Тема 3.7 МНОЖЕСТВА

Множество в Python (set) – «контейнер», содержащий не повторяющиеся элементы в случайном порядке, т.е. это структура данных, эквивалентная множествам в математике.

В Python множество – это неупорядоченная коллекция элементов со следующими особенностями:

- все элементы множества уникальны. Иными словами в множестве не может быть двух одинаковых элементов;
- элементы множества являются неизменными объектами;
- элементы множества могут быть разных типов.

В качестве элементов могут выступать любые неизменяемые объекты, такие как числа, символы, строки. Множество может состоять из различных элементов, порядок элементов в множестве не определён.

Создать объект типа «множество» можно следующими способами:

1) С помощью фигурных скобок **{ }**

```
x = { 0, 0, 1, 2, 3 }   # x = {0, 1, 2, 3} - элементы в множестве не повторяются
y = { 'a', "textbook", 5 }     # y = {'a', 'textbook', 5}
```

2) С помощью функции **set( )**

```
x = set(['a', 5, "textbook", 5])        # x = {5, 'textbook', 'a'}
y = set('Hello!')                       # y = {'l', 'o', 'e', '!', 'H'}
z = set([0,1,2,3,4,5])                  # z = {0, 1, 2, 3, 4, 5}
```

Функция **set()** позволяет превратить в множество объекты других типов.

Каждый элемент может входить в множество только один раз, порядок задания элементов неважен. Множества удобно использовать для удаления повторяющихся элементов.

Пустое множество можно создать при помощи функции **set()** без перечисления элементов.

```
a = set()
```

Множества — неупорядоченная и не индексируемая последовательность, т.е. нельзя получить доступ к элементам множества по индексу, так как они не упорядочены, а элементы без индекса. Но можно проходить по множеству с помощью цикла **for** или уточнять есть ли значение в множестве, используя оператор **in**, или он отсутствует, используя оператор **not in**:

- оператор in – проверка элемента на вхождение в множество;
- оператор not in - проверка отсутствия элемента.

Пример перебора вхождения элементов множества с помощью цикла:

```
a = {0, 1, 2}
for в in a:
    print(в)
0
1
2
```

Язык Python поддерживает следующие операции над множествами:

1) *объединение множеств* - **A | B** или **A.union(B) -** возвращает множество, являющееся объединением множеств A и B

```
A = {0, 1, 2, 3}
B = {4, 3, 2, 1}
C = A.union(B)
print(C)          # {0, 1, 2, 3, 4}
```

2) *разность множеств* - **A - B** или **A.difference(B) -** возвращает разность множеств A и B (элементы, входящие в A, но не входящие в B)

```
A = {0, 1, 2, 3}
B = {4, 3, 2, 1}
C= A.difference(B)
print(C)          # {0}
```

3) *пересечение множеств* - **A & B** или **A.intersection(B) -** возвращает множество, являющееся пересечением множеств A и B

```
A = {0, 1, 2, 3}
B = {4, 3, 2, 1}
C = A.intersection(B)
print(C)          # { 1, 2, 3}
```

4) *симметричная разность* - **A ^ B** или **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 = { 1, 3, 5 }
B = { 2, 3, 8 }
C = A.symmetric_difference(B)
print(C)    # C = {1, 2, 5, 8}
```

Существует еще ряд операций для работы с множествами таблица 1.

Таблица 1 – Дополнительные операции для работы с множествами

Название	Назначение
A |= B A.update(B)	Добавляет в множество A все элементы из множества B.
A &= B A.intersection_update(B)	Оставляет в множестве A только те элементы, которые есть в множестве B.
A -= B A.difference_update(B)	Удаляет из множества A все элементы, входящие в B.
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.

Для управления содержимым множеств в языке Python присутствуют специальные методы, дающие возможность добавлять и удалять отдельные элементы (таблица 2).

Таблица 2 – Методы для работы с множествами

Название	Назначение	Пример
len()	число элементов в множестве (длина множества).	a = {0, 1, 2, 3} print(len(a)) # 4
add ()	добавление элемента	a = {0, 1, 2, 3} a.add(4) print(a) # {0, 1, 2, 3, 4}
update()	добавление всех элементов одного множества в другое	a = {0, 1, 2, 3} b = {4, 3, 2, 1} a.update(b) print(a) # {0, 1, 2, 3, 4}
remove()	удаление элемента с генерацией исключения в случае, если такого элемента нет	a = {0, 1, 2, 3} a.remove(3) print(a) # {0, 1, 2}
discard()	удаление элемента без генерации исключения, если элемент отсутствует	a = {0, 1, 2, 3} a. discard(3) print(a) # {0, 1, 2}
clear()	полная очистка	a = {0, 1, 2, 3} a.clear() print(a) # set()
copy()	копирование содержимое одного множества в другую переменную	a = {0, 1, 2, 3} в = a.copy() print(в) # {0, 1, 2, 3}

Тип **frozen set** является видом множества, которое не может быть изменено. Для его создания используется функция **frozenset().**

В функцию frozenset передается набор элементов - список, кортеж, другое множество. В такое множество нельзя добавить новые элементы, как и удалить из него уже имеющиеся.