

Python

СПИСКИ

list()

Пример создания списка

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

list()

Конструктор может принимать другой список

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2 numbers2 = list(numbers)
```

Обращение через индексы

```
1 numbers = [1, 2, 3, 4, 5]
2 print(numbers[0])      # 1
3 print(numbers[2])     # 3
4 print(numbers[-3])    # 3
5
6 numbers[0] = 125      # изменяем первый элемент списка
7 print(numbers[0])     # 125
```

Последовательный список чисел

`range(end)`: создается набор чисел от 0 до числа `end`

`range(start, end)`: создается набор чисел от числа `start` до числа `end`

`range(start, end, step)`: создается набор чисел от числа `start` до числа `end` с шагом `step`

Последовательный список чисел

```
1 numbers = list(range(10))
2 print(numbers)          # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3 numbers = list(range(2, 10))
4 print(numbers)          # [2, 3, 4, 5, 6, 7, 8, 9]
5 numbers = list(range(10, 2, -2))
6 print(numbers)          # [10, 8, 6, 4]
```

Одинаковые ли это команды?

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2 numbers2 = list(range(1, 10))
```

Объекты списка

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


Перебор элементов

```
1 companies = ["Microsoft", "Google", "Oracle", "Apple"]
2 for item in companies:
3     print(item)
```

```
1 companies = ["Microsoft", "Google", "Oracle", "Apple"]
2 i = 0
3 while i < len(companies):
4     print(companies[i])
5     i += 1
```

Сравнение списков

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2 numbers2 = list(range(1,10))
3 if numbers == numbers2:
4     print("numbers equal to numbers2")
5 else:
6     print("numbers is not equal to numbers2")
```

Методы и функции по работе со списками

Для управления элементами списки имеют целый ряд методов. Некоторые из них:

- `append(item)`: добавляет элемент `item` в конец списка
- `insert(index, item)`: добавляет элемент `item` в список по индексу `index`
- `remove(item)`: удаляет элемент `item`. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение `ValueError`
- `clear()`: удаление всех элементов из списка
- `index(item)`: возвращает индекс элемента `item`. Если элемент не найден, генерирует исключение `ValueError`

Методы и функции по работе со списками

Для управления элементами списки имеют целый ряд методов. Некоторые из них:

- `pop([index])`: удаляет и возвращает элемент по индексу `index`. Если индекс не передан, то просто удаляет последний элемент.
- `count(item)`: возвращает количество вхождений элемента `item` в список
- `sort([key])`: сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра `key` мы можем передать функцию сортировки.
- `reverse()`: расставляет все элементы в списке в обратном порядке

Методы и функции по работе со списками

Кроме того, Python предоставляет ряд встроенных функций для работы со списками:

- `len(list)`: возвращает длину списка
- `sorted(list, [key])`: возвращает отсортированный список
- `min(list)`: возвращает наименьший элемент списка
- `max(list)`: возвращает наибольший элемент списка

Добавление и удаление элементов

```
1 users = ["Tom", "Bob"]
2
3 # добавляем в конец списка
4 users.append("Alice") # ["Tom", "Bob", "Alice"]
5 # добавляем на вторую позицию
6 users.insert(1, "Bill") # ["Tom", "Bill", "Bob", "Alice"]
7
8 # получаем индекс элемента
9 i = users.index("Tom")
10 # удаляем по этому индексу
11 removed_item = users.pop(i) # ["Bill", "Bob", "Alice"]
12
13 last_user = users[-1]
14 # удаляем последний элемент
15 users.remove(last_user) # ["Bill", "Bob"]
16
17 print(users)
18
19 # удаляем все элементы
20 users.clear()
```

Проверка наличия элемента

```
1 companies = ["Microsoft", "Google", "Oracle", "Apple"]
2 item = "Oracle" # элемент для удаления
3 if item in companies:
4     companies.remove(item)
5
6 print(companies)
```

Подсчет вхождений

```
1 users = ["Tom", "Bob", "Alice", "Tom", "Bill", "Tom"]
2
3 users_count = users.count("Tom")
4 print(users_count)      # 3
```


Сортировка

```
1 users = ["Tom", "Bob", "Alice", "Sam", "Bill"]
2
3 users.sort()
4 print(users)          # ["Alice", "Bill", "Bob", "Sam", "Tom"]
```

```
1 users = ["Tom", "Bob", "Alice", "Sam", "Bill"]
2
3 users.sort()
4 users.reverse()
5 print(users)          # ["Tom", "Sam", "Bob", "Bill", "Alice"]
```

Сортировка

