

# Разработка программного обеспечения на языке Python

[Обзорная панель](#)[Мои курсы](#)[Разработка ПО на языке Python](#)[Программирование на языке Python](#)[Лекция 3. Основы структур данных](#)

## Лекция 3. Основы структур данных

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

### Списки



## МЕТОД APPEND

добавляет элемент item в конец списка

```
words = ["first", "second", "third"]  
words.append("fourth")
```

00:00 / 05:19



Здравствуйте! В этой лекции мы подробно поговорим о работе с наборами данных в Python. Для работы с наборами данных Python предоставляет такие встроенные типы как списки, кортежи и словари. В этом видеоуроке мы подробнее остановимся на списках.

Список (list) представляет тип данных, который хранит набор или последовательность элементов. Во многих языках программирования есть аналогичная структура данных, которая называется массив.

Для создания списка применяются квадратные скобки [], внутри которых через запятую перечисляются элементы списка. На экране вы можете увидеть определение списка чисел и списка строк.

```
numbers = [1, 2, 3, 4, 5]  
words = ['first', 'second', 'third']
```

Для создания пустого списка можно использовать функцию-конструктор list() или просто указать квадратные скобки. Оба этих определения списка аналогичны - они создают пустой список.

```
numbers1 = []  
numbers2 = list()
```

Список необязательно должен содержать только однотипные объекты. Мы можем поместить в один и тот же список одновременно строки, числа, объекты других типов данных, например целочисленные значения, вещественные числа, строки, логического типа.

```
objects = [1, 2.6, "Hello", True]
```

Для обращения к элементам списка надо использовать индексы, которые представляют номер элемента в списка. Индексы начинаются с нуля. То есть первый элемент будет иметь индекс 0, второй элемент - индекс 1 и так далее. Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего - -2 и так далее.

```
words = ['first', 'second', 'third']
print(words[0]) #first
print(words[1]) #second
print(words[2]) #third
```

Для перебора элементов можно использовать как цикл for, так и цикл while.

На экране показан перебор элементов списка words с помощью цикла for.

```
words = ["first", "second", "third"]

for word in words:
    print(word)
```

Перебор также можно сделать с помощью цикла while. Здесь с помощью функции len() мы получили длину списка. Обращаясь к каждому i-му элементу списка мы выводим его на экран, пока значение счетчика не станет равно длине списка.

```
words = ["first", "second", "third"]

i = 0
while i < len(words):
    print(words[i])
    i+=1
```

Два списка считаются равными, если они содержат один и тот же набор элементов. Для сравнения двух списком мы можем воспользоваться операциями сравнения, с которыми познакомились в предыдущих видео.

```
numbers1 = [1, 2, 3, 4, 5]
numbers2 = [6, 7, 8]
if numbers1 == numbers2:
    print("numbers1 равен numbers2")
else:
    print("numbers1 НЕ равен numbers2")
```

Для управления элементами списки имеют целый ряд методов. Для добавления элемента применяются методы append(), extend и insert,

Метод append добавляет элемент item в конец списка. Использование метода append приведено на экране. Здесь мы вставляем в конец списка слово «fourth», а также выводим на консоль итоговый список.

```
words = ["first", "second", "third"]
words.append("fourth")

i = 0
while i < len(words):
    print(words[i])
    i+=1
```

Метод insert добавляет элемент item в список по индексу. Так мы можем вставить в наш список слово «name» под индексом 2.

```
words = ['first', 'second', 'third', 'fourch']
words.insert(2, 'name')

print(words)
```

Метод `extend` добавляет набор элементов `items` в конец списка. Здесь мы вставляем в конец наш списка три слова.

```
words = ['first', 'second', 'third', 'fourch']
words.extend(['fifth', 'sixth', 'seventh'])

print(words)
```

Для удаления элементов из списка применяют методы `remove()`, `pop()` и `clear()`.

Метод `remove` удаляет элемент `item`. Причем удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение `ValueError`. На экране вы можете увидеть, что мы удалили из списка элемент `second`.

```
words = ['first', 'second', 'third', 'fourch']
words.remove('second')
```

Метод `clear` позволяет удалить все элементы.

```
words = ['first', 'second', 'third', 'fourch']
words.clear()
```

Метод `pop` удаляет и возвращает элемент по индексу `index`. Если индекс не передан, то просто удаляет последний элемент.

```
words = ['first', 'second', 'third', 'fourch']
removed_item = words.pop(2)
```

Разберем еще несколько полезных методов, которые можно применять в работе со списками.

