Строка (str). Методы строки.

Программирование в лингвистике.

Верещагина Анна Дмитриевна

Строка в Python

str – последовательность символов.
str неизменяема, но итерируема (ее элементы можно перебрать по частям).

Итерируем:

```
[18] string = "кот"
     for sym in string:
       print(sym)

     к
     о
     т
```

Попробуем изменить символ в строке:

```
string = "кот"
string[0] = "в"

---------------------------------------------------------------------
TypeError                                 Traceback (most recent
<ipython-input-12-0583f8633c5a> in <cell line: 2>()
      1 string = "кот"
----> 2 string[0] = "в"

TypeError: 'str' object does not support item assignment
```

Вместо этого:

```
string = "кот"
print("в" + string[1:])

вот
```

Закавычивание строк

'Hello' '''Hello''' VS. "Hello" """Hello"""

Одинаковое количество кавычек с обеих сторон (двойных или одиночных) говорит Python, что перед нами строка

Апострофы:

```
'Kitty's fur'
```

```
  File "<ipython-input-19-8de6dea8a63e>"
    'Kitty's fur'
           ^
SyntaxError: unterminated string literal
```

```
"Kitty's fur"
```

```
'Kitty's fur'
```

Переносы:

```
print("""Смешной
кот""")
```

```
Смешной
кот
```

```
print("Смешной\nкот")
```

```
Смешной
кот
```

Комментарии:

```
'''Сколько будет 2 + 2?
Это комментарий.'''
# И предыдущая строка, и эта - комментарии.
2 + 2
```

Многострочный

Однострочный

```
4
```

Символ \ - применение

Экранирование		Спец. последовательности	
\\	Один обратный слэш.	\n	Новая строка.
\'	Один апостроф.	\t	Горизонтальная табуляция.
\"	Одна кавычка.	\r	Возврат каретки.
Обозначения			
\u	16-битовый символ Юникода.	\U	32-битовый символ Юникода.
\x	16-ричное значение	\o	8-ричное значение

В случае экранирования бэкслэш говорит нам, что перед нами – последовательность символов, а не элемент синтаксиса Python.

Длина строки – функция len()

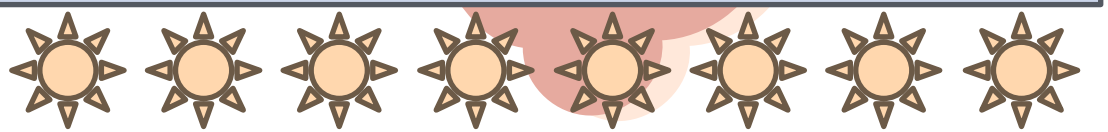
```
s = 'abcdefg'
len(s)
```

```
7
```

```
s.rfind('g')
```

```
6
```

Так как **len()** - это функция, она может применяться как к строкам, так и к другим объектам Python: спискам, множествам и словарям.



Заметим, что индекс последней буквы – не 7, как мы ожидали, а 6. Об этом далее.

Срезы

s = 'abcdefghij'

s[x:y:z]
x – начало, **y** – конец, **z** - шаг

0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	j

Код	Результат		Пояснение
s[0]	'a'	**a**bcdefghij	Возвращает нулевой символ в строке.
s[1:4]	'bcd'	a**bcd**efghij	Возвращает все символы с 1 до 3.
s[1:4:2]	'bd'	a**b**c**d**efghij	Возвращает все символы с 1 до 3 через один.
s[:3]	'abc'	**abc**defghij	Возвращает первые три символа.
s[2::2]	'cegi'	ab**c**d**e**f**g**h**i**j	Возвращает все символы со 2 до конца через один.
s[-1]	'j'	abcdefghi**j**	Возвращает первый символ с конца.
s[:-5:-1]	'jihg'	abcdef**ghij**	Возвращает последние 4 символа в обр. порядке

Нумерация символов всегда начинается с 0. Правый край среза не входит в последовательность.

Префиксы строк

	Пример	Результат	
r (raw)	r'd:\games\nexus'	d:\games\nexus	Префикс r **отменяет экранированные последовательности**. Идеально для путей и регулярных выражений.
b (bytes)	b'abc' b[0] b[1]	b'abc' 97 98	Делает строку последовательностью байтов. Возвращает значение **только при обращении по индексу**. То же самое, что и **ord**.
f (format -ted)	f'{1 + 1} это 2'	2 это 2	Строка с возможностью **вставить** в нее **значение переменной или выражения** в фигурные скобки.

