**Лекция 11**

**Множества в python**

**1. Что такое множество?**

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

Проще говоря, множество в python - "контейнер", содержащий не повторяющиеся элементы в случайном порядке.

В отличие от массивов, где элементы хранятся в виде последовательного списка, в множествах порядок хранения элементов не определен (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.

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.

**2. Как создать множество?**

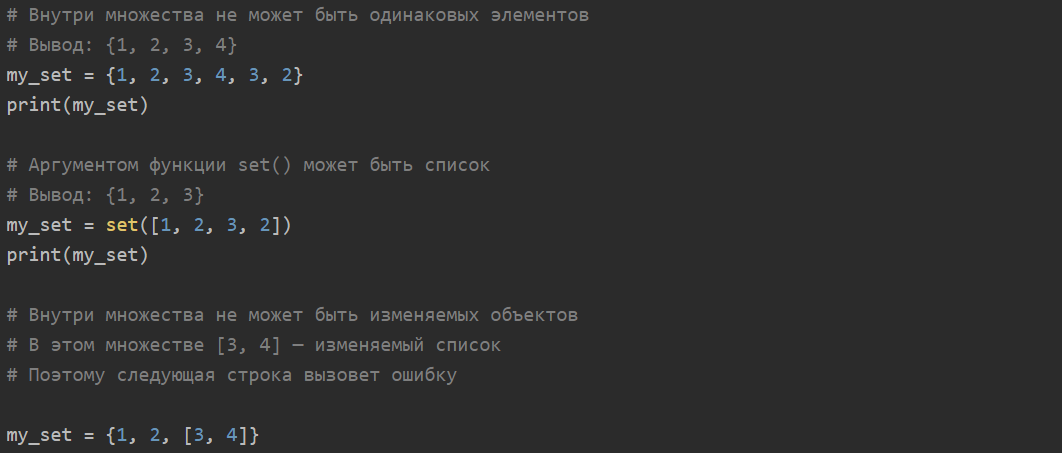
Множество объявляется так: элементы помещаются в фигурные скобки {} и разделяются запятыми. Сделать это можно и с помощью встроенной функции set().

Внутри множества может быть любое количество элементов любого типа (целые числа, числа с плавающей точкой, кортежи, строки и т. д.). Внутрь множества нельзя помещать элементы изменяемыхтипов: списки, другие множества или словари.



Вывод:

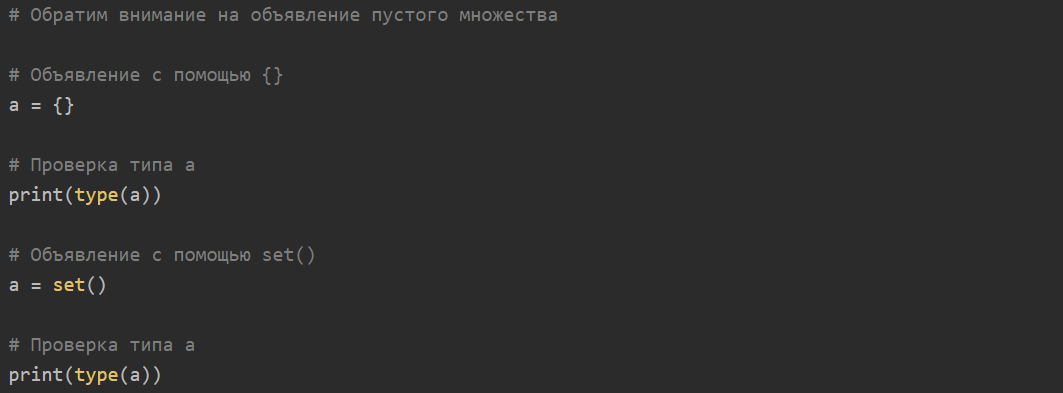
{1, 2, 3}  
{1.0, (1, 2, 3), 'Привет'}



Вывод:

{1, 2, 3, 4}  
{1, 2, 3}  
Traceback (most recent call last):  
  File "<string>", line 15, in <module>  
    my\_set = {1, 2, [3, 4]}  
TypeError: unhashable type: 'list'

Пустые фигурные скобки {} — это словарь. Чтобы объявить пустое множество, нужно использовать функцию set() без аргументов.



