Python Starter

Множини та відображення

План заняття

1. Що таке множини?
2. Створення множин
3. Зміна множин
4. Видалення елементів з множини
5. Операції з множинами Python
6. Що таке відображення?
7. Сортування словників

Після уроку обов'язково

Повторіть цей урок у відео форматі на ITVDN.com

Доступ можна отримати через керівництво вашого навчального центру

Перевірте, як Ви засвоїли цей матеріал на TestProvider.com

Тема

Множини та відображення

Об'єкти, що хешуються

- Об'єкт називається **хешованим**, якщо він має хеш-значення (ціле число), яке ніколи не змінюється протягом його життєвого циклу і повертається методом *hash()* і може порівнюватися з іншими об'єктами (реалізує метод *eq()*). Рівні об'єкти, що хешуються, повинні мати рівні хеш-значення.
- Об'єкти, що хешуються, можуть бути використані як ключі словників і члени множин.
- Всі стандартні незмінювані об'єкти є хешованими. Всі стандартні змінювані об'єкти є не хешованими.

Множини

- **Множина** – це невпорядкована колекція хешованих об'єктів, які не повторюються.
- Зазвичай використовуються для перевірки елементу на входження в множину та видалення повторень елементів і виконання таких операцій, як об'єднання, перетин, різниця та симетрична різниця.
- У множинах немає поняття позиції елементу. Відповідно, вони не підтримують індексацію та зрізи.
- Вбудовані класи множин: set (змінювана множина), frozenset (незмінювана множина).

Створення множин

Створення множини:

- використання конструктора типу;
- перерахування елементів у фігурних дужках (тільки set);
- включення множин (аналогічно до спискових включень, тільки set).

```
empty_set = set()
empty_set = frozenset()

my_set = {1, 3, 2, 5}
my_set = frozenset([1, 3, 2, 5])

my_set = {x ** 3 for x in range(5)}
```

Операції з множинами

Операція	Опис
set([iterable]) frozenset([iterable])	створення множини (порожньої або з елементів ітерабельного об'єкта)
len(s)	кількість елементів множини
x in s x not in s	перевірка знаходження елемента у множині
s.isdisjoint(t)	перевірка того, що дана множина не має спільних елементів із заданою
s.issubset(t) s <= t	перевірка того, що всі елементи множини s є елементами множини t
s < t	перевірка того, що s <= t і s != t
s.issuperset(t) s >= t	перевірка того, що всі елементи множини t є елементами множини s
s > t	перевірка того, що s >= t і s != t

Операції з множинами

Операція	Опис
s.union(t, …) s | t | …	створення нової множини, яка є об'єднанням даних множин
s.intersection(t, …) s & t & …	створення нової множини, яка є перетином даних множин
s.difference(t, …) s - t - …	створення нової множини, яка є різницею даних множин
s.symmetric_difference(t) s ^ t	створення нової множини, яка є симетричною різницею даних множин (тобто, різниця об'єднання та перетину множин)
s.copy()	неповна копія множини s

Операції над множинами, які є методами, приймають у якості аргументів будь-які ітерабельні об'єкти. Операції над множинами, записані у вигляді бінарних операцій, вимагають, щоб другий операнд операції теж був множиною, і повертають множину того типу, яким була перша множина.

Операції зі змінними множинами

Операція	Опис
s.update(t, …) s |= t | …	додати до цієї множини елементи з інших множин
s.intersection_update(t, …) s &= t & …	залишити в даній множині тільки ті елементи, які є і в інших множинах
s.difference_update(t, …) s -= t | …	видалити з цієї множини ті елементи, які є в інших множинах
s.symmetric_difference_update(t) s ^= t	залишити або додати до s елементи, які є або в s, або в t, але не в обох множинах
s.add(element)	додати новий елемент до множини
s.remove(element)	видалити елемент із множини; якщо такого елементу немає, виникає KeyError
s.discard(element)	видалити елемент із множини, якщо він у ній знаходиться
s.pop()	видалити з множини і повернути довільний елемент (KeyError, якщо порожнє)
s.clear()	видалити всі елементи множини

Відображення

Відображення (mapping) – це об'єкт-контейнер, який підтримує довільний доступ до елементів за ключами і реалізує наступні методи, описані в абстрактному базовому класі collections.Mapping:

- get(key, default=None)
- items()
- keys()
- values()

Відображення, що змінюються, також повинні підтримувати наступні методи, описані в абстрактному базовому класі collections.MutableMapping:

- clear()
- pop(key)
- popitem()
- setdefault(key, default=None)
- update()

До відображень відносяться класи dict, collections.defaultdict, collections.OrderedDict і collections.Counter.

Словники (асоціативні масиви)

- Вбудованим класом відображення є dict, що реалізує таку структуру даних, як **словник**, чи **асоціативний масив**, тобто, невпорядковану змінювану колекцію пар (ключ, значення), яка підтримує довільний доступ до її елементів за їх ключами.
- Ключі словників повинні бути значеннями, що хешуються.

