Коллекции Python

Последовательности

Последовательности

- Последовательности – это один из типов данных, поддерживающих оператор проверки на вхождение (in), функцию определения размера (len()), оператор извлечения срезов ([]) и возможность выполнения итераций. В языке Python имеется пять встроенных типов последовательностей: bytearray, bytes, list, str и tuple.

Кортежи

- Кортеж (tuple) – это упорядоченная последовательность из нуля или более ссылок на объекты.
- Кортежи поддерживают тот же синтаксис получения срезов, что и строки.
- Пустой кортеж создается с помощью пары пустых круглых скобок (), а кортеж, состоящий из одного или более элементов, может быть создан с помощью запятых.

```
>>> 1,2,3,99
(1, 2, 3, 99)
```

Методы кортежей

- t.count(x) - возвращает количество объектов x в кортеже t;
- t.index(x) - возвращает индекс самого первого (слева) вхождения объекта x в кортеж t или возбуждает исключение ValueError, если объект x отсутствует в кортеже.
- Кроме того, кортежи могут использоваться с оператором + (конкатенации), * (дублирования) и [] (получения среза), а операторы in и not in могут применяться для проверки на вхождение.
- Кортежи могут сравниваться с помощью стандартных операторов сравнения (<, <=, ==, !=, >=, >), при этом сравнивание производится поэлементно (и рекурсивно, при наличии вложенных элементов, таких как кортежи в кортежах).

Списки

Список

- Список (list) – это упорядоченная последовательность из нуля или более ссылок на объекты. Списки поддерживают тот же синтаксис получения срезов, что и строки с кортежами.

Методы списков

Синтаксис	Описание
L.append (x)	Добавляет элемент x в конец списка L
L.count(x)	Возвращает число вхождений элемента x в список L
L.extend(m) L += m	Добавляет в конец списка L все элементы итерируемого объекта m; оператор += делает то же самое
L.index(x, start, end)	Возвращает индекс самого первого (слева) вхождения элемента x в список L (или в срез start:end списка L), в противном случае возбуждает исключение ValueError
L.insert(i, x)	Вставляет элемент x в список L в позицию int i

Методы списков

Синтаксис	Описание
L.pop()	Удаляет самый последний элемент из списка L и возвращает его в качестве результата
L.pop(i)	Удаляет из списка L элемент с индексом int i и возвращает его в качестве результата
L.remove(x)	Удаляет самый первый (слева) найденный элемент x из списка L или возбуждает исключение ValueError, если элемент x не будет найден
L.reverse()	Переставляет в памяти элементы списка в обратном порядке
L.sort(...)	Сортирует список в памяти. Этот метод принимает те же необязательные аргументы key и reverse, что и встроенная функция sorted()

Генераторы списков

- Генератор списков – это выражение и цикл с дополнительным условием, заключенное в квадратные скобки, в котором цикл используется для создания элементов списка, а условие используется для исключения нежелательных элементов.
- В простейшем виде генератор списков записывается, как показано ниже:

```
[item for item in iterable]
```

Генераторы списков

- Генераторы могут использоваться как выражения и они допускают включение условной инструкции, вследствие чего мы получаем две типичные синтаксические конструкции использования генераторов списков:

```
[expression for item in iterable]
```

```
[expression for item in iterable if condition]
```

Примеры

```
>>> a = [i*10 for i in range(5)]
>>> a
[0, 10, 20, 30, 40]

>>> a = [x**2 for x in range(50) if x % 10 == 5]
>>> a
[25, 225, 625, 1225, 2025]
```

Множества

Множества

- Тип set – это неупорядоченная коллекция из нуля или более ссылок на объекты, указывающих на хешируемые объекты.
- Множества относятся к категории изменяемых типов, поэтому легко можно добавлять и удалять их элементы, но, так как они являются неупорядоченными коллекциями, к ним не применимо понятие индекса и не применима операция извлечения среза.
- Хешируемые объекты – это объекты, имеющие специальный метод __hash__(), на протяжении всего жизненного цикла объекта всегда возвращающий одно и то же значение, которые могут участвовать в операциях сравнения на равенство посредством специального метода __eq__().
- Все встроенные неизменяемые типы данных, такие как float, frozenset, int, str и tuple, являются хешируемыми объектами и могут добавляться во множества.

Методы множеств

Синтаксис	Описание
s.add(x)	Добавляет элементы x во множество s, если они отсутствуют в s
s.clear()	Удаляет все элементы из множества s
s.difference(t) s - t	Возвращает новое множество, включающее элементы множества s, которые отсутствуют в множестве t
s.discard(x)	Удаляет элемент x из множества s, если он присутствует в множестве s; смотрите также метод set.remove()
s.intersection(t) s & t	Возвращает новое множество, включающее элементы, присутствующие одновременно в множествах s и t
s.isdisjoint(t)	Возвращает True, если множества s и t не имеют общих элементов

Методы множеств

Синтаксис	Описание
s.issubset(t) s <= t	Возвращает True, если множество s эквивалентно множеству t или является его подмножеством; чтобы проверить, является ли множество s только подмножеством множества t, следует использовать проверку s < t
s.issuperset(t) s >= t	Возвращает True, если множество s эквивалентно множеству t или является его надмножеством; чтобы проверить, является ли множество s только надмножеством множества t, следует использовать проверку s > t
s.pop()	Возвращает и удаляет случайный элемент множества s или возбуждает исключение KeyError, если s – это пустое множество

Методы множеств

Синтаксис	Описание
s.remove(x)	Удаляет элемент x из множества s или возбуждает исключение KeyError, если элемент x отсутствует в множестве s; смотрите также метод set.discard()
s.union(t) s | t	Возвращает новое множество, включающее все элементы множества s и все элементы множества t, отсутствующие в множестве s
s.update(t) s |= t	Добавляет во множество s все элементы множества t, отсутствующие в множестве s

Генераторы множеств

- Генератор множества – это выражение и цикл с необязательным условием, заключенные в фигурные скобки.
- Подобно генераторам списков, генераторы множеств поддерживают две формы записи:

```
{expression for item in iterable}

{expression for item in iterable if condition}
```

Пример

```
>>> m = {x for x in range(20) if x % 2 == 0}
>>> m
{0, 2, 4, 6, 8, 10, 12, 14, 16, 18}
```