

## Практическая работа №15. Множества

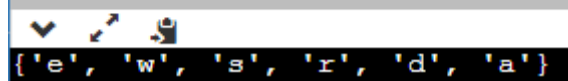
**Множество в языке Питон** — это структура данных, эквивалентная множествам в математике. Множество может состоять из различных элементов, порядок элементов в множестве неопределен. В множество можно добавлять и удалять элементы, можно перебирать элементы множества, можно выполнять операции над множествами (объединение, пересечение, разность). Можно проверять принадлежность элемента множеству.

В отличие от массивов, где элементы хранятся в виде последовательного списка, в множествах порядок хранения элементов неопределен (более того, элементы множества хранятся не подряд, как в списке, а при помощи хитрых алгоритмов). Это позволяет выполнять операции типа “проверить принадлежность элемента множеству” быстрее, чем просто перебирая все элементы множества.

### Задание множеств


Множество задается перечислением всех его элементов в фигурных скобках. Исключением является пустое множество, которое можно создать при помощи функции `set()`. Если функции `set` передать в качестве параметра список, строку или кортеж, то она вернёт множество, составленное из элементов списка, строки, кортежа. Например:

```
1 A = {1, 2, 3}
2 A = set('wasder')
3 print(A)
```



Каждый элемент может входить в множество только один раз, порядок задания элементов неважен. Например, программа, представленная ниже, выведет `True`, так как `A` и `B` – равные множества.


```
1 A = {1, 2, 3}
2 B = {3, 2, 3, 1}
3 print(A == B)
```



### Работа с элементами множеств


Узнать число элементов в множестве можно при помощи функции `len`.

```
1 A = {1, 2, 3, 3}
2 print(len(A))
```



Перебрать все элементы множества (в неопределенном порядке!) можно при помощи цикла `for`:

```
1 primes = {1, 2, 3, 4, 4, 5, 7}
2 for num in primes:
3     print(num, end=' |')
```



Проверить, принадлежит ли элемент множеству можно при помощи операции `in`, возвращающей значение типа `bool`. Аналогично есть противоположная операция `not in`.

```

1 A = {1, 2, 3}
2 print(1 in A, 4 not in A)

```

True True

Для добавления элемента в множество есть метод **add**:

```

1 A = {1, 2, 3}
2 print(A)
3 A.add(4)
4 print(A)

```

{1, 2, 3}  
{1, 2, 3, 4}

Для удаления элемента  $x$  из множества есть два метода: **discard** и **remove**. Их поведение различается только в случае, когда удаляемый элемент отсутствует в множестве. В этом случае метод **discard** не делает ничего, а метод **remove** генерирует исключение **KeyError**.

```

1 A = {1, 2, 3}
2 A.discard(4)
3 A.remove(4)

```

Traceback (most recent call last):  
File "main.py", line 3, in <module>  
A.remove(4)  
KeyError: 4

Наконец, метод **pop** удаляет из множества один случайный элемент и возвращает его значение. Если же множество пусто, то генерируется исключение **KeyError**.

```

1 A = {1, 2, 3}
2 t = A.pop()
3 print(t)

```

1

Из множества можно сделать список при помощи функции **list**.

```

1 A = {1, 2, 3}
2 print(A)
3 print(list(A))

```

{1, 2, 3}  
[1, 2, 3]

## Операции с множествами

$A \cup B$ <b>A.union(B)</b>	Возвращает множество, являющееся объединением множеств A и B.
$A \cup= B$ <b>A.update(B)</b>	Добавляет в множество A все элементы из множества B.

<b>A &amp; B</b> <b>A.intersection(B)</b>	Возвращает множество, являющееся пересечением множеств A и B.
<b>A &amp;= B</b> <b>A.intersection_update(B)</b>	Оставляет в множестве A только те элементы, которые есть в множестве B.
<b>A - B</b> <b>A.difference(B)</b>	Возвращает разность множеств A и B (элементы, входящие в A, но не входящие в B).
<b>A -= B</b> <b>A.difference_update(B)</b>	Удаляет из множества A все элементы, входящие в B.
<b>A ^ B</b> <b>A.symmetric_difference(B)</b>	Возвращает симметрическую разность множеств A и B (элементы, входящие в A или в B, но не в оба из них одновременно).
<b>A ^= B</b> <b>A.symmetric_difference_update(B)</b>	Записывает в A симметрическую разность множеств A и B.
<b>A &lt;= B</b> <b>A.issubset(B)</b>	Возвращает true, если A является подмножеством B.
<b>A &gt;= B</b> <b>A.issuperset(B)</b>	Возвращает true, если B является подмножеством A.
<b>A &lt; B</b>	Эквивалентно $A <= B$ and $A != B$
<b>A &gt; B</b>	Эквивалентно $A >= B$ and $A != B$

## Задачи

### №1

Дан список чисел, вводимый с клавиатуры. Определите, сколько в нем встречается различных чисел.

**Пример:**

**Входные данные:**

1 2 3 4 5 5 5 6

**Выходные данные:**

5

### №2

Даны два списка чисел, вводимых с клавиатуры. Посчитайте, сколько чисел содержится одновременно как в первом списке, так и во втором.

**Пример:**

**Входные данные:**

1 3 2 5 7 8

4 3 2 5

**Выходные данные:**

3

### №3

Даны два списка чисел. Найдите все числа, которые входят как в первый, так и во второй список и выведите их в порядке возрастания.

**Пример:**

**Входные данные:**

1 3 2 5 7 8

4 3 2 5

**Выходные данные:**

2 3 5

### №4

Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово **YES** (в отдельной строке), если это число ранее встречалось в последовательности или **NO**, если не встречалось.

**Пример:**

**Входные данные:**

1 2 3 2 3 4

**Выходные данные:**

NO

NO

NO

YES

YES

NO

### №5

Напишите функцию **superset()**, которая принимает 2 множества. Результат работы функции: вывод в консоль одного из сообщений в зависимости от ситуации:

1 - «Супермножество не обнаружено»

2 – «Объект {X} является чистым супермножеством»

3 – «Множества равны»

**Тесты:**

set\_1 = {1, 8, 3, 5}

set\_2 = {3, 5}

set\_3 = {5, 3, 8, 1}

set\_4 = {90, 100}

superset(set\_1, set\_2) -> Объект {8, 1, 3, 5} является чистым супермножеством

superset(set\_1, set\_3) -> Множества равны

superset(set\_2, set\_3) -> Объект {8, 1, 3, 5} является чистым супермножеством

superset(set\_4, set\_2) -> Супермножество не обнаружено

## №6

Напишите программу, которая удаляет из строки все повторяющиеся символы. На вход программы подаётся строка, содержащая символы таблицы ASCII. Программа должна вывести исходную строку, из которой удалены все повторяющиеся символы.

### Примеры

**входные данные**

abc13a1b2z3c

**выходные данные**

abc132z

**входные данные**

QWasd123

**выходные данные**

QWasd123

## №7

Напишите программу, которая определяет правильность записи целого числа в восьмеричной системе счисления. На вход программы поступает символьная строка. Программа должна вывести ответ 'YES', если строка представляет собой правильную запись целого числа в восьмеричной системе счисления, и 'NO', если запись ошибочна.

### Примеры

**входные данные**

12345

**выходные данные**

YES

**входные данные**

1a234

**выходные данные**

NO