Числові ключі у словниках підпорядковуються правилам порівняння чисел. Таким чином, int(1) та float(1.0) вважаються однаковим ключем. Однак через те, що значення типу float зберігаються приблизно, не рекомендується використовувати їх як ключі

Довільна кількість іменованих параметрів функції

- Подібно до того, як можна передавати до функцій довільну кількість позиційних аргументів, які зберігаються в кортежі, можна передавати довільну кількість іменованих аргументів, які зберігаються у словнику.
- Для цього перед ім'ям даного словника у списку формальних параметрів ставляться два символи **.
- Якщо використовуються обидва способи передачі довільної кількості аргументів, параметр у формі **kwargs у сигнатурі функції має йти після параметру в формі *args.
- Аналогічно можна розпаковувати будь-які відображення в іменовані параметри при виклику функції.

```
def function(*args, **kwargs):
    # type(args) ==tuple
    #type(kwargs) ==dict
    pass
```

Створення словників

- Перелік пар ключ-значення, розділених символом двокрапки, через коми у фігурних дужках:

```
{'John': 18, 'Mike': 30}
```

- Включення словників (аналогічно до спискових включень):

```
{key: value for key in keys for value in values}
```

- Використання конструктора класу dict:

```
dict(**kwargs)
dict(mapping, **kwargs)
dict(iterable, **kwargs)
```

Операції зі словниками та іншими відображеннями

Операція	Опис
len(d)	Кількість елементів.
d[key]	Отримання значення з ключем key. Якщо такий ключ не існує і відображення реалізує спеціальний метод __missing__(self, key), то він викликається. Якщо ключ не існує і метод __missing__ не визначено, викидається виняток KeyError.
d[key] = value	Змінити значення або створити нову пару ключ-значення, якщо ключ не існує.
key in d key not in d	Перевірка наявності ключа у відображенні.
iter(d)	Те ж саме, що й iter(d.keys()).
clear()	Видалити всі елементи словника.
copy()	Створити неповну копію словника.
@classmethod dict.fromkeys(sequence[, value])	Створює новий словник із ключами з послідовності sequence та заданим значенням (за замовчуванням – None).

Операції зі словниками та іншими відображеннями

Операція	Опис
d.get(key[, default])	Безпечне отримання значення за ключем (ніколи не викидає KeyError). Якщо ключ не знайдено, повертається значення default (за замовчуванням – None).
d.items()	В Python 3 повертає об'єкт представлення словника, відповідний парам виду (ключ, значення). В Python 2 повертає відповідний список, а метод iteritems() повертає ітератор. Аналогічний метод у Python 2.7 – viewitems().
d.keys()	В Python 3 повертає об'єкт представлення словника, що відповідає ключам словника. В Python 2 повертає відповідний список, а метод iterkeys() повертає ітератор. Аналогічний метод у Python 2.7 – viewkeys().
d.pop(key[, default])	Якщо ключ key існує, видаляє елемент зі словника та повертає його значення. Якщо ключ не існує і задано значення default, повертається це значення, інакше викидається виняток KeyError.
d.popitem()	видаляє довільну пару ключ-значення та повертає її. Якщо словник порожній, виникає виняток KeyError.

Операції зі словниками та іншими відображеннями

Операція	Опис
d.setdefault(key[, default])	Якщо ключ key існує, повертає відповідне значення. Інакше створює елемент з ключем key та значенням default. default за замовчуванням дорівнює None.
d.update(mapping)	Приймає або інший словник або відображення, або ітерабельний об'єкт, що складається з ітерабельних об'єктів – пар ключ-значення, або іменовані аргументи. Додає відповідні елементи до словника, перезаписуючи елементи із існуючими ключами.
d.values()	В Python 3 повертає об'єкт представлення словника, відповідний значенням. В Python 2 повертає відповідний список, а метод itervalues() повертає ітератор. Аналогічний метод у Python 2.7 – viewvalues().

Об'єкти представлення словника

Об'єкти, що повертаються методами *items(), keys() та values() (viewitems(), viewkeys(), viewvalues()* у Python 2.7) – це об'єкти **представлення словника**. Вони надають динамічне представлення елементів словника, тобто зміни даного словника автоматично відображаються і на цих об'єктах.

Операції з представленнями словників:

- iter(dictview) – отримання ітератора за ключами, значеннями або парами ключів та значень. Усі представлення словників при ітеруванні повертають елементи словника в однаковому порядку. При спробі змінити словник під час ітерування може виникнути виняток RuntimeError.
- len(dictview) – кількість елементів у словнику.
- x in dictview – перевірка існування ключа, значення або пари ключ-значення в словнику.

Сортування словників

Сортування словника можливе за допомогою:
- циклу for;
- функції sorted();
- функції itemgetter() в модулі operator;
- лямбда-функції (в курсі Advanced познайомимося з цією можливістю).

У версіях Python до 3.7 необхідно використовувати з модуля collections клас OrderedDict, щоб зберегти відсортований словник після сортування словника за значеннями. Дані об'єкти виглядають як словники, які зберігають порядок вставки.

Дивіться наші уроки у відео форматі

ITVDN.com

Перегляньте цей урок у відео форматі на освітньому порталі ITVDN.com
для закріплення пройденого матеріалу.

Усі курси записані сертифікованими тренерами, які працюють у навчальному центрі CyberBionic Systematics

ITVDN
IT VIDEO DEVELOPERS NETWORK

Перевірка знань

TestProvider.com

TestProvider – це online сервіс перевірки знань з інформаційних технологій. За його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT спеціаліста.

Після кожного уроку проходьте тестування для перевірки знань на TestProvider.com

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.

TestProvider

Q&A

Дякую за увагу!

Інформаційний відеосервіс для розробників програмного забезпечення

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association