Лабораторная работа 4

Множество

Множество в языке Python — это неупорядоченная структура уникальных элементов. Что значит неупорядоченная? Это значит, что два множества эквивалентны, если содержат одинаковые элементы.

Элементы могут быть различных типов. Порядок элементов не определён.

Действия, которые можно выполнять с множеством:

- добавлять и удалять элементы,
- проверять принадлежность элемента множеству,
- перебирать его элементы,
- выполнять операции над множествами (объединение, пересечение, разность).

Операция “проверить принадлежность элемента” выполняется в множестве намного быстрее, чем в списке. 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 (вместо этого используйте неизменяемый кортеж) или другое множество. Требование неизменяемости элементов множества накладывается особенностями представления множества в памяти компьютера.

Действия с элементами множеств:

x in A принадлежит ли элемент x множеству A (возвращают значение типа bool)

x not in A то же, что not x in A

A.add(x) добавить элемент x в множество A

A.discard(x) удалить элемент x из множества A

A.remove(x) удалить элемент x из множества A

A.pop() удаляет из множества один случайный элемент и возвращает его

Операции над множествами:

Операция	**Значение**
A | B A.union(B)	Возвращает множество, являющееся объединением множеств A и B.
A | = B A.update(B)	Записывает в A объединение множеств A и B.

Операция	Значение
A & B A.intersection(B)	Возвращает множество, являющееся пересечением множеств A и B.
A &= B A.intersection_update(B)	Записывает в A пересечение множеств A и 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)	Записывает в A разность множеств A и 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

Словарь

Словари в Python - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

В словаре ключ может быть любого неизменяемого типа! Ключи, как и сами хранимые значения, задаются явно. Значения ключей уникальны, двух одинаковых ключей в словаре быть не может. А вот значения могут быть одинаковыми. Ключом может быть произвольный неизменяемый тип данных: целые и действительные числа, строки, кортежи. Ключом в словаре не может быть множество, но может быть элемент типа frozenset:

специальный тип данных, являющийся аналогом типа set, который нельзя изменять после создания. Значением элемента словаря может быть любой тип данных, в том числе и изменяемый.

Создание словаря

Пустой словарь можно создать при помощи функции dict() или пустой пары фигурных скобок {} (вот почему фигурные скобки нельзя использовать для создания пустого множества).

Для создания словаря с некоторым набором начальных значений можно использовать следующие конструкции:

Capitals = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington'}

Capitals = dict(Russia = 'Moscow', Ukraine = 'Kiev', USA = 'Washington')

Capitals = dict([("Russia", "Moscow"), ("Ukraine", "Kiev"), ("USA", "Washington")])

Capitals = dict(zip(["Russia", "Ukraine", "USA"], ["Moscow", "Kiev", "Washington"]))

Также можно использовать генерацию словаря через Dict comprehensions:

Cities = ["Moscow", "Kiev", "Washington"]

States = ["Russia", "Ukraine", "USA"]

CapitalsOfState = {state: city for city, state in zip(Cities, States)}

Это особенно полезно, когда нужно "вывернуть" словарь наизнанку:

StateByCapital = {CapitalsOfState[state]: state for state in CapitalsOfState}

Методы для словарей:

dict.clear() - очищает словарь.

dict.copy() - возвращает копию словаря.

classmethod dict.fromkeys(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None).

dict.get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).

dict.items() - возвращает пары (ключ, значение).

dict.keys() - возвращает ключи в словаре.

dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).

dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None).

dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

dict.values() - возвращает значения в словаре.

Словари нужно использовать в следующих случаях:

- Подсчет числа каких-то объектов. В этом случае нужно завести словарь, в котором ключами являются объекты, а значениями — их количество.
- Хранение каких-либо данных, связанных с объектом. Ключи — объекты, значения — связанные с ними данные. Например, если нужно по названию месяца определить его порядковый номер, то это можно сделать при помощи словаря Num['January'] = 1; Num['February'] = 2; ...
- Установка соответствия между объектами (например, “родитель—потомок”). Ключ — объект, значение — соответствующий ему объект.
- Если нужен обычный массив, но при этом максимальное значение индекса элемента очень велико, но при этом будут использоваться не все возможные индексы (так называемый “разреженный массив”), то можно использовать ассоциативный массив для экономии памяти.

Задание 1. Ввести с клавиатуры 10 целых чисел. Необходимо записать эти данные в список целых чисел. Затем, перебрать этот список в цикле for и просуммировать все нечетные значения. Результат вывести на экран.

Задание 2. Ввести с клавиатуры названия 5 городов. На их основе сформировать кортеж. Если в этом кортеже нет города "Самара", то следует его добавить в конец кортежа. Результат вывести на экран в виде строки с названиями городов через пробел.

Задание 3. Ввести 5 названий городов с клавиатуры. Необходимо записать входные данные в список. Затем, перебрать его циклом for и заменить значения элементов на длину названия соответствующего города. Результат вывести на экран в виде последовательности чисел через пробел в одну строчку.

Задание 4. Ввести с клавиатуры натуральное число n. С помощью цикла for найти все делители этого числа. Найденные делители выводить сразу в столбик без формирования списка.

Задание 5. Ввести список в виде 5 целых чисел в одну строку через пробел. Необходимо сначала сформировать список на основе введенной строки, а затем, каждое значение этого списка изменить, возведя в квадрат. Отобразить результат на экране в виде строки полученных чисел, записанных через пробел.

Задание 6. Практическое задание по использование словарей.

В файле en-ru.txt находятся строки англо-русского словаря в таком формате:

cat - кошка

dog - собака

Требуется создать русско-английский словарь и вывести его в файл ru-en.txt в таком формате:

делать - to do

дом - home

Порядок строк в выходном файле должен быть словарным (так называемый лексикографический порядок слов). То есть выходные строки нужно отсортировать.

Задание 7. Практическое задание по использованию множеств.

Вывести на экран все элементы множества A, которых нет в множестве B.

A = set('bqlpzlkwehrlulsdhfliuywemrlkjhsdlfjhlzxcovt')

B = set('zmxcvnboaiyerjhbziuxdytvasenbriutsdvinjhgik')

Задание 8. Практическое задание по использованию множеств.

Во входном файле input.txt записан текст. Посчитайте количество уникальных слов в тексте.