СОДЕРЖАНИЕ

Что такое множество?

Множество — это структура данных, которая представляет собой неупорядоченную коллекцию уникальных элементов.

Множество создаётся с помощью фигурных скобок {} или с помощью функции set(). В отличие от списка, элементы множества не имеют фиксированного порядка и не могут повторяться. Пример множества:

```
my_set = {1, 2, 3}
```

Ключевые отличия множества от списка:

• Множество хранит только уникальные элементы.

• Множество не сохраняет порядок элементов.

• Доступ к элементам множества возможен только через перебор, так как нет индексов.

Создание пустого множества:

```
empty_set = set()
# Важно: {} создаёт пустой словарь, а не множество
```

Множество можно получить из других коллекций, применив к ним функцию set(). Например, создадим множество из строки:

```
my_string = "hello"
my_set = set(my_string)
print(my_set)
```

Результат:

```
{'h', 'e', 'l', 'o'}
```

В этом примере:

• Строка "hello" содержит повторяющиеся символы 'l', но в результате работы set() они будут удалены.

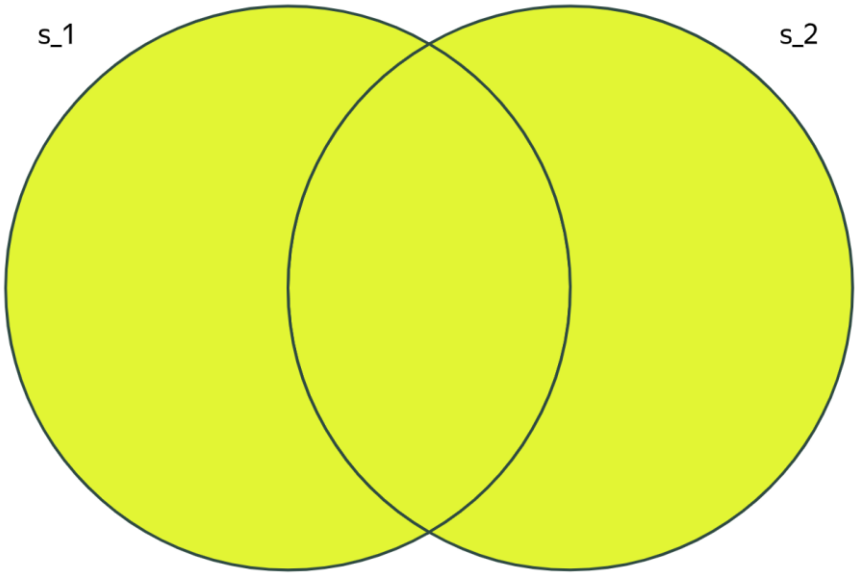
• В итоге множество будет содержать только уникальные символы.

Обратите внимание: порядок вывода элементов множества при выполнении примера может меняться произвольно из-за свойства неупорядоченности множества. Так, элементы множества не имеют индексов, и можно только проверить принадлежность элемента множеству.

Операции между множествами

Множества в Python позволяют выполнять следующие операции:

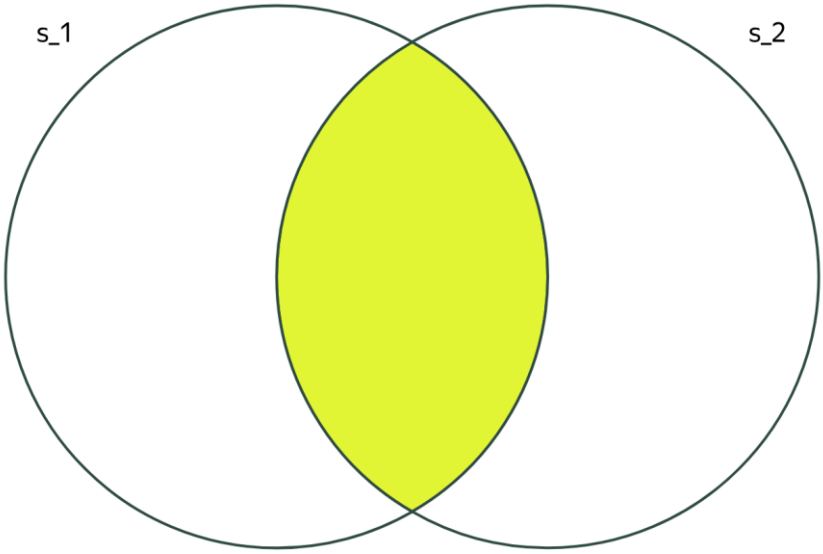
Объединение множеств. Возвращает множество, состоящее из элементов всех объединяемых множеств. Обозначается union() или |. Графическое представление операции:



Пример:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1 | set2  # {1, 2, 3, 4, 5}
```

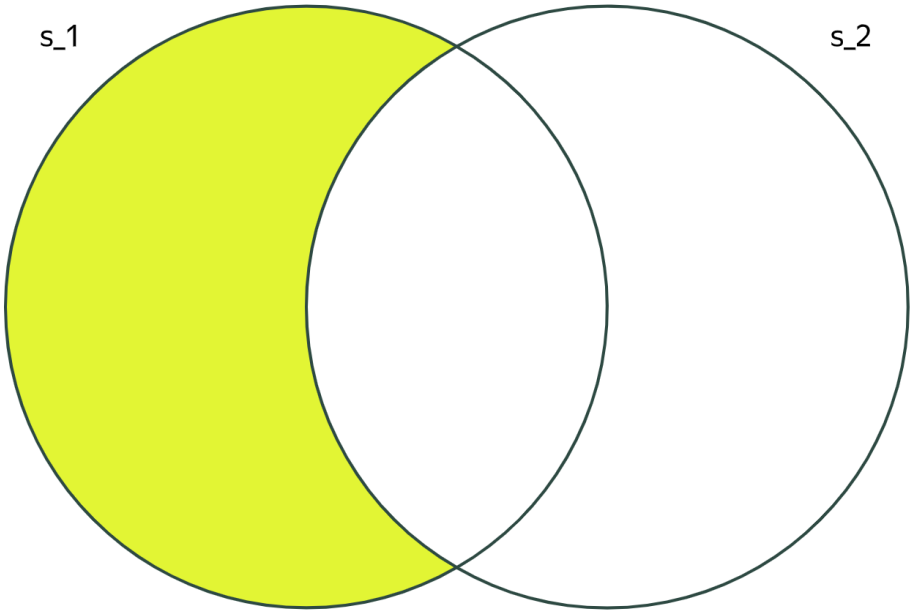
Пересечение множеств. Возвращает множество, состоящее из общих элементов пересекаемых множеств. Обозначается intersection или &. Графическое представление операции:



Пример:

```
intersection_set = set1 & set2  # {3}
```

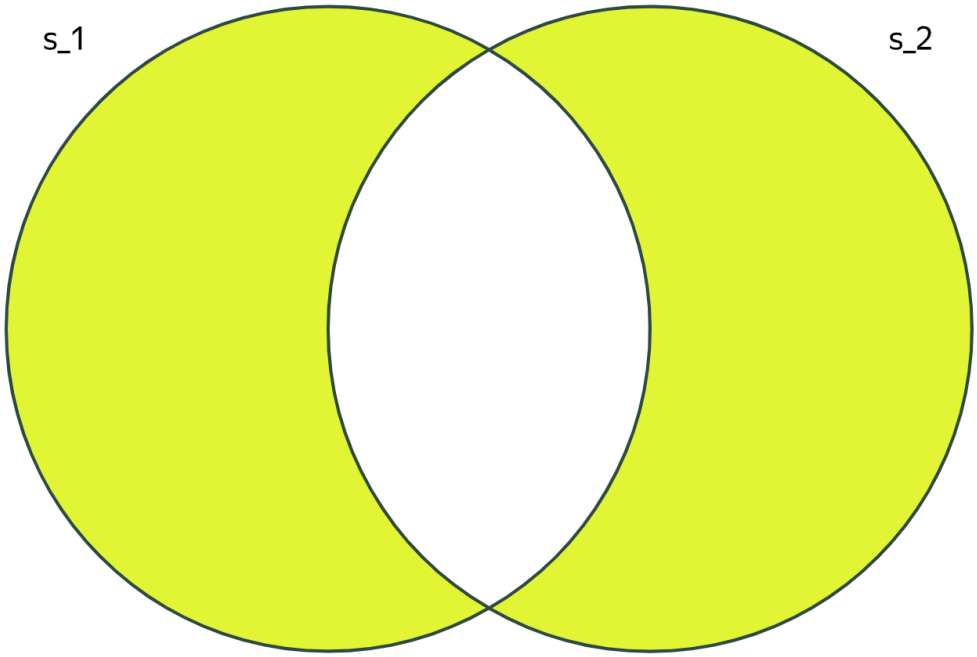
Разность множеств. Возвращает множество, где указаны элементы из первого множества, которых нет во втором — вычитаемом — множестве. Обозначается difference или -. Графическое представление операции:



Пример:

```
difference_set = set1 - set2  # {1, 2}
```

Симметричная разность множеств. Возвращает множество, состоящее из элементов, встречающихся в первом или втором множестве, но не в обоих сразу. Обозначается symmetric_difference или ^.



Пример:

```
symmetric_difference_set = set1 ^ set2  # {1, 2, 4, 5}
```

Методы множеств

Множества поддерживают ряд методов для работы с элементами. Вот основные методы и их примеры:

Синтаксис	Описание	Пример выполнения
set.add(elem)	Добавляет элемент в множество	my_set.add(5)
set.remove(elem)	Удаляет элемент из множества, если он существует	my_set.remove(3)
set.discard(elem)	Удаляет элемент, если он есть, не вызывает ошибку	my_set.discard(10)
set.pop()	Удаляет и возвращает случайный элемент	my_set.pop()
set.clear()	Удаляет все элементы множества	my_set.clear()
set.copy()	Создаёт копию множества	new_set = my_set.copy()
set.update(iterable)	Добавляет в множество элементы из итерируемого объекта	my_set.update([6, 7])