



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных  
технологий

«Списки в Python: Подробное руководство»

Студент группы ИКБО-68-23

Никонов Никита Григорьевич

Москва, 2024

# Оглавление

<b>«Списки в Python: Подробное руководство» .....</b>	<b>1</b>
<b>Списки в Python: Подробное руководство .....</b>	<b>3</b>
<b>История создания списков в Python.....</b>	<b>3</b>
<b>Почему списки, а не массивы? .....</b>	<b>3</b>
1. Динамическая природа списка: .....	3
2. Преимущества списков перед массивами: .....	3
3. Массивы в других языках: .....	4
<b>Зачем выбрать список в Python? .....</b>	<b>4</b>
<b>1. Основы работы со списками .....</b>	<b>5</b>
1.1. Создание списка .....	5
1.2. Индексация и срезы.....	5
<b>2. Операции с элементами списка .....</b>	<b>5</b>
2.1. Добавление элементов .....	5
2.2. Удаление элементов.....	6
2.3. Поиск элементов.....	6
2.4. Сортировка и изменение порядка .....	6
<b>3. Копирование списка .....</b>	<b>6</b>
3.1. Поверхностная копия .....	6
3.2. Копирование с помощью среза .....	7
<b>4. Другие полезные операции .....</b>	<b>7</b>
<b>5. Вложенные списки.....</b>	<b>7</b>
<b>6. Итерации и сложные операции .....</b>	<b>7</b>
<b>Вывод .....</b>	<b>8</b>

# Списки в Python: Подробное руководство

Список в Python — это одна из самых мощных и универсальных структур данных. Он представляет собой изменяемую последовательность элементов, которая может содержать объекты любого типа, включая другие списки. Списки позволяют хранить элементы в упорядоченном виде, и их размер может изменяться в процессе работы программы.

## История создания списков в Python

Python был создан **Гвидо ван Россумом** в конце 1980-х годов в Нидерландах, а первая версия Python была выпущена в 1991 году. Вдохновением для создания языка послужили такие языки, как **ABC**, **C** и **Modula-3**, но также и широкий спектр других языков программирования.

Когда ван Россум разрабатывал Python, он хотел, чтобы этот язык был простым в использовании, но в то же время мощным и гибким. Это желание привело его к решению предоставить разработчикам инструменты для удобной работы с коллекциями данных.

## Почему списки, а не массивы?

Основной особенностью списков в Python является то, что они — **динамические массивы** (или динамические списки), в отличие от статических массивов, которые есть в других языках программирования, таких как C, C++, Java и других.

### 1. Динамическая природа списка:

- В отличие от массивов в языках, таких как C, которые имеют фиксированный размер, списки в Python могут изменять свой размер в процессе работы программы.
- Когда в список добавляется элемент, Python автоматически перераспределяет память и при необходимости увеличивает размер массива, не требуя от программиста явного управления этим процессом.
- Это делает списки гораздо более удобными и гибкими, особенно при работе с изменяющимся набором данных.

### 2. Преимущества списков перед массивами:

- **Проще в использовании:** в Python список можно создать, просто перечислив элементы в квадратных скобках, без необходимости указывать его размер. Например:

```
Копировать код
fruits = ["apple", "banana", "cherry"]
```

- **Гибкость типов данных:** элементы списка могут быть разных типов, включая другие списки, что делает их идеальными для хранения сложных и гибких структур данных.

Копировать код  
`mixed = [1, "apple", True, [1, 2, 3]]`

- **Методы для работы с данными:** списки в Python поддерживают широкий набор методов для добавления, удаления и модификации элементов, что делает их удобными для работы в повседневной разработке.

### 3. Массивы в других языках:

- В языках, таких как C или Java, массивы имеют фиксированный размер, что требует явного управления памятью и размера массива. Это может быть удобным в ситуациях, где размер данных заранее известен, но может быть неудобным при работе с переменными размерами данных.
- В Python массивы существуют (например, с помощью библиотеки `array`), но они не являются основным инструментом для работы с коллекциями данных, поскольку списки уже выполняют эту роль, предлагая более широкую функциональность.

## Зачем выбрать список в Python?

Списки были выбраны в Python как основной способ хранения коллекций данных по нескольким ключевым причинам:

1. **Простота и удобство использования:** Списки предоставляют пользователям удобный и интуитивно понятный интерфейс для работы с данными.
2. **Гибкость:** Списки могут содержать элементы любых типов, что делает их универсальными.
3. **Динамичность:** Python сам управляет памятью, что освобождает программиста от необходимости вручную управлять размерами коллекций.

