InnoDom

Коллекции

Коллекции

это программные объекты (*переменная-контейнер*), хранящие набор значений одного или различных **типов** и позволяющие обращаться к этим значениям, а также применять специальные функции и методы, зависящие от типа коллекции.

Все типы или коллекции, в которых имеется множество других составляющих, либо индексированные типы, вне зависимости от того изменяемые они или нет, называют **итерируемыми объектами**.

Коллекции

- Список - `list()`
- Кортеж - `tuple()`
- Множество - `set()`
- Неизменяемое множество - `frozenset()`
- Словарь - `dict()`

Виды коллекций

- **Изменяемые** – коллекции, которые **можно** изменить в любой момент времени.
- **Неизменяемые** – коллекции, которые изменить **нельзя**, но можно *пересоздать*.

Виды коллекций

- **Изменяемые** – коллекции, которые **можно** изменить в любой момент времени.
- **Неизменяемые** – коллекции, которые изменить **нельзя**, но можно *пересоздать*.

! **Строки** тоже в какой-то степени можно считать коллекцией, т.к. они хранят последовательность символов. Но строки - это базовый тип, к тому же в коллекциях также можно хранить строки, поэтому принято их к коллекциям **не относить**.

Виды коллекций

Название коллекции или типа	Изменяемость
`bool`	Нет
`int`	Нет
`float`	Нет
`list`	Да
`tuple`	Нет
`str`	Нет
`set`	Да
`frozenset`	Нет
`dict`	Да

Список

это стандартная **изменяемая** коллекция, которая позволяет хранить различные типы данных в одном месте и под одним именем.

Списки можно создавать с помощью слова `list` либо с помощью **квадратных скобок**.

```
list1 = list()
list2 = []
```

Для создания списка с элементами, можно записать так: функция `list()` может принимать любой другой итерируемый объект:

```
list1 = list("hello")
list2 = ["h", "e", "l", "l", "o"]
```

Список

Благодаря *динамической типизации* в списках можно хранить **разные типы** данных, т.е. в одном списке одновременно можно хранить и строку, и число, и любой другой тип.

```
list3 = [True, "h", 1, "e", 2, "l", 3, "l", 4, "o", None]
```

Выводить списки можно с помощью функции `print()`, тогда результат вывода является всеми элементами списка.

```
[True, 'h', 1, 'e', 2, 'l', 3, 'l' 4, 'o', None]
```

Индексирование и срезы списков

Списки, как и строки, подчиняются **индексированию**.

```
print(list3[0])
print(list3[-1])
print(list3[1])
print(list3[-3])
```

Срезы также работают на списках, благодаря им можно получить новый список с выборкой данных по числам, передаваемых в срез.

```
print(list3[1:5])
print(list3[7:4:-1])
print(list3[::1])
```

Индексирование и срезы списков

Элементы списка можно **изменять**, обратившись к ним по индексу.

```
list3 = [1, 2, 3, 4, 5]
list3[0] = 10
list3[4] = "hello"
print(list3)
```

Операции списков

Списки можно объединять с помощью оператора **+**

```
list8 = [1, 2, 3]
list9 = ["1", "2", "3"]
print(list8 + list9)
```

Списки можно разложить, передав переменные элементам списка:

```
list8 = [1, 2, 3]
a, b, c = list8
print(a, b, c)
```

Операции списков

Списки можно сравнивать на соответствие, а также на соответствие в памяти

```
lst1 = [1, 2, 3]
lst2 = [1, 2, 3]
lst3 = lst1
print(lst1 == lst2)
print(lst1 is 1st2)
```

Функция `range()`

Функция `range(start, end, step)`, принимает **три** ключевых параметра и создаёт итерируемый объект - последовательность из чисел, начиная с начала `start`, не включаемого конца `end` и шага `step`.

Функция range()

С помощью функции `range()` можно создавать списки, состоящие из **чисел**.

- Когда передаётся **3 параметра**, то это **начало** последовательности, **конец** и **шаг**.
- Когда передаётся **2 параметра**, то это **начало** последовательности и **конец**, а шаг передаётся автоматически 1.
- Когда передаётся **1 параметр**, то это **конец** последовательности, а начало – автоматически 0, шаг – автоматически 1.

```
list5 = list(range(0, 10, 1))
list6 = list(range(7, 106))
list7 = list(range(20))
```

Методы для работы с списками

Метод	Описание
`list.append(variable)`	Добавляет элемент в конец списка
`list.extend(variable)`	Расширяет список list, добавляя в конец все элементы списка variable
`list.insert(variable, index)`	Вставляет значение variable на позицию index
`list.remove(variable)`	Удаляет первый элемент в списке, имеющий значение variable. ValueError, если такого элемента не существует
`list.pop(index)`	Удаляет элемент на позиции index и возвращает его. Если индекс не указан, удаляется последний элемент
`list.index(variable)`	Возвращает положение первого элемента со значением variable
`list.count(variable)`	Возвращает количество элементов со значением variable
`list.sort([reverse=True])`	Сортирует список. Если указан ключевой параметр reverse с значением True, тогда список сортируется в обратном порядке
`list.reverse()`	Разворачивает список
`list.copy()`	Поверхностная копия списка
`list.clear()`	Очищает список

Функции для работы с списками

Функция	Описание
`len(list1)`	Возвращает количество элементов в списке
`sum(list1)`	Возвращает сумму элементов списка. Имеется ввиду, что список содержит только числа
`max(list1)`	Возвращает максимальный элемент списка. Имеется ввиду, что список содержит только числа
`min(list1)`	Возвращает минимальный элемент списка. Имеется ввиду, что список содержит только числа

Типизация списков

Элементы списка можно **изменять**, обратившись к ним по индексу.

Название метода	Описание
`str(list1)`	возвращает строку, при этом стандартный вывод списка сохраняется, убрать квадратные скобки и запятые можно с помощью функции replace()
`set(list1)`	возвращает множество с элементами списка
`frozenset(list1)`	возвращает неизменяемое множество с элементами списка
`tuple(list1)`	возвращает элемент списка. Имеется ввиду, что список содержит только числа
`sorted(list1)`	сортировка списка
`del list1[index]`	удаляет элемент списка list1 на позиции index

Вложенные списки

это списки, которые хранятся внутри друг друга.

Примеры создания вложенных списков, данные внутри них могут отличаться:

```
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
print(matrix)
```

```
matrix = [[1, 2, 3],
          "1 2 3",
          True]
print(matrix)
```

! Вложенные списки классифицируются по глубине. Если список внутри списка - это **двухмерный** список и его глубина - 2. Если список в списке внутри списка - это **многомерный** список и его глубина - 3.

Кортежи

Кортеж – это индексированная структура данных, похожая на список, предназначенная для безопасного извлечения и отправки данных, а также для безопасного их хранения.

Кортеж – это **неизменяемая** коллекция, которая содержит ряд преимуществ:

- данные, хранящиеся в кортеже, невозможно изменить, они полностью защищены от “случайных” изменений;
- кортеж как структура занимает меньший вес в памяти;
- кортеж можно использовать ключом словаря.

Кортежи

Для создания кортежа используются **круглые скобки** либо функция `tuple()`.

! Если заключить данные просто в круглые скобки, то кортеж не будет создан. Для того, чтобы создать кортеж с одним элементом, нужно поставить после него **запятую**. Например: `(1, )` - это кортеж.

```
list1 = [1, 2, 3]
tuple1 = tuple(list1)
tuple2 = (1, 2, 3)
```

На кортежи действую все методы и функции списков, которые **не изменяют** данные внутри структуры.