Microsoft Partner
Silver Learning

Python Essential

Множества и отображения

ITVDN
IT VIDEO DEVELOPERS NETWORK

Информационный видеосервис для разработчиков программного обеспечения

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Множества и отображения

Хешируемые объекты

- Объект называется **хешируемым**, если он имеет *хеш-значение* (целое число), которое никогда не изменяется на протяжении его жизненного цикла и возвращается методом *__hash__()*, и может сравниваться с другими объектами (реализует метод *__eq__()*). Равные хешируемые объекты должны иметь равные хеш-значения.
- Хешируемые объекты могут быть использованы как ключи словарей и члены множеств.
- Все стандартные неизменяемые объекты хешируемые. Все стандартные изменяемые объекты не хешируемые.

Множества

- **Множество** – это неупорядоченная коллекция хешируемых объектов, которые не повторяются.
- Обычно используются для проверки элемента на вхождение в множество и удаление повторений элементов и выполнения таких операций, как объединение, пересечение, разница и симметрическая разница.
- В множествах нет понятия позиции элемента. Соответственно, они не поддерживают индексацию и срезы.
- Встроенные классы множеств: set (изменяемое множество), frozenset (неизменяемое множество).

Создание множеств

Создание множества:

- использование конструктора типа;
- перечисление элементов в фигурных скобках (только set);
- включение множеств (аналогично списковым включениям, только 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)}
```

Операции с множествами

Операция	Описание
set([iterable]) frozenset([iterable])	создание множества (пустого или из элементов итерабельного объекта)
len(s)	количество элементов множества
x in s x not in s	проверка нахождения элемента в множестве
s.isdisjoint(t)	проверка того, что данное множество не имеет общих элементов с заданным
s.issubset(t) s <= t	проверка того, что все элементы множества s являются элементами множества t
s < t	проверка того, что s <= t и s != t
s.isuperset(t) s >= t	проверка того, что все элементы множества t являются элементами множества s
s > t	проверка того, что s >= t и s != t

Операции с множествами

Операция	Описание
s.union(t, …) s | t | …	создание нового множества, которое является объединением данных множеств
s.intersection(t, …) s & t & …	создание нового множества, которое является пересечением данных множеств
s.difference(t, …) s - t - …	создание нового множества, которое является разницей данных множеств
s.symmetric_difference(t) s ^ t	создание нового множества, которое является симметрической разницей данных множеств (то есть, разница объединения и пересечения множеств)
s.copy()	неполная копия множества s

Операции над множествами, которые являются методами, принимают в качестве аргументов любые итерабельные объекты. Операции над множествами, записанные в виде бинарных операций, требуют, чтобы второй операнд операции тоже был множеством, и возвращают множество того типа, которым было первое множество.

Множества и отображения

Операции с изменяемыми множествами

Операция	Описание
s.update(t, …) s |= t | …	добавить в данное множество элементы из других множеств
s.intersection_update(t, …) s &= t & …	оставить в данном множестве только те элементы, которые есть и в других множествах
