Списки

Списки - это тип данных, хранящий любое количество разных элементов одного типа. Списки можно изменять. Если элементы, лежащие в списке, допускают изменения себя, то их можно также изменять по отдельности.

Пример работы со списком:

```
my_list = []                # объявляет пустой список, названный `my_list`
my_list.append(1)           # добавляет в конец списка элемент `1`
my_list.append(2)           # добавляет в конец списка элемент `2`
my_list.append(3)           # добавляет в конец списка элемент `3`

print(my_list[0])           # выводит элемент под индексом 0
print(my_list[1])           # выводит элемент под индексом 1
print(my_list[2])           # выводит элемент под индексом 2

for number in my_list:      # для каждого элемента `number` в списке `my_list`:
    print(number)           # выводит элемент `number`
```

Вывод программы:

```
1
2
3
1
2
3
```

Для получения доступа к конкретному элементу в списке используются квадратные скобки, в которые передается *индекс* элемента.
Индекс элемента - это его порядковый номер **от нуля**.
Если попытаться получить доступ к индексу несуществующего элемента, то произойдет ошибка (исключение):

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])              # выводит первый элемент списка `numbers` - 1
print(numbers[-1])             # выводит последний элемент списка - 5
print(numbers[100])            # ошибка: нет элемента под индексом 100
```

Список можно объявить так:

```
a = []           # объявляет пустой список с именем `a`
b = [1, 2]       # объявляет список с именем `b` и элементами 1 и 2
c = list()       # объявляет пустой список с именем `c`. Аналогично записи c = [].
```

Операции над списками

Получить длину списка

Для получения количества элементов в списке используется функция `len()`:

```
a = []
b = ["cat", "dog", "fox", "duck"]

print(len(a))   # выведет 0
print(len(b))   # выведет 4
```

Сложение и умножение списков

По аналогии со строками, списки можно складывать и умножать на целые числа. Сложение двух списков с одинаковыми типами данных "склеит" их между собой в новый список:

```
nums_1 = [1, 2, 3, 4, 5]
nums_2 = [5, 6, 7, 8, 9, 10]
numbers = nums_1 + nums_2

print(numbers)           # выведет [1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10]
```

Умножение списка на число N приведет к повторению списка N раз:

```
numbers = [1, 2, 3]
n = 3

print(numbers * n)       # выведет [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Подсчитать количество определенных элементов в списке:

Если у нас есть список значений, и мы хотим узнать, сколько раз повторяется определенное значение, то можно использовать метод `count()`:

```
sounds = ["meow", "bark", "meow"]

print(sounds.count("meow"))     # выведет 2, так как "meow" встречается в списке 2 р
print(sounds.count("bark"))     # выведет 1, так как "bark" встречается в списке 1 р
print(sounds.count("quack"))    # выведет 0, так как "quack" не встречается в списке
```

Получить диапазон (подсписок), или `slice` из списка:

По аналогии с получением одного элемента из списка мы можем получить диапазон значений *от* и *до* определенных индексов с некоторым *шагом*.

Синтаксис диапазона (или слайса - от англ. `slice`) такой:

$$[A : B : N]$$

где A - индекс начала диапазона, B - индекс конца диапазона (невключительно), N - шаг диапазона.
Если A не указано, то предполагается, что оно равно **нулю**.
Если B не указано, то предполагается, что оно равно **индексу последнего элемента в списке**.
Если N не указано, то предполагается, что оно равно **единице**.

Пример:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(numbers[0:5:1])   # выведет элементы с 0-го по 5-ый: [1, 2, 3, 4, 5]
print(numbers[5::])     # выведет элементы с 5-го по последний: [6, 7, 8, 9, 10]
print(numbers[::2])     # выведет элементы с первого по последний с шагом 2:
                        # [1, 3, 5, 7, 9]
print(numbers[::-1])    # выведет элементы с первого по последний в обратном порядке
                        # [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Развернуть список наоборот:

Для этого можно использовать синтаксис `slice`, метод `reverse()` или свободную функцию `reversed()`. Все три примера ниже дадут одинаковый вывод:

```
nums = [1, 2, 3, 4, 5]

a = nums[::-1]
b = list(reversed(nums))
```

```
nums.reverse()

print(a)
print(b)
print(nums)
```

Вывод:

```
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
```

Добавление элемента в список:

Для добавления элемента в конец списка используется метод `append()`. Для вставки элемента в произвольный индекс - метод `insert()`:

```
animals = list()            # []

animals.append('cat')       # ['cat']
animals.append('fox')       # ['cat', 'fox']

animals.insert(1, 'dog')    # ['cat', 'dog', 'fox']
animals.insert(0, 'cow')    # ['cow', 'cat', 'dog', 'fox']
```

Удаление элемента из списка:

Для этого есть два метода:

- `remove(x)` - удалит первый элемент, имеющий значение `x`
- `pop(i)` - удалит элемент с индексом `i`
  Пример:

```
names = ['Max', 'Sasha', 'Vova', 'Alexa', 'Nikita', 'John', 'Charles']

# удалим все имена, не заканчивающиеся на `a`
ends_not_with_a = list()
for name in names:
    if not name.endswith('a'):
        ends_not_with_a.append(name)
for name in ends_not_with_a:
    names.remove(name)
```

```
print(names) # ['Sasha', 'Vova', 'Alexa', 'Nikita']

# удалим последние три элемента
for i in range(3):
    names.pop(len(names) - 3)
print(names) # ['Alexa']

# преобразуем одиночный список в строку
name = ''.join(names) # это идиоматический способ преобразования списка в строку
print(names) # Alexa
```

Вывод программы:

```
['Sasha', 'Vova', 'Alexa', 'Nikita']
['Alexa']
Alexa
```

Очистка списка:

По аналогии с удалением отдельного элемента, список можно очистить целиком с использованием метода `clear()`

```
names = ['Max', 'Sasha', 'Vova', 'Alexa', 'Nikita', 'John', 'Charles']

if 'Max' in names:
    names.clear()
print(names)
```

Вывод:

```
[]
```

Сортировка списка

Для сортировки можно использовать функцию `sorted()` и метод `sort()`. Обе этих функции ведут себя одинаково - числа сортируются по возрастанию, строки - по алфавиту:

```
numbers = [3, 1, -1, 0, 8, 4, 12]
animals = ['cat', 'fox', 'dog', 'cow', 'horse']
```

```
# следующие 2 строки дают идентичный результат
numbers = sorted(numbers)
numbers.sort()

print(sorted(numbers))
print(sorted(numbers, reverse=True))   # сортировка в обратном порядке
print(sorted(animals))
print(sorted(animals, reverse=True))
```

Вывод:

```
-1 0 1 3 4 8 12
12 8 4 3 1 0 -1
cat cow dog horse fox
fox horse dog cow cat
```

Для сортировки сложных списков (например, списков *кортежей*) можно задать сортировочную функцию через синтаксис **лямбды**:

```
students = [('Max', 20), ('Sasha', 19), ('Nick', 21), ('Nikita', 20)]

# отсортируем по имени
print(sorted(students, key=lambda x: x[0]))

# по возрасту
print(sorted(students, key=lambda x: x[1]))

# по возрасту в обратном порядке
print(sorted(students, key=lambda x: x[1], reverse=True))
```

Вывод:

```
[('Max', 20), ('Nick', 21), ('Nikita', 20), ('Sasha', 19)]
[('Sasha', 19), ('Max', 20), ('Nikita', 20), ('Nick', 21)]
[('Nick', 21), ('Nikita', 20), ('Max', 20) , 'Sasha', 19)]
```