```
1 users = ["Tom", "bob", "alice", "Sam", "Bill"]
2
3 users.sort(key=str.lower)
4 print(users)      # ["alice", "Bill", "bob", "Sam", "Tom"]
```

Сортировка

Дополнительные методы сортировки

`sorted(list)`: сортирует список `list`

`sorted(list, key)`: сортирует список `list`, применяя к элементам функцию `key`

```
1 users = ["Tom", "bob", "alice", "Sam", "Bill"]
2
3 sorted_users = sorted(users, key=str.lower)
4 print(sorted_users)           # ["alice", "Bill", "bob", "Sam", "Tom"]
```

Минимальное и максимальное значения

```
1 numbers = [9, 21, 12, 1, 3, 15, 18]
2 print(min(numbers))      # 1
3 print(max(numbers))     # 21
```

Копирование списков

При копировании списков следует учитывать, что списки представляют изменяемый (mutable) тип, поэтому если обе переменных будут указывать на один и тот же список, то изменение одной переменной, затронет и другую переменную

```
1 users1 = ["Tom", "Bob", "Alice"]
2 users2 = users1
3 users2.append("Sam")
4 # users1 и users2 указывают на один и тот же список
5 print(users1)    # ["Tom", "Bob", "Alice", "Sam"]
6 print(users2)    # ["Tom", "Bob", "Alice", "Sam"]
```

Копирование списков

И чтобы происходило копирование элементов, но при этом переменные указывали на разные списки, необходимо выполнить глубокое копирование (deep copy). Для этого можно использовать метод `deepcopy()`, который определен во встроенном модуле `copy`

```
1 import copy
2
3 users1 = ["Tom", "Bob", "Alice"]
4 users2 = copy.deepcopy(users1)
5 users2.append("Sam")
6 # переменные users1 и users2 указывают на разные списки
7 print(users1)    # ["Tom", "Bob", "Alice"]
8 print(users2)    # ["Tom", "Bob", "Alice", "Sam"]
```

Копирование части списка

`list[:end]`: через параметр `end` передается индекс элемента, до которого нужно копировать список

`list[start:end]`: параметр `start` указывает на индекс элемента, начиная с которого надо скопировать элементы

`list[start:end:step]`: параметр `step` указывает на шаг, через который будут копироваться элементы из списка. По умолчанию этот параметр равен 1.

Копирование части списка

```
1 users = ["Tom", "Bob", "Alice", "Sam", "Tim", "Bill"]
2
3 slice_users1 = users[:3] # с 0 по 3
4 print(slice_users1) # ["Tom", "Bob", "Alice"]
5
6 slice_users2 = users[1:3] # с 1 по 3
7 print(slice_users2) # ["Bob", "Alice"]
8
9 slice_users3 = users[1:6:2] # с 1 по 6 с шагом 2
10 print(slice_users3) # ["Bob", "Sam", "Bill"]
```


Соединение списков

```
1 users1 = ["Tom", "Bob", "Alice"]
2 users2 = ["Tom", "Sam", "Tim", "Bill"]
3 users3 = users1 + users2
4 print(users3) # ["Tom", "Bob", "Alice", "Tom", "Sam", "Tim", "Bill"]
```

Списки списков

```
1 users = [  
2     ["Tom", 29],  
3     ["Alice", 33],  
4     ["Bob", 27]  
5 ]  
6  
7 print(users[0])           # ["Tom", 29]  
8 print(users[0][0])       # Tom  
9 print(users[0][1])       # 29
```

Списки списков

Добавление, удаление и изменение общего списка, а также вложенных списков аналогично тому, как это делается с обычными (одномерными) списками:

```
1 users = [  
2     ["Tom", 29],  
3     ["Alice", 33],  
4     ["Bob", 27]  
5 ]  
6  
7 # создание вложенного списка  
8 user = list()  
9 user.append("Bill")  
10 user.append(41)  
11 # добавление вложенного списка  
12 users.append(user)  
13  
14 print(users[-1])           # ["Bill", 41]  
15  
16 # добавление во вложенный список  
17 users[-1].append("+79876543210")  
18  
19 print(users[-1])           # ["Bill", 41, "+79876543210"]  
20  
21 # удаление последнего элемента из вложенного списка
```

```
21 # удаление последнего элемента из вложенного списка
22 users[-1].pop()
23 print(users[-1])          # ["Bill", 41]
24
25 # удаление всего последнего вложенного списка
26 users.pop(-1)
27
28 # изменение первого элемента
29 users[0] = ["Sam", 18]
30 print(users)              # [ ["Sam", 18], ["Alice", 33], ["Bob", 27]]
```

Перебор вложенных списков

```
1 users = [  
2     ["Tom", 29],  
3     ["Alice", 33],  
4     ["Bob", 27]  
5 ]  
6  
7 for user in users:  
8     for item in user:  
9         print(item, end=" | ")
```

Tom | 29 | Alice | 33 | Bob | 27 |