s.difference_update(t, …) s -= t | …	удалить из данного множества те элементы, которые есть в других множествах
s.symmetric_difference_update(t) s ^= t	оставить или добавить в s элементы, которые есть либо в s, либо в t, но не в обоих множествах
s.add(element)	добавить новый элемент в множество
s.remove(element)	удалить элемент из множества; если такого элемента нет, возникает KeyError
s.discard(element)	удалить элемент из множества, если он в нём находится
s.pop()	удалить из множества и вернуть произвольный элемент (KeyError, если пустое)
s.clear()	удалить все элементы множества

Отображения

Отображение (mapping) – это объект-контейнер, который поддерживает произвольный доступ к элементам по ключам и реализует следующие методы, описанные в абстрактном базовом классе collections.Mapping:

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

Изменяемые отображения также должны поддерживать следующие методы, описанные в абстрактном базовом классе collections.MutableMapping:

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

К отображениям относятся классы dict, collections.defaultdict, collections.OrderedDict и collections.Counter.

Словари (ассоциативные массивы)

- Встроенным классом отображения является dict, который реализует такую структуру данных, как **словарь**, или **ассоциативный массив**, то есть неупорядоченную изменяемую коллекцию пар (ключ, значение), которая поддерживает произвольный доступ к её элементам по их ключам.
- Ключи словарей должны быть хешируемыми значениями.

Числовые ключи в словарях подчиняются правилам сравнения чисел. Таким образом, int(1) и float(1.0) считаются одинаковым ключом. Однако из-за того, что значения типа float сохраняются приближенно, не рекомендуется использовать их в качестве ключей.

Произвольное количество именованных параметров функции

- Подобно тому, как можно передавать в функции произвольное количество позиционных аргументов, которые сохраняются в кортеже, можно передавать произвольное количество именованных аргументов, которые сохраняются в словаре.
- Для этого перед именем данного словаря в списке формальных параметров ставится два символа **.
- Если используются оба способа передачи произвольного количества аргументов, параметр в форме **kwargs в сигнатуре функции должен идти после параметра в форме *args.
- Аналогично можно и распаковывать любые отображения в именованные параметры при вызове функции.

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

Создание словарей

- Перечисление пар ключ-значение, разделённых символом двоеточия, через запятые в фигурных скобках:

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

- Включения словарей (аналогично списковым включениям):

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

- Использование конструктора класса dict:

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

Множества и отображения

Операции со словарями и другими отображениями

Операция	**Описание**
len(d)	Количество элементов.
d[key]	Получение значения с ключом key. Если такой ключ не существует и отображение реализует специальный метод __missing__(self, key), то он вызывается. Если ключ не существует и метод __missing__ не определён, выбрасывается исключение KeyError.
d[key] = value	Изменить значение или создать новую пару ключ-значение, если ключ не существует.