Префиксы можно спокойно сочетать. Например:
rf'{path}\train'

Арифметические операции со строками

```
name = "Вася"
print("Имя: " + name)
```

```
Имя: Вася
```

Сложение

Умножение

```
a = "hello"
print(a * 4)
```

```
hellohellohellohello
```

Как сравниваются строки

```
string1 = "ab"
string2 = "ad"
print(string1 < string2)
```

```
True
```

По сути мы сравниваем последовательно номера символов в нашей кодировке (в Google Colab это Unicode - UTF-8).

Так, 97 (a) == 97 (a),
а вот 98 (b) < 100 (d).
Значит, string2 больше, чем string1.

```
print(ord("a"))
print(ord("b"))
print(ord("d"))
```

```
97
98
100
```

ord и chr – взаимно-обратные функции

```
print(ord("a"))
```

```
97
```

Ord – сокращение от *ordinal* (порядковый номер). Возвращает номер символа в кодировке.

Chr – сокращение от *character* (символ). Возвращает символ по номеру в кодировке.

```
print(chr(97))
```

```
a
```

Функции и методы Python

print() – функция. Она может применяться к объектам разных типов – к примеру, как к **int**, так и к **str**.

Функция print() в документации **очень укороченно** выглядит так:
print(*args, sep='', end='')
*args – неопределенное кол-во аргументов, данное в определенной последовательности.

.strip() – метод. Он применяется только к объектам типа **str**. Это по сути функция, привязанная только к классу **str**.

```
print("abc", 1)
print(" abc ".strip())
number = 1
number.strip()
```

```
abc 1
abc
---------------------------------------
AttributeError
```

Методы строки, возвращающие строки

Метод	Результат	
'abc'.replace('a', 'b') 'abc'.replace('ab', 'cb')	bbc cbc	Заменяет символы. Можно заменить сразу несколько символов.
' abc '.strip() '1abc1'.strip('1')	abc abc	Отрезает приведенные в аргументах символы слева и справа. По умолчанию отрезает все пробельные символы.
'1abc1'.rstrip('1')	1abc	Отрезает символы только справа.
'1abc1'.lstrip('1')	abc1	Отрезает символы только слева.
'aBc'.upper()	ABC	ВЕРХНИЙ РЕГИСТР.
'ABc'.lower()	abc	нижний регистр.
'abc'.capitalize()	Abc	Первая заглавная буква.
'ab c'.title()	Ab C	Все Первые Буквы Заглавные.

Методы строки, возвращающие числа

Метод	Результат	
'abc'.find('b') 'abc'.find('bc')	0 1	Поиск символов с начала. Возвращает индекс первого совпадения. Если не найдено, возвращает -1
'abab'.rfind('a') 'abab'.rfind('ab')	2 2	Поиск символов с конца. Возвращает индекс первого совпадения. Если не найдено, возвращает -1
'abc'.index('a') 'abc'.index('d')	0 ValueError	Работает так же, как find. Используется только в случае, если точно известно, что символ присутствует.
'abab'.count('a') 'abab'.count('ab')	2 2	Считает, сколько раз в строке встречается символ или последовательность символов.

Проверки: возвращают bool

Метод	Результат	
'abc'.startswith('ab')	True	Начинается ли строка с этих символов?
'abc'.endswith('ab')	False	Заканчивается ли строка этими символами?
'ABc'.isupper()	False	Все буквы заглавные?
'abc'.islower()	True	Все буквы строчные?
'Abc B'.istitle()	True	Все первые буквы заглавные?
'1234'.isdigit()	True	Строка состоит только из цифр?
'ab1'.isalpha()	False	Строка состоит только из букв? || ' '.isspace()	True	Строка содержит только пробельные символы?
'ab123'.isalnum()	True	Строка состоит из цифр и/или букв?

Операция “in”

Возвращает bool: есть ли в строке этот символ?

```
if "a" in "abcd":
  print('"a" есть в строке.')
```

```
"a" есть в строке.
```

True

```
print("a" not in "abcd")
```

```
False
```

False

Спасибо за внимание!

admvereshchagina@gmail.com