Вывод:

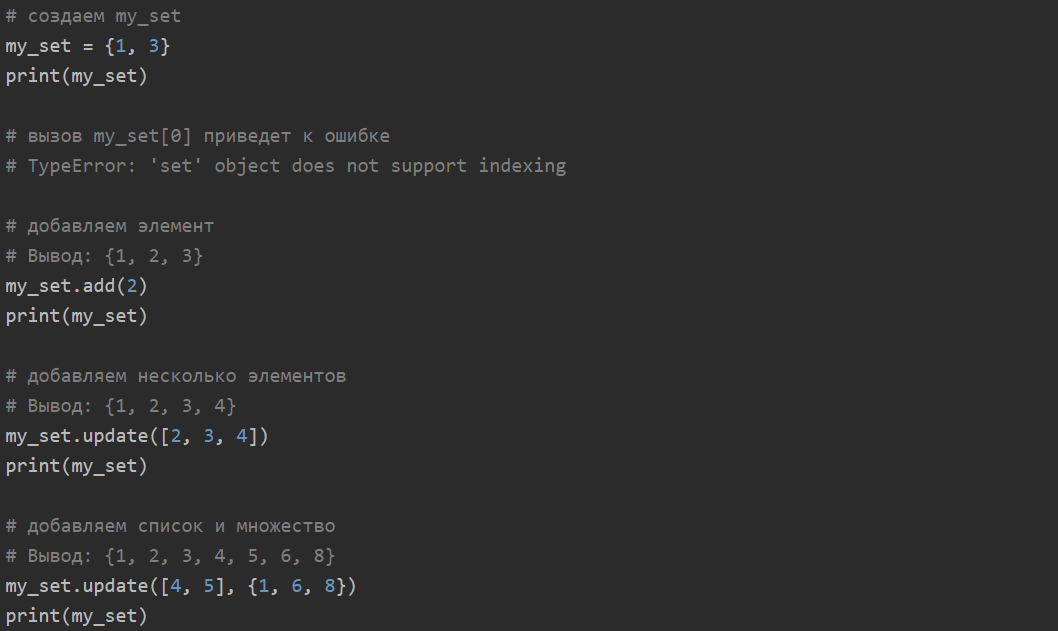
<class 'dict'>  
<class 'set'>

**3. Как изменять множество?**

Множества изменяемы и не упорядочены. Поэтому в индексации нет никакого смысла.

Так что получить доступ к элементам с помощью индексов или срезов не получится. Множества просто не поддерживают эти операции.

Чтобы добавить один элемент, нужно использовать метод add(). Если нужно добавить несколько элементов — метод update(). Метод update() принимает в качестве аргументов [кортежи](https://codechick.io/tutorials/python/data-types-tuples), [списки](https://codechick.io/tutorials/python/data-types-lists) или другие множества. Во всех случаях следует избегать одинаковых элементов.



Вывод:

{1, 3}  
{1, 2, 3}  
{1, 2, 3, 4}  
{1, 2, 3, 4, 5, 6, 8}

**4. Как удалять элементы из множества?**

Определенный элемент множества можно удалить с помощью методов discard() и remove().

Разница между ними вот в чем. Если элемент, который вы хотите удалить с помощью discard(), отсутствует в множестве — оно не изменится. А вот remove() вызовет ошибку, если элемента нет в множестве.



