Коллекции данных

Термины, используемые на уроке

Коллекция данных

Структура данных, позволяющая сохранять несколько объектов данных в себе и выполнять с ними определенные операции.

Sequence type — тип последовательных данных, основа для последующих типов.

Тремя базовыми последовательными типами являются

1. list — список
2. tuple — кортеж
3. range — диапазон

List и Tuple в Python

Это классы структур данных Python. Список является динамическим, тогда как кортеж имеет статические характеристики.

Это означает, что списки могут быть изменены, тогда как кортежи не могут быть изменены, кортеж быстрее списка из-за статичности по своей природе. Списки обозначаются квадратными скобками, а кортежи обозначаются скобками.

Различия между List и Tuple в Python

List (список)	Tuple (Кортеж)
Списки изменяемы	Кортежи неизменны
Последствия итераций отнимают много времени	Последствия итераций сравнительно быстрее
Список лучше подходит для выполнения операций, таких как вставка и удаление.	Тип данных Tuple подходит для доступа к элементам
Списки потребляют больше памяти	Кортеж потребляет меньше памяти по сравнению со списком
Списки имеют несколько встроенных методов	Tuple не имеет большого количества встроенных методов
Более вероятны непредвиденные изменения и ошибки	В кортеже это трудно осуществить

Функция range

Функция range возвращает объект, создающий последовательность чисел, начинающуюся с 0 (по умолчанию), последовательно увеличивающуюся (по умолчанию на 1)
и останавливающуюся перед заданным числом (обязательный параметр).

range возвращает не последовательность, а объект, вызывающий её.

Чтобы получить последовательность, объект нужно проитерировать. Благодаря тому, что функция range() относится к перезапускаемым генераторам (т. е. для неё можно создавать сколько угодно итераторов, и для каждого из них значения будут генерироваться заново), вызвать объект достаточно один раз.

Типы данных

Которые являются типами последовательности, имеют общие свойства.

Например, над такими типами возможны следующие операции:

x in s, x not in s: проверка вхождения, невхождения элемента в контейнер.

s + t: объединение двух контейнеров (кроме range).

s * n или n * s: добавление s самого к себе n раз (кроме range).

s[i]: доступ к элементу контейнера.

s[i:j], s[i:j:k] — слайсы.

len(s) — число реальных элементов в контейнере.

min(s), max(s): можно находить минимальное и максимальное значения.

s.index(x[, i[, j]]): нахождение места первого вхождения x в s.

s.count(x): сколько x встретился в s.