# Урок 15.04. Множества. Функция set

Как мы знаем программирование и математика очень тесно связанные вещи. И python тут не исключение, и рассмотрим такую задачу.

Задача «Уникальные числа».

У нас есть список из 10 случайных чисел. Нас нужно написать программу, которая выведет на экран все уникальные числа, т.е. числа без повторов.

import random

number\_list = [random.randint(1,10) for \_ in range(10)]

print(number\_list)

new\_list = []

for i in number\_list:

if i not in new\_list:

new\_list.append(i)

print(new\_list)

Такая задача не представляет для нас проблем. А теперь давайте подумаем, как мы можем оптимизировать нашу программу уже с имеющимися знаниями.

Давайте вспомним, что наш словарь содержит ключи, которые являются уникальными и не повторяются, может нам стоит этим воспользоваться, тогда попробуем преобразовать наш список в словарь с помощью команды ***dict***.

import random

number\_list = [random.randint(1,10) for \_ in range(10)]

print(number\_list)

new\_list = []

for i in number\_list:

if i not in new\_list:

new\_list.append(i)

print(new\_list)

uniq = dict(number\_list)

Запустим, и… И у нас ошибка, не получилось. И это логично, ведь для создания словаря нужен не только ключ, но и значение. А мы передаем только ключи. Но выход есть, ведь, как вы понимаете, для программистов такая задача (выделение уникальных элементов) является типовой, но тут нам уже понадобиться другая функция, а именно ***set***. Т.е. такая функция создает из предложенного списка список уникальных элементов, правда такой тип данных уже называется не список, а множество. И программа будет выглядеть следующим образом:

import random

number\_list = [random.randint(1,10) for \_ in range(10)]

print(number\_list)

new\_list = []

for i in number\_list:

if i not in new\_list:

new\_list.append(i)

print(new\_list)

uniq = set(number\_list)

print(uniq)

И теперь такая программа работает, и мы можем сравнить и проверить, два результата. Однако тут можно увидеть два отличия: первое, это скобки (для множества это фигурные скобки {}), и, второе, это то, что элементы идут по возрастанию. На самом деле множества очень похожи на словари, а именно на ключи словарей: элементы множества уникальны и не привязаны к какому-то положению в самом множестве. Обычно множества используют для быстрого поиска какого элемента в этом множестве. И этот типа данных очень похоже на множества из математики, и с множествами связаны некоторые математические операции, которые мы можем изучить в Python’е. Иногда такие операции нам будут нужны.

Давайте посмотрим примеры в консоле:

Создадим два множества

num1 = {1, 2, 3, 4, 5}

nums2 = {4, 5, 6, 7, 8}

Предположим нам надо создать множество, которое состоит из чисел, которые есть в первом множестве и во втором, или как говорят математики, нам нужно найти пересечение множеств. И делается это через методы одного из множеств:

print(nums1.intersection(nums2))

И вот получили результат. А если вы знакомы с алгеброй логики, то можно записать это же действие по-другому:

print(nums1 & nums2)

И это тоже работает.

Объединить:

print(nums1.union(nums2))

print(nums1 | nums2)

Или если мы хотим оставить все элементы в первом множестве, кроме тех, что есть во втором:

print(nums1.difference(nums2))

print(nums1 - nums2)

А вот сложить множества нам не получится.