Тема 5. Строки

Строки в Python – упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме. Длина строки в Python ограничена только «объемом оперативной памяти».

В **Python 3** для строк используется кодировка **Unicode**. Первые 128 символов по таблице **Unicode** такие же, как и в таблице символов **ASCII**.

При объявлении строк их нужно заключать в кавычки. В Python для строк используют как одинарные кавычки (апострофы), так и двойные кавычки, а также и тройные.

Строки в апострофах и кавычках

Строки в апострофах и кавычках – это одно и то же. Используются они для того, чтобы в строку можно было заключать либо одинарные, либо двойные кавычки без экранирования. Если нужно разместить в строке одинарную кавычку, то строку заключаем в двойные и наоборот.

```
s = '"Python"'
print(s)
```

```
"Python"
```

```
s = "'Python'"
print(s)
```

```
'Python'
```

Тройные кавычки

Тройные кавычки позволяют размещать в тексте как одинарные, так и двойные кавычки. Но это не главное их достоинство. Главное достоинство заключается в том, что в тройных кавычках можно расположить многострочный текст, чего нельзя сделать в одинарных и двойных кавычках без использования спецсимволов (**\n** – перевод строки).

```
s = """Текст в
несколько
строчек"""
print(s)
```

```
Текст в
несколько
строчек
```

```
s = '''Текст в
несколько
строчек'''
print(s)
```

```
Текст в
несколько
строчек
```

5.1. Специальные символы

В строке могут присутствовать специальные символы, такие как перенос строки и т. п.

Рассмотрим некоторые из них:

\n – перевод строки
\r – возврат каретки
\t – знак табуляции
\" – двойная кавычка
\' – апостроф
**** – обратный слеш

Все специальные символы в строках **Python** интерпретируются автоматически. Но если все-таки возникает необходимость их вывести именно как символы строки, то их **экранируют** при помощи обратного слеша (\).

Примеры:

```
print('Привет\nPython')
```

```
Привет
Python
```

```
print('Привет\\nPython')
```

```
Привет\nPython
```

```
print('Привет'Python')
```

Получим ошибку.

```
print('Привет\'Python')
```

```
Привет'Python
```

Подавление экранирования

Кроме предложенного выше способа есть еще одна возможность вывести строку без преобразования спецсимволов. Для этого перед строкой (до кавычек) нужно поставить букву **r**. Это позволит вывести строку в неформатированном виде.

Пример:

```
s=r'D:\new\tutor'
print(s)
```

```
D:\new\tutor
```

5.2. Методы и функции работы со строками

Для работы со строками в **Python** создано немало различных методов и функций. Строки относятся к **неизменяемым типам данных**, поэтому практически все методы в качестве значения возвращают новую строку, а не изменяют текущую. **Представим методы в виде таблицы.**

Функция, метод	Описание	Пример
s1+s2	Конкатенация (сложение строк).	s1= 'Привет' s2= 'Python' print(s1+' '+s2) **Получим:** Привет Python
s1*n	Повторение строки **n** раз.	s1='Привет' print(s1*3) **Получим**: ПриветПриветПривет
s[i]	Обращение к символу по индексу. В Python нумерация символов в строке начинается с 0. Можно только обратиться к символу для его получения, а вот изменить символ по номеру нельзя. Аналогично можно использовать и отрицательный индекс, тогда нумерация справа налево, т. е. -1 – это номер последнего символа.	s1='Привет' print(s1[1]) **Получим:** р print(s1[-1]) **Получим:** т **А вот так писать уже нельзя:** s[1]='р' В результате получим ошибку
s[i:j:step]	Извлечение подстроки, срез **i** - номер символа, с которого начинаем копировать **j** - номер символа, до которого копируем, причем j не включено **step** - шаг.	s1='Привет' print(s1[1:3]) **Получим:** ри print(s1[1:6:2]) **Получим:** рвт print(s1[:2]) **Получим:** Пр print(s1[2:]) **Получим:** ивет print(s1[::-1]) **Получим:** тевирП print(s1[-1:]) **Получим:** т print(s1[:-1]) **Получим:** Приве
len(s)	Длина строки (количество символов в строке).	s1='Привет' print(len(s1)) **Получим:** 6
in	Проверяет присутствие подстроки в строке. Если присутствует, возвращает значение True иначе False.	s1='Привет' print('ри' in s1) **Получим:** True
s.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения, если подстрока найдена или -1, если не найдена. Обратите внимание, что start и end это не обязательные параметры, если их не указать, то поиск идет от начала до конца строки. Параметр start включается в диапазон, а end не включается.	s1='Привет' print(s1.find('ри')) **Получим:** 1 s1='Программа' print(s1.find('р',3,9)) **Получим:** 4 s1='Программа' print(s1.find('а',0,3)) **Получим:** -1
s.rfind(str, [start],[end])	Делает то же что и find, но только ищет в обратном направлении.	s1='Привет' print(s1.rfind('ри')) **Получим**: 1
s.replace(шаблон, замена)	Замена шаблона в строке.	s1='Привет' print(s1.replace('т', 'д')) **Получим**: Привед
s.split(символ)	Разбиение строки по символу, разделителю. В результате получаем список из слов.	s1='Привет' print(s1.split('и')) **Получим**: ['Пр', 'вет']

