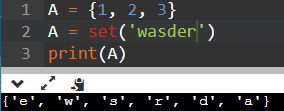
**Практическая работа №15. Множества**

**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.

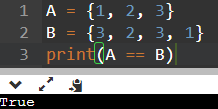
В отличие от массивов, где элементы хранятся в виде последовательного списка, в множествах порядок хранения элементов неопределен (более того, элементы множества хранятся не подряд, как в списке, а при помощи хитрых алгоритмов). Это позволяет выполнять операции типа “проверить принадлежность элемента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

**Задание множеств**

Множество задается перечислением всех его элементов в фигурных скобках. Исключением явлеется 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:

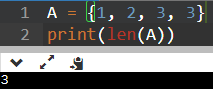


Каждый элемент может входить в множество только один раз, порядок задания элементов неважен. Например, программа, представленная ниже, выведет True, так как А и В – равные множества.

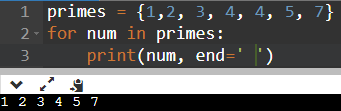


**Работа с элементами множеств**

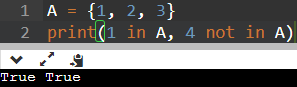
Узнать число элементов в множестве можно при помощи функции **len**.



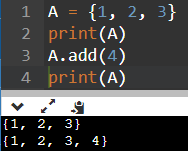
Перебрать все элементы множества (в неопределенном порядке!) можно при помощи цикла **for**:



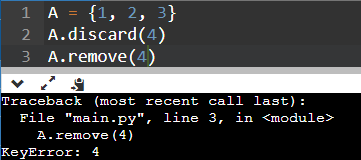
Проверить, принадлежит ли элемент множеству можно при помощи операции **in**, возвращающей значение типа **bool**. Аналогично есть противоположная операция **not in**.



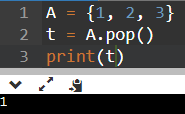
Для добавления элемента в множество есть метод **add**:



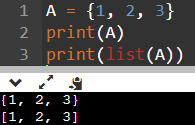
Для удаления элемента x из множества есть два метода: **discard** и **remove**. Их поведение различается только в случае, когда удаляемый элемент отсутствует в множестве. В этом случае метод **discard** не делает ничего, а метод **remove** генерирует исключение **KeyError**.



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



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



**Операции с множествами**

|  |  |
| --- | --- |
| **A | B A.union(B)** | Возвращает множество, являющееся объединением множеств A и B. |
| **A |= B A.update(B)** | Добавляет в множество A все элементы из множества B. |
| **A & B A.intersection(B)** | Возвращает множество, являющееся пересечением множеств A и B. |
| **A &= B A.intersection\_update(B)** | Оставляет в множестве A только те элементы, которые есть в множестве B. |
| **A - B A.difference(B)** | Возвращает разность множеств A и B (элементы, входящие в A, но не входящие в B). |
| **A -= B A.difference\_update(B)** | Удаляет из множества A все элементы, входящие в B. |
| **A ^ B A.symmetric\_difference(B)** | Возвращает симметрическую разность множеств A и B (элементы, входящие в A или в B, но не в оба из них одновременно). |
| **A ^= B A.symmetric\_difference\_update(B)** | Записывает в A симметрическую разность множеств A и B. |
| **A <= B A.issubset(B)** | Возвращает true, если A является подмножеством B. |
| **A >= B A.issuperset(B)** | Возвращает true, если B является подмножеством A. |
| **A < B** | Эквивалентно A <= B and A != B |
| **A > 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