key in d key not in d	Проверка наличия ключа в отображении.
iter(d)	То же самое, что iter(d.keys()).
clear()	Удалить все элементы словаря.
copy()	Создать неполную копию словаря.
@classmethod dict.fromkeys(sequence[, value])	Создаёт новый словарь с ключами из последовательности sequence и заданным значением (по умолчанию – None).

Операции со словарями и другими отображениями

Операция	Описание
d.get(key[, default])	Безопасное получение значения по ключу (никогда не выбрасывает KeyError). Если ключ не найден, возвращается значение default (по-умолчанию – None).
d.items()	В Python 3 возвращает объект представления словаря, соответствующий парам вида (ключ, значение). В Python 2 возвращает соответствующий список, а метод iteritems() возвращает итератор. Аналогичный метод в Python 2.7 – viewitems().
d.keys()	В Python 3 возвращает объект представления словаря, соответствующий ключам словаря. В Python 2 возвращает соответствующий список, а метод iterkeys() возвращает итератор. Аналогичный метод в Python 2.7 – viewkeys().
d.pop(key[, default])	Если ключ key существует, удаляет элемент из словаря и возвращает его значение. Если ключ не существует и задано значение default, возвращается данное значение, иначе выбрасывается исключение KeyError.
d.popitem()	удаляет произвольную пару ключ-значение и возвращает её. Если словарь пустой, возникает исключение KeyError.

Операции со словарями и другими отображениями

Операция	Описание
d.setdefault(key[, default])	Если ключ key существует, возвращает соответствующее значение. Иначе создаёт элемент с ключом key и значением default. default по умолчанию равен None.
d.update(mapping)	Принимает либо другой словарь или отображение, либо итерабельный объект, состоящий из итерабельных объектов – пар ключ-значение, либо именованные аргументы. Добавляет соответствующие элементы в словарь, перезаписывая элементы с существующими ключами.
d.values()	В Python 3 возвращает объект представления словаря, соответствующий значениям. В Python 2 возвращает соответствующий список, а метод itervalues() возвращает итератор. Аналогичный метод в Python 2.7 – viewvalues().

Объекты представления словаря

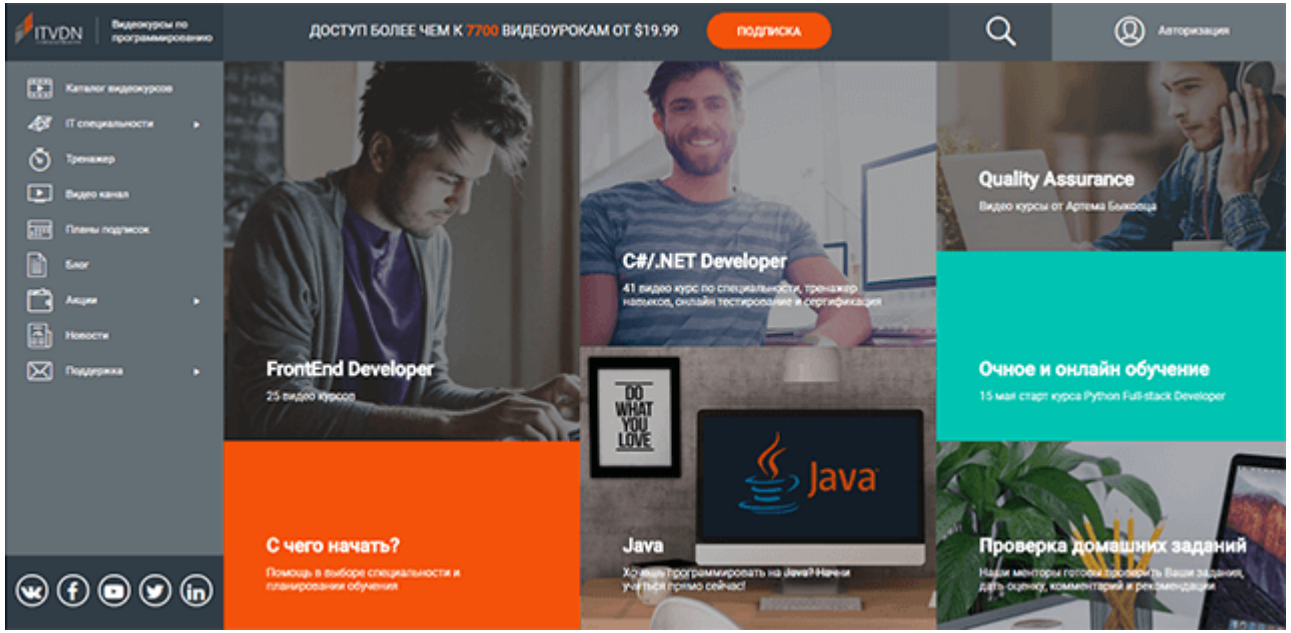
Объекты, возвращаемые методами *items()*, *keys()* и *values()* (*viewitems()*, *viewkeys()*, *viewvalues()* в Python 2.7) – это объекты **представления словаря**. Они предоставляют динамическое представление элементов словаря, то есть изменения данного словаря автоматически отображаются и на этих объектах.

Операции с представлениями словарей:

- iter(dictview) – получение итератора по ключам, значениям или парам ключей и значений. Все представления словарей при итерировании возвращают элементы словаря в одинаковом порядке. При попытке изменить словарь во время итерирования может возникнуть исключение RuntimeError.
- len(dictview) – количество элементов в словаре.
- x in dictview – проверка существования ключа, значения или пары ключ-значение в словаре.

Смотрите наши уроки в видео формате

ITVDN.com



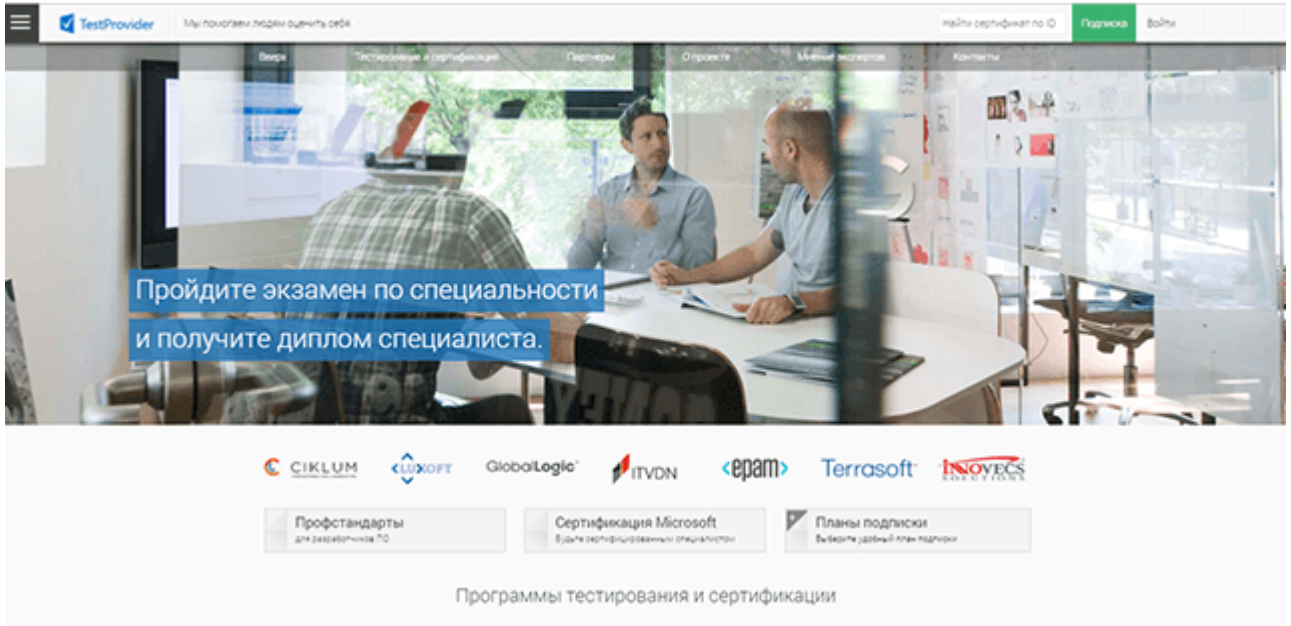
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics и другими высококвалифицированными разработчиками.

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

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