s.isdigit()	Состоит ли строка только из цифр. Если да, то возвращает True иначе False.	s1='a1' print(s1.isdigit()) **Получим**: False s1='21 ' print(s1.isdigit()) **Получим**: True
s.isalpha()	Состоит ли строка только из букв. Если да, то возвращает True иначе False.	s1='a1' print(s1.isalpha()) **Получим**: False s1='ab' print(s1.isalpha()) **Получим**: True
s.isalnum()	Состоит ли строка из цифр или букв. Если да, то возвращает True иначе False.	s1='a1' print(s1.isalnum()) **Получим**: True s1='ab_' print(s1.isalnum()) **Получим**: False
s.islower()	Состоит ли строка из символов в нижнем регистре. Если да, то возвращает True иначе False.	s1='ab' print(s1.islower()) **Получим**: True s1='aB' print(s1.islower()) **Получим**: False
s.isupper()	Состоит ли строка из символов в верхнем регистре. Если да, то возвращает True иначе False.	s1='AB' print(s1.upper()) **Получим**: True s1='aB' print(s1.isupper()) **Получим**: False
s.istitle()	Начинаются ли слова в строке с заглавной буквы. Если да, то возвращает True иначе False.	s1='Иванов Иван Иванович' print(s1.istitle()) **Получим**: True s1='Иванов иван Иванович' print(s1.istitle()) **Получим**: False
s.lower()	Преобразование строки к нижнему регистру.	s1='PYTHON' print(s1.lower()) **Получим**: python
s.startswith(str)	Начинается ли строка **s** с шаблона str.	s1='python' s2='py' print(s1.startswith(s2)) **Получим**: True Если же код будет выглядеть так: s1='Python' s2='py' print(s1.startswith(s2)) **Получим**: False
s.endswith(str)	Заканчивается ли строка **s** шаблоном str.	s1='python' s2='on' print(s1.endswith(s2)) **Получим**: True
s.join(список)	Сборка строки из списка с разделителем **s**.	s=':' print(s.join(['1','2','3','5'])) **Получим**: 1:2:3:4
ord(символ)	Перевод символа в его код ASCII.	s1='P' print(ord(s1)) **Получим**: 80
chr(число)	Перевод кода ASCII в символ.	a=80 print(chr(a)) **Получим**: P
s.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний.	s1='иванов Иван Иванович' print(s1.capitalize()) **Получим**: Иванов иван иванович
s.count(str, [start],[end])	Возвращает количество не пересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию).	s1='Иванов иван иванович' s2='иван' print(s1.count(s2)) **Получим**: 2 Обратите внимание на регистр символов

s.lstrip([chars])	Удаление символов **chars** в начале строки. По умолчанию удаляются пробелы.	s=' Python' print(s.lstrip()) **Получим**: 'Python'
s.rstrip([chars])	Удаление символов **chars** в конце строки. По умолчанию удаляются пробелы.	s='Python ' print(s.rstrip()) **Получим**: 'Python'
s.strip([chars])	Удаление символов **chars** в начале и конце строки. По умолчанию удаляются пробелы.	s=' Python ' print(s.strip()) **Получим**: 'Python'
s.swapcase()	Перевод символов из нижнего регистра в верхний, а из верхнего в нижний.	s='PyThOn' print(s1.swapcase()) **Получим**: pYtHoN
s.title()	Перевод первой буквы каждого слова в верхний регистр, а все остальные буквы в нижний.	s='иванов иван иванович' print(s1.capitalize()) **Получим**: Иванов Иван Иванович

5.3. Циклическая обработка строк

Так как строка представляет собой последовательность символов, мы можем перебрать все символы, используя циклы.

Рассмотрим **2 способа** перебора символов строки:

1. Использование цикла **for** и функции нахождения длины строки.

```
s = 'Python'
for i in range(len(s)):
  print(s[i], end=' ')
```

```
P y t h o n
```

2. Использование цикла **for** и его возможности работать с любыми последовательностями (строка – это последовательность).

```
s = 'Python'
for i in s:
    print(i, end=' ')
```

```
P y t h o n
```