# 1. Основы работы со списками

## 1.1. Создание списка

Списки могут содержать элементы различных типов данных. Вот несколько примеров:

```
# Список чисел
numbers = [1, 2, 3, 4, 5]

# Список строк
fruits = ["apple", "banana", "cherry"]

# Список с разными типами данных
mixed = [1, "apple", True, 3.14]

# Пустой список
empty_list = []
```

## 1.2. Индексация и срезы

Элементы списка могут быть доступны через индекс. Индексация начинается с 0. Списки также поддерживают срезы:

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0]) # 'apple'
print(fruits[1]) # 'banana'
print(fruits[-1]) # 'cherry' (отрицательные индексы начинаются с конца)

numbers = [1, 2, 3, 4, 5]
print(numbers[1:4]) # [2, 3, 4]
```

# 2. Операции с элементами списка

## 2.1. Добавление элементов

Для добавления элементов в список можно использовать методы:

- `append(x)` — добавляет элемент в конец списка:

```
fruits.append("cherry")
print(fruits) # ['apple', 'banana', 'cherry']
```

- `insert(i, x)` — вставляет элемент на позицию `i`:

```
fruits.insert(1, "banana")
print(fruits) # ['apple', 'banana', 'cherry']
```

- `extend(iterable)` — добавляет элементы из другого списка:

```
fruits.extend(["orange", "kiwi"])
print(fruits) # ['apple', 'banana', 'cherry', 'orange', 'kiwi']
```

## 2.2. Удаление элементов

Для удаления элементов из списка есть несколько методов:

- `remove(x)` — удаляет первый элемент, равный `x`:

```
fruits.remove("banana")
print(fruits)  # ['apple', 'cherry']
```

- `pop(i)` — удаляет элемент на позиции `i` и возвращает его:

```
last = fruits.pop()
print(last)  # 'cherry'
```

- `clear()` — удаляет все элементы:

```
fruits.clear()
print(fruits)  # []
```

## 2.3. Поиск элементов

- `index(x)` — возвращает индекс первого элемента, равного `x`:

```
print(fruits.index("banana"))  # 1
```

- `count(x)` — возвращает количество элементов, равных `x`:

```
print(numbers.count(2))  # 3
```

## 2.4. Сортировка и изменение порядка

Списки поддерживают методы сортировки и изменения порядка элементов:

- `sort()` — сортирует список:

```
numbers.sort()
print(numbers)  # [1, 2, 5, 9]
```

- `reverse()` — переворачивает список:

```
numbers.reverse()
print(numbers)  # [9, 5, 2, 1]
```

## 3. Копирование списка

### 3.1. Поверхностная копия

Метод `copy()` создает поверхностную копию списка:

```
copy_list = original.copy()
```

### 3.2. Копирование с помощью среза

```
copy_list = original[:]
```

## 4. Другие полезные операции

- Проверка наличия элемента с помощью оператора `in`:

```
print("banana" in fruits) # True
```

- Перебор элементов с помощью цикла `for`:

```
for fruit in fruits:  
    print(fruit)
```

- Генераторы списков для создания новых списков:

```
squares = [x**2 for x in range(5)]  
print(squares) # [0, 1, 4, 9, 16]
```

## 5. Вложенные списки

Списки могут содержать другие списки, что делает их отличным выбором для представления многомерных данных (например, матриц):

```
matrix = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]  
print(matrix[0][1]) # 2
```

## 6. Итерации и сложные операции

Списки поддерживают различные способы итерации. Например, с помощью функций `map()` и `filter()` можно легко применить операции ко всем элементам:

- `map()` применяет функцию ко всем элементам:

```
squared = list(map(lambda x: x**2, numbers))  
print(squared) # [1, 4, 9, 16]
```

- `filter()` фильтрует элементы по условию:

```
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))  
print(even_numbers) # [2, 4]
```

## Вывод

Списки в Python — это гибкий и мощный инструмент для работы с данными. Они позволяют эффективно организовывать, изменять и обрабатывать данные с помощью множества встроенных методов. Списки являются основным типом данных для работы с коллекциями элементов в Python благодаря своей универсальности и динамичности, что делает их идеальными для большинства задач программирования.