Урок 7.  
Стандартные функции и методы.

короткая линия

# План урока

1. Методы строк
2. Методы списков
3. Изменяют объект vs Возвращают значение

Данный урок содержит немного авторского материала, так как таблицы с описанием методов взяты с замечательного сайта [pythonworld.ru](https://pythonworld.ru/) (там вы можете найти для себя много полезного).

# Методы строк

Вы уже умеете работать со строками на базовом уровне — складывать, умножать на число, брать срезы, а также пользоваться методами upper() и lower() (если вы, конечно, делали задачи с урока 6), — однако Python предоставляет множество дополнительных инструментов по работе над строками. Основные методы и функции, которые стоит изучить, представлены ниже:

|  |  |
| --- | --- |
| **S1 + S2** | Конкатенация (сложение строк) |
| **S1 \* 3** | Повторение строки |
| **S[i]** | Обращение по индексу |
| **S[i:j:step]** | Извлечение среза |
| **len**(S) | Длина строки |
| **S.find**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или -1 |
| **S.rfind**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или -1 |
| **S.replace**(шаблон, замена) | Замена шаблона |
| **S.split**(символы) | Разбиение строки по разделителю |
| **S.isdigit**() | Состоит ли строка из цифр |
| **S.isalpha**() | Состоит ли строка из букв |
| **S.isalnum**() | Состоит ли строка из цифр или букв |
| **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 |
| **ord(символ)** | Символ в его код ASCII |
| **chr(число)** | Код ASCII в символ |
| **S.capitalize()** | Переводит первый символ строки в верхний регистр, а все остальные в нижний |
| **S.swapcase()** | Переводит символы нижнего регистра в верхний, а верхнего – в нижний |
| **S.lstrip()** | Удаление пробельных символов в начале строки |
| **S.rstrip()** | Удаление пробельных символов в конце строки |
| **S.strip()** | Удаление пробельных символов в начале и в конце строки |

Не стоит пугаться такого обширного количества функций и методов: вы можете обращаться к таблице в любое время, главное запомнить суть самых часто используемых инструментов и хорошо понимать, что именно вы можете делать со строками. В любом случае, самые популярные методы вы хорошо запомните, если решите несколько задач и попрактикуетесь. Если суть какого-либо из методов вам непонятна, вам стоит поискать объяснение в интернете или спросить у учителя.

Разумеется, данная таблица является лишь выборкой из всех методов. Ознакомиться с первоисточников таблицы можно [здесь](https://pythonworld.ru/tipy-dannyx-v-python/stroki-funkcii-i-metody-strok.html). Также вы можете попрактиковаться [чтению документации](https://docs.python.org/2/library/stdtypes.html#string-methods).

**Задача “Инверсия регистра”**

**Задача “Сохранить уникальность”**

**Задача “Редактирование пробелов”**

**Задача “Заменить цифры”**

# Методы списков

Мы уже говорили, что списки сами по себе являются наиболее удобным способом хранения произвольных данных из-за того, что они изменяемые. Как вы можете догадаться, возможности списков не ограничиваются добавлением элемента в конец и применением срезов. Ниже представлены основные методы (полная версия статьи [тут](https://pythonworld.ru/tipy-dannyx-v-python/spiski-list-funkcii-i-metody-spiskov.html)):

|  |  |
| --- | --- |
| **list.append**(x) | Добавляет элемент в конец списка |
| **list.extend**(L) | Расширяет список list, добавляя в конец все элементы списка L |
| **list.insert**(i, x) | Вставляет на i-ую позицию элемент x |
| **list.remove**(x) | Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует |
| **list.pop**([i]) | Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент |
| **list.index**(x, [start [, end]]) | Возвращает индекс первого элемента со значением x (при этом поиск ведется от start до end) |
| **list.count**(x) | Возвращает количество элементов со значением x |
| **list.sort**() | Сортирует список |
| **list.reverse**() | Разворачивает список |
| **list.copy**() | Поверхностная копия списка |
| **list.clear**() | Очищает список |

Кстати, о списках в целом очень хорошо [написано](https://docs.python.org/2/tutorial/datastructures.html) на официальном сайте (если, конечно, вам подвластно чтение на английском языке).

**Задача “Определить количество”**

**Задача “Два максимума”**

**Задача “Сортировка половинок”**

Также стоит понимать, когда метод возвращает значение, а когда изменяет объект, к которому применяется. Здесь пригодится понимание того, является ли объект изменяемым или неизменяемым. В случае списков, если методы призваны изменить исходный список, вам не нужно перезаписывать его значение в переменную (вы убедились в этом на примере использования .append(); так же работает, например, метод .sort()). В обратном случае метод возвращает значение, которое может быть сохранено в переменной (например, индекс элемента или количество). В методах строк есть некоторые различия. Как вы помните, строки — неизменяемые объекты, поэтому все методы, модернизирующие исходную строку (upper(), replace() и т. д.) возвращают новую строку, которую нужно сохранить.

**Домашнее задание:**

**Задача “Палиндром”**

**Задача “Индексы минимального”**

**Задача “Санта-Барбара”**