Python: списки | Я.Шпора

Списки — упорядоченные коллекции, которые могут хранить элементы произвольных типов.

Как объявить список

Перечислить элементы списка в квадратных скобках, разделяя их запятыми:

```
digits = [1, 2, 3, 4, 5]
fruits = ['яблоко', 'банан', 'вишня']

# Пустой список.
empty = []
```

С помощью встроенной функции `list()`. Эта функция может принимать в качестве аргумента коллекцию и преобразовывать её в список:

```
letters = list('яблоко')  # Строка - это коллекция текстовых символов.
# Строка будет преобразована в список символов:
# ['я', 'б', 'л', 'о', 'к', 'о']

# Пустой список.
another_empty = list()
```

Индексы и типы элементов в списке

У каждого элемента в списке есть порядковый номер — «индекс». Отсчёт индексов начинается с нуля: индекс первого элемента в списке — `0`.

Любой элемент списка можно получить по его индексу: `имя_списка[индекс]`.

```
vegetables = ['помидоры', 'огурцы', 'баклажаны']
                    |           |           |
# Индексы:          0           1           2

print(vegetables[0])
```

```
# Вывод в терминал: "помидоры" (индекс первого элемента - 0).
# Всего в списке vegetables три элемента, их индексы - 0, 1 и 2.
```

Элементы списка могут быть разного типа: строки, числа, другие коллекции, включая списки.

```
mixed_things = ['солёные помидоры', 15, ['огурцы', 'помидоры'], 3.14]
#                       |            |              |              |
# Типы элементов:      str          int          list          float
```

Работа со списками

Распаковка

Операция позволяет одним выражением присвоить значения элементов списка переменным:

```
week = [
    'Понедельник', 'Втроник', 'Среда',
    'Четверг', 'Пятница', 'Суббота', 'Воскресенье'
]

mon, tue, wed, thu, fri, sat, sun = week

print(wed)
# Вывод в терминал: Среда
```

Добавление элемента: list.append(<list>, element)

Добавляет новый элемент в конец списка:

```
vegetables = ['Помидоры', 'Огурцы', 'Кабачки']
list.append(vegetables, 'Баклажаны')
print(vegetables)

# Вывод в терминал: ['Помидоры', 'Огурцы', 'Кабачки', 'Баклажаны']
```

> Не следует присваивать переменной результат работы этой команды. Значением такой переменной будет `None`.

Расширение списка: list.extend(<list_1>, <list_2>)

Добавляет в конец списка `list_1` все элементы списка `list_2`. Список `list_2` сохранится в неизменном виде, а `list_1` изменится:

```
vegetables = ['Помидоры', 'Огурцы', 'Кабачки', 'Баклажаны']
vegetables_yields = [10, 15, 5, 12]

list.extend(vegetables, vegetables_yields)
print(vegetables)         # Список изменился.
print(vegetables_yields)  # Список не изменился.

# Вывод в терминал:
# ['Помидоры', 'Огурцы', 'Кабачки', 'Баклажаны', 10, 15, 5, 12]
# [10, 15, 5, 12]
```

Вставка элемента по индексу: list.insert(<list>, index, value)

Добавляет элемент со значением `value` на позицию `index`. Как следствие — индексы всех элементов, следующих за новым элементом, увеличатся на единицу:

```
vegetables = ['Помидоры', 'Огурцы', 'Кабачки']
# Индексы:          0         1         2

# Добавить новый элемент на позицию с индексом 1.
list.insert(vegetables, 1, 'Баклажаны')
print(vegetables)

# Вывод в терминал: ['Помидоры', 'Баклажаны', 'Огурцы', 'Кабачки']
# Индексы:               0           1          2           3
```

Удаление элемента: list.remove(<list>, value)

Читает список слева направо и удаляет первый элемент, значение которого совпадает с аргументом `value`. Если такого элемента нет в списке, возникнет ошибка:

```
vegetables = ['Помидоры', 'Баклажаны', 'Огурцы', 'Кабачки']
list.remove(vegetables, 'Огурцы')
print(vegetables)
# Вывод в терминал: ['Помидоры', 'Баклажаны', 'Кабачки']

# Попытка удалить несуществующий объект вызовет ошибку.
list.remove(vegetables, 'Патиссоны')
# Вывод в терминал: ValueError: list.remove(x): x not in list
```

Извлечение элемента: list.pop(<list>, index)

Удаляет из списка элемент с индексом `index` и возвращает его. Если индекс не указан — удаляет из списка последний элемент и возвращает его:

```
vegetables = ['Помидоры', 'Баклажаны', 'Огурцы', 'Кабачки']
vegetable = list.pop(vegetables, 2)
print(vegetable)
# Вывод в терминал: Огурцы

print(vegetables)
# Вывод в терминал: ['Помидоры', 'Баклажаны', 'Кабачки']
```

Индекс элемента: list.index(<list>, value, start, end)

Читает список слева направо и возвращает индекс первого найденного элемента со значением `value`. Диапазон поиска можно ограничить индексами `start` и `end`. Если элемент с заданным значением не найден, возникнет ошибка `ValueError`:

```
vegetables_yields = [10, 15, 5, 12]
result = list.index(vegetables_yields, 5)
print(result)
# Вывод в терминал: 2
```

Подсчёт количества: list.count(<list>, value)

Возвращает количество элементов со значением `value`:

```
yields = [10, 12, 15, 10, 8, 10]
quantity = list.count(yields, 10)
print(quantity)
# Вывод в терминал: 3
```

Сортировка списка: list.sort()

У этой команды есть параметр `reverse`, который определяет направление сортировки. По умолчанию `reverse = False`: элементы сортируются «по возрастанию», от меньшего к большему.

```
yields = [10, 12, 15, 10, 8, 10]
list.sort(yields)
print(yields)
# Вывод в терминал: [8, 10, 10, 10, 12, 15]
```

Инвертирование списка: list.reverse(<list>)

Меняет порядок элементов списка на противоположный:

```
yields = [4, 12, 15, 11, 8, 10]

list.reverse(yields)

print(yields)
# Вывод в терминал: [10, 8, 11, 15, 12, 4]
```

Копирование списка: list.copy(<list>)

Возвращает новый список, независимую копию исходного списка. Слова «независимая копия» означают, что если в дальнейшем изменить один из двух списков, то второй не изменится.

```
yields = [10, 12, 15, 10, 8, 10]
yields_copy = list.copy(yields)
```

```
# Применить сортировку к исходному списку.
list.sort(yields)
print(yields)
# Вывод в терминал: [8, 10, 10, 10, 12, 15]

# Проверить, изменилась ли копия.
print(yields_copy)
# Вывод в терминал: [10, 12, 15, 10, 8, 10]
# Список-копия остался неотсортированным.
```

Очистка списка: list.clear(<list>)

Удаляет из списка все элементы:

```
yields = [10, 12, 15, 10, 8, 10]
list.clear(yields)
print(yields)
# Вывод в терминал: []
```