**Множества**

Класс set (множество) — это одна из ключевых структур данных в Python. Она представляет собой неупорядоченную коллекцию уникальных элементов. Класс set, в некоторой степени, соответствует математическому множеству. Многие широко используемые математические операции, применимые к множествам, существуют и в Python. Часто вычисления, производимые над множествами, оказываются гораздо быстрее, чем альтернативные операции со списками.

**Создать объект set в Python можно двумя путями:**

1. Использовать фигурные скобки {}
2. Использовать встроенную функцию set()

Множество создается при размещении всех элементов внутри фигурных скобок {}, как показано на примере ниже.

s1 = {} # Это создаст пустое множество

s2 = {1, 'pythonru', 20.67}

Еще один способ создать (или определить) множество Python — использовать функцию set(). Пример ниже.

s1 = set() # Это создаст пустое множество

s2 = set({1, 'pythonru', 20.67})

Нет ограничений на количество элементов в объекте set, но запрещено добавлять элементы изменяемых типов, такие как список или словарь. Если попробовать добавить список (с набором элементов), интерпретатор выдаст ошибку.

**Добавление элементов в множества Python**

Объекты set в Python поддерживают добавление элементов двумя путями: по одному с помощью метода add() или группами с помощью update(). Оба описаны дальше.

Один элемент можно добавить с помощью метода add(). Такой код выглядит следующим образом.

set1 = {1, 3, 4}

set1.add(2)

print(set1)

{1, 2, 3, 4}

Добавление нескольких элементов в множество Python

Больше одного элемента можно добавить с помощью update(). Код следующий.

set2 = {1, 2, 3}

set2.update([4, 5, 6])

print(set2) # {1, 2, 3, 4, 5, 6}

**Удаление элементов из множеств Python**

Один или несколько элементов можно удалить из объекта set с помощью следующих методов. Их отличие в виде возвращаемого значения.

remove()

discard()

pop()

remove()

Метод remove() полезен в тех случаях, когда нужно удалить из множества конкретный элемент и вернуть ошибку в том случае, если его нет в объекте.

Следующий код показывает метод remove() в действии.

set1 = {1, 2, 3, 4, 'a', 'p'}

set1.remove(2)

print(set1)

{1, 3, 4, 'a', 'p'}

set1.remove(5) # Error element not found

**discard()**

Метод discard() полезен, потому что он удаляет конкретный элемент и не возвращает ошибку, если тот не был найден во множестве.

set1 = {1, 3, 4, 'a', 'p'}

set1.discard('a')

print(set1)

# {1, 3, 4 'p'}

set1.discard(6)

print(set1)

# {1, 3, 4, 'p'}

**pop()**

Метод pop() удаляет по одному элементу за раз в случайном порядке. Set — это неупорядоченная коллекция, поэтому pop() не требует аргументов (индексов в этом случае). Метод pop() можно воспринимать как неконтролируемый способ удаления элементов по одному из множеств в Python.

set1 = {1, 3, 4, “p”}

set1.pop() # случайный элемент будет удален (ваш результат может отличаться)

**Методы множеств Python**

У объектов set есть несколько встроенных методов. Увидеть их все можно с помощью команды dir(). dir(object) в Python показывает самые важные атрибуты разных типов объектов

Вот что выдаст функция для объекта set в Python.

set1 = { 1, 2, 3, 4}

dir(set1)

['\_\_and\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr