

Python Essential

Множества и отображения

Python Essential

После урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://itvdn.com)

Доступ можно получить через руководство вашего учебного центра



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://testprovider.com)

Множества и отображения

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 и отображения

Множества



- **Множество** – это неупорядоченная коллекция хешируемых объектов, которые не повторяются.
- Обычно используются для проверки элемента на входжение 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).

Множества и отображения

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)}
```

Множества и отображения

Операции с множествами

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

Множества и отображения

Операции с множествами

| Операция | Описание |
|--|---|
| <code>s.union(t, ...)</code> $s \mid t \mid \dots$ | создание нового множества, которое является объединением данных множеств |
| <code>s.intersection(t, ...)</code> $s \& t \& \dots$ | создание нового множества, которое является пересечением данных множеств |
| <code>s.difference(t, ...)</code> $s - t - \dots$ | создание нового множества, которое является разницей данных множеств |
| <code>s.symmetric_difference(t)</code> $s \wedge t$ | создание нового множества, которое является симметрической разницей данных множеств (то есть, разница объединения и пересечения множеств) |
| <code>s.copy()</code> | неполная копия множества s |



Операции над множествами, которые являются методами, принимают в качестве аргументов любые итерируемые 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 |
|---|--|
| <code>s.update(t, ...)</code> <code>s = t ...</code> | добавить в данное множество элементы из других множеств |
| <code>s.intersection_update(t, ...)</code> <code>s &= t & ...</code> | оставить в данном множестве только те элементы, которые есть и в других множествах |
| <code>s.difference_update(t, ...)</code> <code>s -= t ...</code> | удалить из данного множества те элементы, которые есть в других множествах |
| <code>s.symmetric_difference_update(t)</code> <code>s ^= t</code> | оставить или добавить в <code>s</code> элементы, которые есть либо в <code>s</code> , либо в <code>t</code> , но не в обоих множествах |
| <code>s.add(element)</code> | добавить новый элемент в множество |
| <code>s.remove(element)</code> | удалить элемент из множества; если такого элемента нет, возникает <code>KeyError</code> |
| <code>s.discard(element)</code> | удалить элемент из множества, если он в нём находится |
| <code>s.pop()</code> | удалить из множества и вернуть произвольный элемент (<code>KeyError</code> , если пустое) |
| <code>s.clear()</code> | удалить все элементы множества |

Множества и отображения

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`.

Множества и отображения

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.

Множества и отображения

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
```

Множества и отображения

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

Множества и отображения

Операции со словарями и другими отображениями

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

Множества и отображения

Операции со словарями и другими отображениями

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

Множества и отображения

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



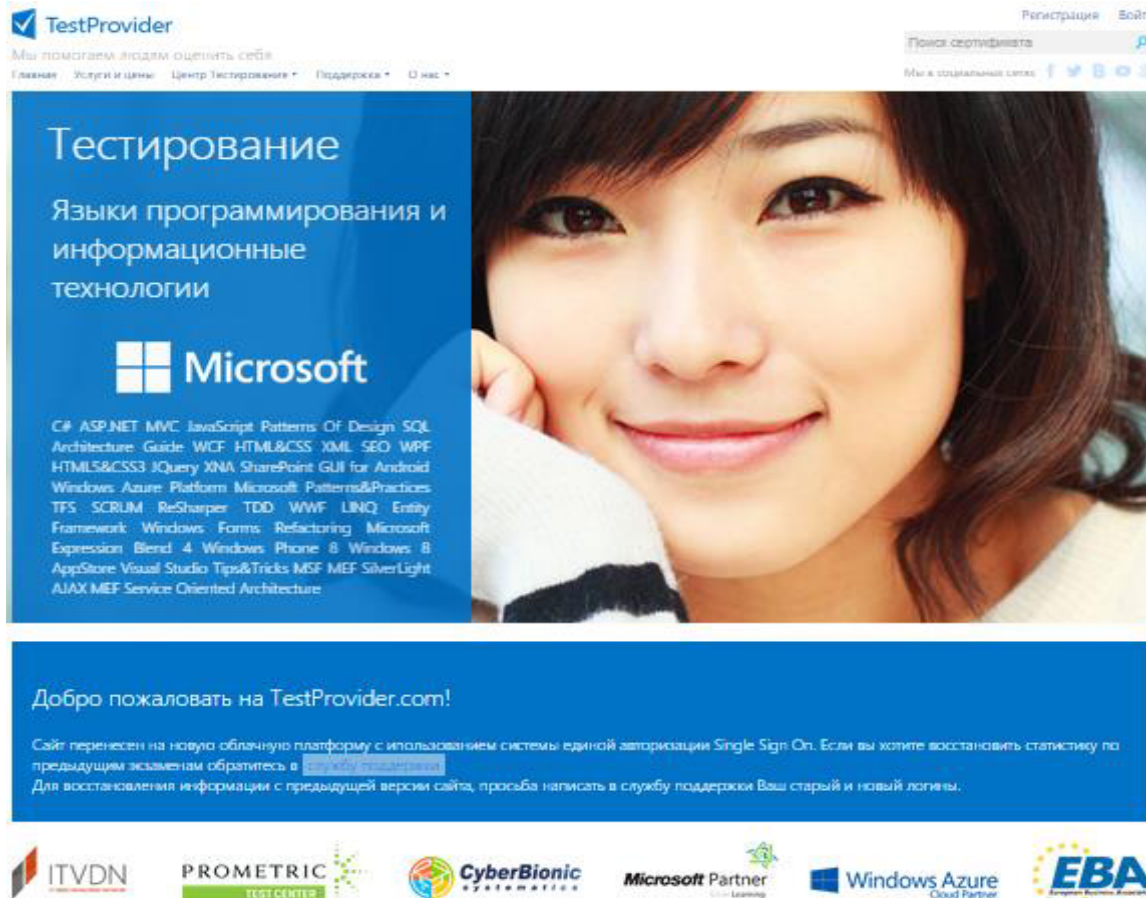
Посмотрите этот урок в видео формате на образовательном портале [ITVDN.com](http://itvdn.com) для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics



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 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.



Python Essential

Q&A

Информационный видеосервис для разработчиков программного обеспечения

