

# Множества

Множество (от англ. set) — это изменяемая неупорядоченная коллекция неизменяемых и уникальных объектов.

Программист на Python использует множества, чтобы решать задачи, связанные с удалением повторяющихся элементов из коллекций, проверкой уникальности элементов и выполнением операций пересечения, объединения или разности между коллекциями

## Как создать множество

1. Литерально:

```
movie_ratings_set = {5.0, 4.1, 4.3, 4.7, 4.7, 3.8}  
print(movie_ratings_set)  
# Вывод в консоль: {5.0, 4.3, 3.8, 4.1, 4.7}  
print(type(movie_ratings_set))  
# Вывод в консоль: <class 'set'>
```

2. С помощью функции `set`:

```
movie_ratings = [5.0, 4.1, 4.3, 4.7, 4.7, 3.8]  
# Объявление множества через функцию set().  
movie_ratings_set = set(movie_ratings)  
print(movie_ratings_set)  
# Вывод в консоль: {5.0, 4.3, 3.8, 4.1, 4.7}  
# Элементов получилось меньше, чем в исходном списке:  
# неуникальные значения удалены.  
  
print(type(movie_ratings_set))  
# Вывод в консоль: <class 'set'>
```



Создать пустое множество можно только через функцию `set()`.

## Особенности множества

### Все элементы множества уникальны

При добавлении неуникальных элементов они не будут включены в множество.

```
movies = ['Матрица', 'Сеть', 'Хакеры', 'Трон', 'Тихушники', 'Сеть', 'Трон']  
uniq_movies = set(movies)  
print(uniq_movies)  
# Вывод в терминал: {'Сеть', 'Матрица', 'Тихушники', 'Хакеры', 'Трон'}
```

### Элементы множества — хешируемый объект

Элементами множества могут быть только неизменяемые объекты.

Множество, созданное из словаря, будет содержать только ключи исходного словаря.

```
movie_info = {'Матрица': 4.5, 'Трон': 4.8}
movie_names = set(movie_info)
print(movie_names)
# Вывод в терминал: {'Трон', 'Матрица'}
```

Из списка множество будет создано лишь в том случае, если элементы списка — неизменяемые объекты.

```
# Один из элементов исходного списка — список (ненехешируемый объект).
movie_ratings = [5.0, 4.1, [4.3, 4.7], 4.7, 3.8]
movie_ratings_set = set(movie_ratings)
print(movie_ratings_set)
# Вывод в терминал: TypeError: unhashable type: 'list'
```

## Работа с множествами

### Принадлежность объекта множеству

Чтобы определить, есть ли нужный элемент в множестве, применяют оператор `in`:

```
full_baggage = {'Диван', 'Чемодан', 'Саквояж', 'Картина', 'Корзина', 'Картонка'}

# Если собачонки нет...
if 'Маленькая собачонка' not in full_baggage:
    # ...устраиваем скандал:
    print('— Товарищи! Где собачонка?')
```

### Добавление элемента

Для добавления элемента в множество применяют метод `add()`. Если добавляемый элемент уже есть во множестве — ничего не произойдёт, множество не изменится.

```
maxim_toys = {'машинка', 'скакалка', 'кубики', 'пистолетик'}
print(id(maxim_toys))
# Вывод в терминал: 33482552

maxim_toys.add('самолётик')
# Проверим что элемент добавлен к исходному множеству.
print(id(maxim_toys))
# Вывод в терминал: 33482552
print(maxim_toys)
# Вывод в терминал:{'скакалка', 'машинка', 'самолетик', 'пистолетик', 'кубики'}

# Попробуем добавить ещё кубики в множество.
maxim_toys.add('кубики')
print(maxim_toys)
# Вывод в терминал:{'скакалка', 'машинка', 'самолетик', 'пистолетик', 'кубики'}
```

## Удаление элемента множества

Для удаления элемента есть три метода: `remove()`, `discard()` и `pop()`:

1. Методы `remove()` и `discard()` очень схожи: они удаляют элемент из множества. При попытке удалить несуществующий элемент методом `remove()` будет ошибка.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}

# Удаление существующего элемента
maxim_toys.remove('кубики')
print(maxim_toys)
# Вывод в терминал: {'машина', 'скакалка', 'пистолетик'}

maxim_toys.discard('машина')
print(maxim_toys)
# Вывод в терминал: {'скакалка', 'пистолетик'}

# Удаление несуществующих элемента
maxim_toys.discard('кукла')
print(maxim_toys)
# Вывод в терминал: {'скакалка', 'пистолетик'}

maxim_toys.remove('кукла')
print(maxim_toys)
# Вывод в терминал: KeyError: 'кукла'
```

2. Метод `pop()` удаляет и возвращает случайный элемент множества.

## Очистка множества

Для очистки существующего множества есть метод `clear()`:

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
maxim_toys.clear()
```

## Операции над множествами

### Пересечение

Вернуть новое множество, в котором собраны элементы, одновременно присутствующие в обоих исходных множествах, можно используя:

1. Оператор `&` (логическое `И`).

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}
overlap = maxim_toys & lera_toys
print(overlap)
# Вывод в терминал: {'кубики', 'скакалка'}
```

## 2. Метод `intersection()`.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}
overlap = maxim_toys.intersection(lera_toys)
print(overlap)
# Вывод в терминал: {'кубики', 'скакалка'}
```

## Объединение

Создать новое множество, содержащее все элементы исходных, можно используя:

### 1. Операнд `|` (логическое ИЛИ).

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}
unite = maxim_toys | lera_toys
print(unite)
# Вывод в терминал: {'кубики', 'кукла', 'скакалка', 'юла', 'машина', 'пистолетик'}
```

## 2. Метод `union()`.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}
unite = maxim_toys.union(lera_toys)
print(unite)
# Вывод в терминал: {'кубики', 'кукла', 'скакалка', 'юла', 'машина', 'пистолетик'}
```

## Разность

Вернуть новое множество, в которое войдут элементы первого множества, не пересекающиеся с элементами второго, можно используя:

### 1. Оператор `-` (обычный минус).

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}

diff = maxim_toys - lera_toys
print(diff)
# Вывод в терминал: {'машина', 'пистолетик'}
```

## 2. Метод `difference()`.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}

diff = maxim_toys - lera_toys
print(diff)
# Вывод в терминал: {'машина', 'пистолетик'}
```

## Симметрическая разность

Симметрической разностью двух множеств будет третье множество, каждый элемент которого принадлежит либо первому, либо второму множеству, но не их пересечению.

Для поиска симметрической разности используют:

1. Символ «карет» ^.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}

sym_diff = maxim_toys ^ lera_toys
print(sym_diff)
# Вывод в терминал: {'машина', 'пистолетик', 'юла', 'кукла'}
```

2. Метод `symmetric_difference()`.

```
maxim_toys = {'машина', 'скакалка', 'кубики', 'пистолетик'}
lera_toys = {'скакалка', 'кукла', 'кубики', 'юла'}

# Создаём множество, состоящее из элементов обоих множеств,
# с исключением пересекающихся:
sym_diff = maxim_toys.symmetric_difference(lera_toys)
print(sym_diff)
# Вывод в терминал: {'машина', 'пистолетик', 'юла', 'кукла'}
```

## Полезные ресурсы

[Раздел документации о множествах](#)