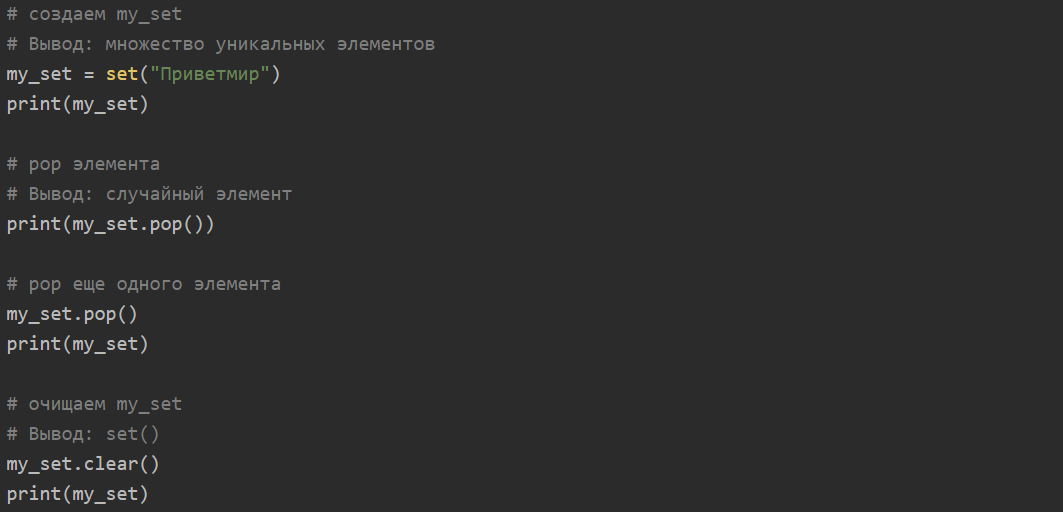
Вывод:

{1, 3, 4, 5, 6}  
{1, 3, 5, 6}  
{1, 3, 5}  
{1, 3, 5}  
Traceback (most recent call last):  
  File "<string>", line 28, in <module>  
KeyError: 2

Удалить и вернуть элемент мы можем с помощью метода pop().

Так как множество — неупорядоченный тип данных, невозможно определить, какой из элементов будет удален. Это произойдет случайным образом.

Удалить все элементы из множества можно с помощью метода clear().



Вывод:

{'П', 'и', 'р', 'т', 'е', 'м', 'в'}  
П  
{'р', 'т', 'е', 'м', 'в'}  
set()

**5. Операции со множествами**

С множествами можно выполнять множество операций: находить объединение, пересечение...

* len(s) - число элементов в множестве (размер множества).
* x in s - принадлежит ли x множеству s.
* set.isdisjoint(other) - истина, если set и other не имеют общих элементов.
* set == other - все элементы set принадлежат other, все элементы other принадлежат set.
* set.issubset(other) или set <= other - все элементы set принадлежат other.
* set.issuperset(other) или set >= other - аналогично.
* set.union(other, ...) или set | other | ... - объединение нескольких множеств.
* set.intersection(other, ...) или set & other & ... - пересечение.
* set.difference(other, ...) или set - other - ... - множество из всех элементов set, не принадлежащие ни одному из other.
* set.symmetric\_difference(other); set ^ other - множество из элементов, встречающихся в одном множестве, но не встречающиеся в обоих.
* set.copy() - копия множества.

И операции, непосредственно изменяющие множество:

* set.update(other, ...); set |= other | ... - объединение.
* set.intersection\_update(other, ...); set &= other & ... - пересечение.
* set.difference\_update(other, ...); set -= other | ... - вычитание.
* set.symmetric\_difference\_update(other); set ^= other - множество из элементов, встречающихся в одном множестве, но не встречающиеся в обоих.
* set.add(elem) - добавляет элемент в множество.
* set.remove(elem) - удаляет элемент из множества. KeyError, если такого элемента не существует.
* set.discard(elem) - удаляет элемент, если он находится в множестве.
* set.pop() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
* set.clear() - очистка множества.

Также существует frozenset.

Единственное отличие set от frozenset заключается в том, что set - изменяемый тип данных, а frozenset - нет.