Метод `index(item)` возвращает индекс элемента `item`. Если элемент не найден, генерирует исключение `ValueError`. Так мы нашли элемент под индексом 2, записали его в переменную `find_item` и вывели его на экран.

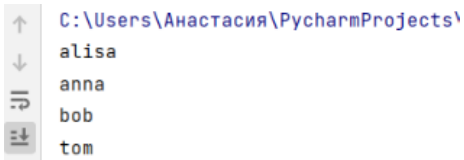
```
words = ['first', 'second', 'third', 'fourch']
find_item = words.index('second')
print(find_item)
```

Метод `Count(item)`: возвращает количество вхождений элемента `item` в список. Так слово «second» здесь встречается 2 раза.

```
words = ['first', 'second', 'third', 'fourch']
count_second = words.count('second')
print(count_second)
```

Метод `sort([key])`: сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра `key` мы можем передать функцию сортировки. В случае, который вы видите на экране в метод `sort` не передается параметров, поэтому список будет отсортирован по алфавиту.

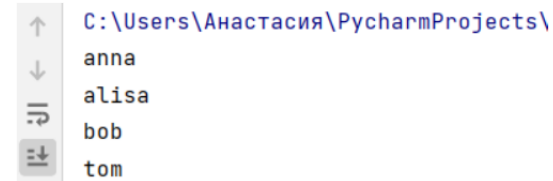
```
words = ["tom", "bob", "alisa", "anna"]
words.sort()
i = 0
while i < len(words):
    print(words[i])
    i+=1
```



```
C:\Users\Анастасия\PycharmProjects\  
alisa  
anna  
bob  
tom
```

Метод `reverse()`: расставляет все элементы в списке в обратном порядке

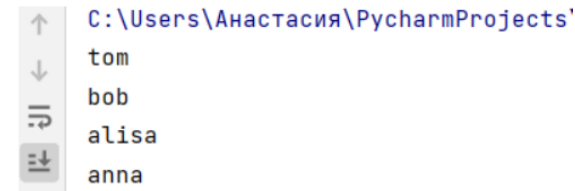
```
words = ["tom", "bob", "alisa", "anna"]  
words.reverse()  
i = 0  
while i < len(words):  
    print(words[i])  
    i+=1
```



```
C:\Users\Анастасия\PycharmProjects\  
anna  
alisa  
bob  
tom
```

Метод `copy()`: копирует список. Здесь мы скопировали список `words` в список `words1` и вывели последний на консоль.

```
words = ["tom", "bob", "alisa", "anna"]  
words1 = words.copy()  
i = 0  
while i < len(words1):  
    print(words1[i])  
    i+=1
```



```
C:\Users\Анастасия\PycharmProjects\  
tom  
bob  
alisa  
anna
```

Если определенный элемент не найден, то методы `remove` и `index` генерируют исключение. Чтобы избежать подобной ситуации, перед операцией с элементом можно проверять его наличие с помощью ключевого слова `in`. Так, здесь, перед удалением элемента «bob» мы проверили его наличие в списке `words`.

```
words = ['tom', 'bob', 'alisa', 'anna']  
if 'bob' in words:  
    words.remove('bob')
```

Встроенный функции Python `min()` и `max()` позволяют найти минимальное и максимальное значения соответственно.

В этом довольно большом видеоуроке мы рассмотрели понятие списка в Python, процесс создания нового списка, а также подробно разобрали методы работы со списками.

Кортежи и словари

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◀ [Задание 4. Числа Фибоначчи](#)

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

[Задание 5. Строки в Python ►](#)

© 2010-2023 Центр обучающих систем  
Сибирского федерального университета, sfu-kras.ru

Разработано на платформе moodle  
Beta-version (3.9.1.5.w3)

[Политика конфиденциальности](#)

[Соглашение о Персональных данных](#)

[Политика допустимого использования](#)

**Контакты** +7(391) 206-27-05  
[info-ms@sfu-kras.ru](mailto:info-ms@sfu-kras.ru)

[Скачать мобильное приложение](#)

[Инструкции по работе в системе](#)