

## 11.2 Основные приемы работы с данными множественного типа

Количество элементов в множестве определяется с помощью функции `len()`.

Для проверки принадлежности элемента множеству используется зарезервированное слово `in`:

### Демонстрация `len()` и `in`

```
b = {1, 3, 3, 5, 9}
print('кол-во элементов: ', len(b))           # результат 4
print('наличие элемента 4 в b: ', 4 in b)      # результат False
print('отсутствие элемента 4 в b: ', not 4 in b) # результат True
print('отсутствие элемента 3 в b: ', 3 not in b) # результат False
```

Множества легко вывести на экран (порядок вывода элементов в множестве может быть непредсказуем)

### Печать множеств

```
a = set()
b = {1, 3, 3, 5, 9}
c = set('hello world!')
print(a, b, c, sep='\n')
```

### Результат-1

```
set()
{1, 3, 5, 9}
{'!', 'h', 'o', 'r', 'l', 'w', ' ', 'd', 'e'}
```

### Результат-2

```
set()
{1, 3, 5, 9}
{'d', ' ', 'l', 'h', 'o', 'e', 'r', '!', 'w'}
```

**Пример 11.4** (поэлементная печать множества). Для печати элементов множества можно использовать цикл `for`:

### Печать множеств

```
word = set('programming')
for c in word:
    print(c, end='')
```

### Примеры результатов

```
armpgnio # первый результат
prgonmai # второй результат
```

**Пример 11.5 (преобразование множеств).** Множества можно преобразовывать в строку, список или кортеж, используя соответствующие методы.

**Преобразование множества к другому типу**

```
Set = set('programming') # множество
List = list(Set)          # список
Tuple = tuple(Set)        # кортеж
Str = str(Set)            # строка
```

**Методы добавления, удаления элементов из множества**

Метод	Описание
add(x)	добавление элемента $x$ в множество
discard(x)	удаление элемента $x$ из множества
remove(x)	удаление элемента $x$ из множества. Если удаляемый элемент отсутствует в множестве, то генерируется исключение <code>KeyError</code>
pop()	удаляет из множества первый элемент и возвращает его значение. Если множество пусто, то генерируется исключение <code>KeyError</code> . <b>Примечание:</b> так как множество не упорядочено, какой элемент будет первым, неизвестно
clear()	очистка множества

**Пример 11.6 (добавление элемента в список и очистка списка).** Метод `add()` добавляет элемент в множество.

**Добавление элемента в список и очистка списка**

```
s = {1, 2, 10}
print(s) # {1, 2, 10}
s.add(5)
print(s) # {1, 2, 10, 5}
s.add(7)
print(s) # {1, 2, 5, 7, 10}
s.clear()
print(s) # set()
```

**Пример 11.7 (удаление случайного элемента).** Использование метода `pop()` позволяет не только удалить элемент из множества, но и запомнить значение удаленного элемента. Если применить метод `pop()` к пустому списку, то генерируется исключение `KeyError` — несуществующий ключ (во множестве, в данном случае). Для обработки исключений используется конструкция `try-except`.

#### Удаление случайного элемента

```
s1 = {-1, 2, 10}
print(s1)      # {2, 10, -1}
s1.pop()
print(s1)      # {10, -1}
d = s1.pop()
print(s1, 'удалено число', d)    # {-1} удалено число 10
# обработка исключения, если происходит удаление из пустого списка
s1.clear()      # очистка списка
try:
    s1.pop()     # инструкция, которая может породить исключение
except Exception:
    print('Error') # перехват исключения
print(s1)      # {}
```

**Пример 11.8 (сравнение методов удаления элемента из списка).** Методы `discard(x)` и `remove(x)` удаляют элемент  $x$  из множества. Метод `remove(x)` генерирует исключение `KeyError`, при попытке удалить отсутствующий элемент.

#### Удаление элемента из списка

```
s = {1, 2, 10}
s.discard(2)
print(s)      # {1, 10}
s.discard(5) # попытка удалить несущ-щий элемент ошибку не вызывает
print(s)      # {1, 10}
#-----
s = {1, 2, 10}
s.remove(2)
print(s)      # {1, 10}
'''
s.remove(5) # попытка удалить несущ-щий элемент вызывает ошибку
print(s)
'''
```

### Функции с логическим результатом

<code>b.isdisjoint(c)</code>	проверяет нет ли пересечения множеств (True, если нет)
<code>b.issubset(c)</code> <code>b &lt;= c</code>	$b \subset c$ (True, если все элементы $b$ принадлежат $c$ )
<code>b.issuperset(c)</code> <code>b &gt;= c</code>	$b \supset c$ (True, если все элементы $c$ принадлежат $b$ )

**Пример 11.9** (демонстрация методов `isdisjoint()`, `issubset()`, `issuperset()`). Продемонстрируем работу методов `isdisjoint()`, `issubset()`, `issuperset()`.

