Теоретический материал: множества

Сайт: Дистанционная подготовка
Курс: Д. П. Кириенко. Программирование на языке Python (школа 179 г. Москвы)
Book: Теоретический материал: множества
Printed by: maung myo
Date: Воскресенье 4 Март 2018, 01:21

Table of Contents

Множества

Множество в языке Питон — это структура данных, эквивалентная множствам 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 элементу множества.

В отличии от массивов, где элементы хранятся в виде последовательного списка, в множествах порядок хранения элементов неопределен (более того, элементы множества храняться 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.

Элементами множества может быть любой неизменяемый тип данных: числа, строки, кортежи. Изменяемые типы данных не могут быть элементами множества, в частности, нельзя сделать элементом множества список (но можно сделать кортеж) или другое множество. Требование неизменяемости элементов множества накладывается особенностями представления множества в памяти компьютера.

Задание множеств

Множество задается перечислением всех его элементов в фигурных скобках. Например:

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

Исключением явлеется пустое множество, которое можно создать при помощи функции set(). Если функции set передать в качестве параметра список, строку или кортеж, то она вернет множество, составленное из элементов списка, строки, кортежа. Например:

```
A = set('qwerty')
print(A)
```

выведет {'e', 'q', 'r', 't', 'w', 'y'}.

Каждый элемент может входить в множество только один раз, порядок задания элементов не важен. Например, программа:

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

выведет True, так как A и B — равные множества.

Каждый элемент может входить в множество только один раз. set('Hello') вернет множество из четырех элементов: {'H', 'e', 'l', 'o'}.

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:

```
C = {1, 2, 3, 4, 5}
for elem in C:
    print(elem)
```

Проверить, принадлежит ли элемент множеству можно при помощи операции in, возвращающей значение типа bool:

```
i in A
```

Аналогично есть противоположная операция not in.

Для добавления элемента в множество есть метод add:

```
A.add(x)
```

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 множетсва один случайный элемент и возвращает его значение. Если же множество пусто, то генерируется исключение KeyError.

Из множества можно сделать список при помощи функции list.

Перебор элементов множества

При помощи цикла for можно перебрать все элементы множества:

```
Primes = {2, 3, 5, 7, 11}
for num im Primes:
    print(num)
```

Операции с множествами

С множествами в питоне можно выполнять обычные для математики операции над 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