для строки **'ABABA'** и подстроки **'ABA'** ответ должен быть 0, 2. Ее решение выглядит так:

Методы find и rfind

```
string = input()
substring = input()
pos = string.find(substring)
while pos != -1:
    print(pos)
    pos = string.find(substring, pos + 1)
```

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**

```
print('Hello'.replace('l', 'L')) # вернёт 'HeLLo'
```

Метод replace

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.

```
print('Abrakadabra'.replace('a', 'A', 2))
# вернёт '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:

```
print('Abracadabra'.count('a')) # вернёт 4
print(('aaaa').count('aa')) # вернёт 2
```

Метод count

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

```
print('Abracadabra'.count('a‘, 0, 6)) # вернёт 2
```

Несколько пробелов

```
s = input()
i = 0
while s[i] == ' ':
    i += 1
s = s[i:]
i = len(s)
while s[i-1] == ' ':
    i -= 1
s = s[:i]
```

Другие методы строк

S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.istitle()	Начинаются ли слова в строке с заглавной буквы

Другие методы строк

S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str
S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.lstrip([chars])	Удаление пробельных символов в начале строки
S.rstrip([chars])	Удаление пробельных символов в конце строки
S.strip([chars])	Удаление пробельных символов в начале и в конце строки