Лабораторна робота №2

Словники та множини

Словники

Інший тип даних Python - словники. Словники іноді можна зустріти в інших мовах програмування, як "асоціативні блоки пам'яті" або "асоціативні масиви". На відміну від послідовностей, які індексуються за допомогою діапазону чисел, словники індексуються ключами, які можуть бути будь-яким незмінним типом; рядки і числа можуть завжди бути використані як ключі. Кортежі можуть використовуватися як ключі, якщо вони містять лише стрічки, числа, або кортежі; якщо кортеж містить будь-який змінний об'єкт або безпосередньо чи побічно, то кортеж не може використовуватися як ключ. Ви не можете використовувати списки як ключі, оскільки списки можна змінити в словнику через методи, такі як append() і extend() або можна змінити зрізами чи індексованими призначеннями

Словник по суті є неврегульованим набором ключів: ключ завжди виконує вимогу унікальності (в межах одного словника). Пара фігурних дужок створює порожній словник: {. Розміщення відокремленого комами списку пар `key:value` в межах фігурних дужок додає початкові пари `key:value` до словника; описаний прийом також допустимий при виведенні словника на друк.

Основні операції над словником - збереження значення з деяким ключем і витягання значення, за вказаним ключем. Також можливо видалити пару key:value за допомогою del. Якщо Ви зберігаєте ключ, який знаходиться вже у використанні, про старе значення, що пов'язане з цим ключем, інтерпретатор забуває. При спробі звернеться до такого неіснуючого ключа - виникне помилка.

Метод keys() об'єкту словник повертає список всіх ключів, використовуваних в словнику, в довільному порядку (якщо Ви хочете сортувати ключі, слід застосувати метод sort() для списку ключів). Перевірка, чи знаходиться єдиний ключ в словнику, слід використовувати метод has_key() словника.

Тут приведений маленький приклад, використання словника:

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> type(tel)
<type 'dict'>
>>> help(dict)
Help on class dict in module __builtin__:

class dict(object)

>>> tel['guido']= 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
```

```
>>> tel['irv']= 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> tel.has_key('guido')
True
>>> 'guido' in tel
True
```

dict() конструктор формує словник безпосередньо із списків пар "ключ:значення", збережених як кортежі. Коли пари формують шаблон, останній вноситься до списку для використання, шаблони можна стисло визначити із списками ключових значень.

```
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'jack': 4098, 'guido': 4127}
>>> dict([(x, x**2) for x in (2, 4, 6)])     # безпосереднє використання
списку
{2: 4, 4: 16, 6: 36}
```

Коли ключі - прості стрічки, іноді простіше визначати такі пари, що використовують параметри з ключами:

```
>>> dict(sape=4139, guido=4127, jack=4098)
{'sape': 4139, 'jack': 4098, 'guido': 4127}
```

Множини

Множина (set) – це колекція елементів, яка має наступні властивості:

- Елементи в множині не впорядковані
- Елементи в множині унікальні. Тобто один і той самий елемент не може повторятися в колекції.
- Елементи в множині незмінні. Можна додавати та видаляти елементи з колекції, але змінювати самі елементи не можна.

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket)
>>> fruit set(['orange', 'pear', 'apple', 'banana'])
>>> 'orange' in fruit
True
>>> 'crabgrass' in fruit
False
```

З множинами можна виконувати наступні операції

```
>>> A = {1, 2, 3, 4}
```

```
>>> B = {3, 4, 5, 6}
>>> A & B # Перетин
{3, 4}
>>> A | B # Об'єднання
{1, 2, 3, 4, 5, 6}
>>> A - B # Різниця
{1, 2}
>>> A ^ B # Симетрична різниця
{1, 2, 3, 4, 5, 6}
```

Використовуючи set() можна прибрати дублікати зі списку

```
>>> L = [1, 2, 1, 3, 2, 4, 5]
>>> set(L)
{1, 2, 3, 4, 5}
>>> L = list(set(L)) # Remove duplicates
>>> L
[1, 2, 3, 4, 5]
```

Ще декілька прикладів

```
>>> engineers = {'bob', 'sue', 'ann', 'vic'}
>>> managers = {'tom', 'sue'}

>>> 'bob' in engineers # Is bob an engineer?
True

>>> engineers & managers # Who is both engineer and manager?
{'sue'}

>>> engineers | managers # All people in either category
{'vic', 'sue', 'tom', 'bob', 'ann'}

>>> engineers – managers # Engineers who are not managers
{'vic', 'bob', 'ann'}

>>> managers – engineers # Managers who are not engineers
{'tom'}

>>> engineers > managers # Are all managers engineers? (superset)
False

>>> {'bob', 'sue'} < engineers # Are both engineers? (subset)
True

>>> (managers | engineers) > managers # All people is a superset of
managers
True
```

```
>>> managers ^ engineers # Who is in one but not both?
{'vic', 'bob', 'ann', 'tom'}

>>> (managers | engineers) - (managers ^ engineers) # Intersection!
{'sue'}
```

Задания по вариантам

Используя словари и множества запрограммируйте следующие задания

1. Дан список чисел. Определите, сколько раз в нем каждое из чисел встречается.
2. Определите, сколько раз встречается каждое из слов в неком тексте.
3. Определите, сколько раз встречается каждая из букв в неком тексте.
4. Дан текст. Выведите слово, которое в этом тексте встречается чаще всего.
5. Дан текст. Выведите все слова, встречающиеся в тексте, по одному на каждую строку. Слова должны быть отсортированы по убыванию их количества появления в тексте.
6. Дан текст. Выведите слова, которые встречаются в тексте только по одному разу.
7. Дан список чисел. Выведите числа, которые встречаются в списке только по одному разу.
8. Дан текст. Выведите слова, которые встречаются в тексте по несколько раз.
9. Дан список чисел. Выведите числа, которые встречаются в списке по несколько раз.
10. Дан текст. Посчитаете сколько в нем слов состоящих из 1, 2, …, N букв
11. Дан список состоящий из кортежей. Где в кортеже указано имя и город, вида [('Andrii', 'Kyiv'), ('Nik', 'London'), ('Luba', 'Kyiv')]. Выведите информацию сколько людей живет в каждом городе
12. Дан список состоящий из кортежей. Где в кортеже указано имя и город, вида [('Andrii', 'Kyiv'), ('Nik', 'London'), ('Luba', 'Kyiv')]. Получите словарь в котором ключами были бы города, а в качестве значений – списки состоящие из имен людей, которые живут в данном городе.

Література:

Программирование на Python: Часть 4. Словари https://www.ibm.com/developerworks/ru/library/l-python_part_4/