#### Демонстрация методов `isdisjoint()`, `issubset()`, `issuperset()`

```
A = {1, 2, 10}
B = {1, 12, 100}
C = {11, 14}
print(A.isdisjoint(B)) # False. Нет ли пересечения?
print(A.isdisjoint(C)) # True
#-----
B = {1, 12, 100}
C = {1, 100}
print(B.issubset(C))    # False. Все элементы b принадлежат c
print(B.issuperset(C))  # True. Все элементы c принадлежат b
```

### Объединение, пересечение, разность множеств

<code>a.update(b)</code>	добавление элементов множества $b$ в $a$
<code>F = b.union(c, d)</code> <code>F = b   c   d</code>	объединение множеств ( $F = b \cup c \cup d$ ).
<code>F = b.intersection(c, d)</code> <code>F = b &amp; c &amp; d</code>	пересечение ( $F = b \cap c \cap d$ ).
<code>F = A.difference(B)</code> <code>F = A - B</code>	разность множеств $A$ и $B$ (элементы, входящие в $A$ , но не входящие в $B$ )

**Пример 11.10** (метод `update()`). Метод `update()` служит для добавления элементов одного множества в другое.

#### Демонстрация метода `update()`

```
A = {'a', 'c'}
B = {1, 12, 100}
A.update(B) # добавление элементов множества B в A
print(A)    # {1, 'c', 100, 'a', 12}
```

**Пример 11.11** (объединение, пересечение и разность множеств). Продемонстрируем работу методов для объединения, пересечения нескольких множеств, а также разности множеств.


#### Объединение множеств


```
A = {'a', 'c'}
B = {1, 12, 100}
C = {1, 100}
D = {-5, -7}
S1 = B.union(C, D)
print(S1)          # {1, 100, -7, -5, 12}
S11 = set.union(B, C, D)
print(S11)         # {1, 100, -7, -5, 12}
```


#### Пересечение множеств и разность множеств


```
A = {12, 5, 100, -3}
B = {1, 12, 100}
C = {1, 100}
D = {-5, -7, 1}
P = B.intersection(C, D) # пересечение (или P = B & C & D)
print(P) # {1}
#-----
R = A.difference(B) # разность (или R = A - B)
print(R) # {5, -3}
```

### 11.3 Задачи


 **11.1.** Написать программу, выделяющую из некоторого заданного множества подмножество четных чисел.

 **11.2.** Создать множество согласных букв, входящих в заданную строку. Исходная строка может содержать символы латинского алфавита, пробелы и знаки препинания.


 **11.3.** Дано множество символов, состоящее из различных строчных латинских букв. Напечатать элементы данного множества в алфавитном порядке.

 **11.4.** Дан список  $A$ , содержащий различные целые числа; дан список  $B$ , содержащий различные целые числа. Выяснить, верно ли, что оба списка отличаются не более, чем порядком следования чисел.

**Указание.** Постройте множества, состоящие из элементов списков.


 **11.5.** Из двух исходных списков составить новый список, содержащий только общие элементы (без повторений).

---

 **11.6.** Используя алгоритм «решето Эратосфена», найти все простые числа из промежутка  $[2, n]$ .

#### Алгоритм:

- 1) создать исходное множество натуральных чисел;
- 2) создать пустое множество простых чисел;
- 3) в переменную `nxt` поместить первое простое число;
- 4) пока  $\langle$ исходное множество не пусто $\rangle$  делать
  - 4.1) добавить в множество простых чисел текущее простое число `nxt`;
  - 4.2) удалить из множества натуральных чисел все кратные `nxt`;
  - 4.3) получить следующее простое число — новое значение `nxt` (первое, оставшееся в исходном множестве после выполнения 4.2).

 **11.7.** Используя алгоритм «решето Эратосфена», найти и напечатать в порядке убывания все простые числа из промежутка  $[m, n]$ .

---