Список

- Список (list) – это упорядоченная последовательность из нуля или более ссылок на объекты. Списки поддерживают тот же синтаксис получения срезов, что и строки с кортежами.

Методы списков

Синтаксис	Описание
L.append (x)	Добавляет элемент x в конец списка L
L.count(x)	Возвращает число вхождений элемента x в список L
L.extend(m) L += m	Добавляет в конец списка L все элементы итерируемого объекта m; оператор += делает то же самое
L.index(x, start, end)	Возвращает индекс самого первого (слева) вхождения элемента x в список L (или в срез start:end списка L), в противном случае возбуждает исключение ValueError
L.insert(i, x)	Вставляет элемент x в список L в позицию int i

Методы списков

Синтаксис	Описание
L.pop()	Удаляет самый последний элемент из списка L и возвращает его в качестве результата
L.pop(i)	Удаляет из списка L элемент с индексом int i и возвращает его в качестве результата
L.remove(x)	Удаляет самый первый (слева) найденный элемент x из списка L или возбуждает исключение ValueError, если элемент x не будет найден
L.reverse()	Переставляет в памяти элементы списка в обратном порядке
L.sort(...)	Сортирует список в памяти. Этот метод принимает те же необязательные аргументы key и reverse, что и встроенная функция sorted()

Списки

- Существует четыре способа добавить элементы в список.
- >>> a_list = ['a']
  >>> a_list = a_list + [2.0, 3]
  >>> a_list
  ['a', 2.0, 3]
  >>> a_list.append(True)
  >>> a_list
  ['a', 2.0, 3, True]
  >>> a_list.extend(['four', 'Ω'])
  >>> a_list
  ['a', 2.0, 3, True, 'four', 'Ω']
  >>> a_list.insert(0, 'Ω')
  >>> a_list
  ['Ω', 'a', 2.0, 3, True, 'four', 'Ω']

Разрезание списков

- >>> a_list
  ['a', 'b', 'mpilgrim', 'z', 'example']
  >>> a_list[1:3]
  ['b', 'mpilgrim']
  >>> a_list[1:-1]
  ['b', 'mpilgrim', 'z']
  >>> a_list[0:3]
  ['a', 'b', 'mpilgrim']
  >>> a_list[:3]
  ['a', 'b', 'mpilgrim']
  >>> a_list[3:]
  ['z', 'example']
  >>> a_list[:]
  ['a', 'b', 'mpilgrim', 'z', 'example']

Поиск в списке

- >>> a_list = ['a', 'b', 'new', 'mpilgrim', 'new']
  >>> a_list.count('new')
  2
  >>> 'new' in a_list
  True
  >>> 'c' in a_list
  False
  >>> a_list.index('mpilgrim')
  3
  >>> a_list.index('new')
  2
  >>> a_list.index('c')
  Traceback (innermost last):
  File "<interactive input>", line 1, in ?
  ValueError: list.index(x): x not in list

Удаление элементов из списка

- >>> a_list = ['a', 'b', 'new', 'mpilgrim', 'new']
  >>> a_list[1]
  'b'
  >>> del a_list[1]
  >>> a_list
  ['a', 'new', 'mpilgrim', 'new']
  >>> a_list[1]
  'new'

Удаление элементов по значению

- >>> a_list.remove('new')
  >>> a_list
  ['a', 'mpilgrim', 'new']
  >>> a_list.remove('new')
  >>> a_list
  ['a', 'mpilgrim']
  >>> a_list.remove('new')
  Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
  ValueError: list.remove(x): x not in list

Генераторы списков

- Генератор списков – это выражение и цикл с дополнительным условием, заключенное в квадратные скобки, в котором цикл используется для создания элементов списка, а условие используется для исключения нежелательных элементов.
- В простейшем виде генератор списков записывается, как показано ниже:

```
[item for item in iterable]
```

Генераторы списков

- Генераторы могут использоваться как выражения и они допускают включение условной инструкции, вследствие чего мы получаем две типичные синтаксические конструкции использования генераторов списков:

```
[expression for item in iterable]

[expression for item in iterable if condition]
```

Примеры

```
>>> a = [i*10 for i in range(5)]
>>> a
[0, 10, 20, 30, 40]

>>> a = [x**2 for x in range(50) if x % 10 == 5]
>>> a
[25, 225, 625, 1225, 2025]
```