Урок 12.  
Множества.

короткая линия

# План урока

1. Множества
2. Методы множеств
3. Операции над множествами
4. frozenset
5. Генераторы множеств

# Множества

Четвертая встроенная структура данных в Python – это множество (set). Множество – это неупорядоченная коллекция уникальных элементов. Эта структура данных эквивалентна множествам в математике, и над ними можно совершать знакомые Вам действия: объединение, пересечение, разность и т. д. Как и остальные изученные коллекции, они могут содержать элементы различных типов; можно создать как пустое множество, так и множество с начальными значениями:

numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Мы получили множество чисел от 1 до 10. Возможно, фигурные скобки напомнили Вам словари, однако механизм инициализации у множеств отличается: нет никаких пар “ключ-значение”. Из определения множества следует, что оно не может содержать несколько одинаковых элементов.

|  |
| --- |
| numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  other\_numbers = {1, 1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9, 10, 10}  print(other\_numbers) *# будет выведено множество, аналогичное numbers*  print(numbers == other\_numbers) *# выведет True* |

Однако создать пустое множество с помощью фигурных скобок нельзя, так как они создают пустой словарь. Вместо этого используйте ключевое слово set():

|  |
| --- |
| empty = set() *# создали пустое множество*  print(empty) |

Также Вы можете превратить список или кортеж во множество с помощью функции set():

|  |
| --- |
| set1 = set([**"Hello"**, **"Hello"**, **"World"**, **"World"**])  set2 = set([1] \* 100)  set3 = set(list(input())) *# множество символов введенной строки* |

Возможна и обратная операция перевода множества в список или кортеж c помощью уже знакомых функций list() и tuple().

С помощью множеств удобно решать задачи, в которых нужно получать информацию о принадлежности элемента к какой-либо группе (оператор членства **in** во множествах работает быстрее) или об уникальности обрабатываемых элементов. Например, решим такую простенькую задачу: необходимо найти количество уникальных символов в строке. Мы могли бы решить ее как-то так:

|  |
| --- |
| s = input()  unique = []  **for** char **in** s:  **if** char **not** **in** unique:  unique.append(char)  print(len(unique)) |

Или же без списков, но используя срезы и строковый метод find:

|  |
| --- |
| s = input()  k = 0  **for** i **in** range(len(s)):  **if** s[:i].find(s[i]) == -1:  k = k + 1  print(k) |

Используя множества, решение задачи можно записать намного короче:

|  |
| --- |
| print(len(set(list(input()))))  *# переводим строку в список символов -> переводим список во множество, тем самым удаляя повторяющиеся элементы -> выводим длину множества* |

**Задача “Оптимальная выборка”**

# Методы множеств

Уже созданное множество можно изменять с помощью методов. Для иллюстрации методов будем использовать множество s.

**s.add(element)**

Добавляет элемент во множество s.

**s.remove(element)**

Удаляет элемент из множества. Если такого элемента нет во множестве, программа завершится с ошибкой.

**s.discard(element)**

Удаляет элемент из множества. Если такого элемента нет во множестве, интерпретатор проигнорирует данную команду.

Может возникнуть вопрос, зачем существуют два похожих метода remove и discard, если использовать discard менее рискованно. Когда мы будем проходить **исключения**, то есть научимся обрабатывать ошибки, с которыми сталкивается программа, мы увидим, что уведомление об ошибке может помочь установить обратную связь с пользователем, использующем Вашу программу. Так, встретив ошибку, программа может пояснить, что не так с пользовательским запросом, и предложить поменять его. Это намного продуктивнее, чем игнорирование неверных запросов пользователя.

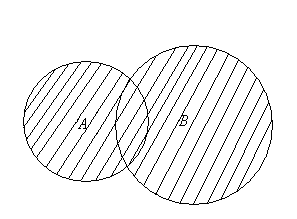
# Операции над множествами

Как уже было сказано, над несколькими множествами можно совершать операции, знакомые Вам из математики. Для иллюстрации будем использовать множества a и b, а также диаграммы Эйлера-Венна. Для каждой операции есть свой оператор (“значок”) и свой метод, они абсолютно равнозначны.

**Объединение множеств**

Полученное множество содержит элементы, содержащиеся хотя бы в одном из множеств.

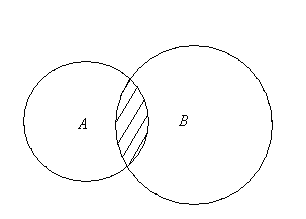
**a | b ИЛИ a.union(b)**



**Пересечение множеств**

Полученное множество содержит элементы, содержащиеся в обоих множествах.

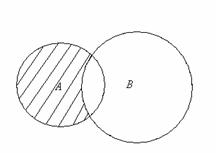
**a & b ИЛИ a.intersection(b)**

****

**Разность множеств**

Полученное множество содержит элементы из множества **a**, но не содержит элементы из множества **b**.

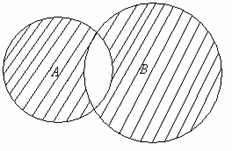
**a - b ИЛИ a.difference(b)**

****

**Симметрическая разность**

Полученное множество содержит элементы, содержащиеся в одном из множеств, но не содержащиеся в обоих.

**a ^ b ИЛИ a.symmetric\_difference(b)**

****

Эти методы возвращают новое множество, поэтому полученное значение нужно сохранять в переменную. Однако, чтобы не писать **a = a.difference(b),** существуют аналогичные методы с добавлением “update”, которые изменяют множество, а не возвращают новое:

**a.update(b) вместо a = a | b**

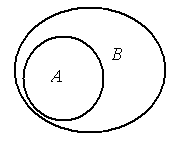
**a.intersection\_update(b) вместо a = a & b**

**a.difference\_update(b) вместо a = a - b**

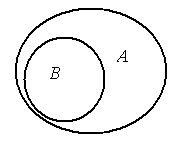
**a.symmetric\_difference\_update(b) вместо a = a ^ b**

Также можно проверить, является ли одно множество подмножеством или надмножеством другого:

**a <= b ИЛИ a.issubset(b)**

****

**a >= b ИЛИ a.issuperset(b)**

****

Эти методы вернут True или False.

**Задача “Круги Эйлера №1”**

**Задача “Круги Эйлера №2”**

**Задача “Круги Эйлера №3”**

**Задача “Корректные замеры”**

**Задача “Измерительная гонка”**

# frozenset

frozenset – это множество, которое нельзя изменить после создания (frozen – “замороженное”).

|  |
| --- |
| set1 = set({1, 2, 3, 4})  set2 = frozenset({3, 1, 4, 2})  print(set1 == set2) *# True, ведь это одно и то же множество*  set1.add(5)  set2.add(5) *# выведет ошибку, так как set2 - это frozenset* |

Между set и frozenset можно провести ту же аналогию, что и между list и tuple. Обе пары реализуют одну и ту же структуру данных, разница лишь в том, что один тип – изменяемый, а другой – неизменяемый. Таким образом, frozenset может являться ключом словаря, а set – нет. Также для frozenset можно выполнять те же операции, что и для обычных множеств.

# Генераторы множеств

Создать множество можно и с помощью генератора. Для этого нужно поместить выражение в фигурные скобки:

|  |
| --- |
| **def** fahrenheit\_to\_celsius(temp):  *# функция перевода температур*  *# ...*  **return** temp  fahrenheit = [ ... ] *# список температур в течение месяца*  celsius = {fahrenheit\_to\_celsius(temp) **for** temp **in** fahrenheit}  print(celsius) *# множество температур* |

**Задача “Мост 2.0”**