Управляющие конструкции и коллекции (часть 2)

Видео 1:

Циклы. Функции range и enumerate

Цели занятия

- Рассмотреть на практике применение функций range и enumerate в решении заданий со списками и циклами.
- Познакомиться с работой структуры list comprehension и множествами, узнать их преимущества в работе с данными.
- Рассмотреть словари как тип данных и методы работы с ними.

Функция range

Многие задачи можно решить не прямо, используя циклы while и for для перебора элементов, а используя разные вспомогательные функции.

Range (англ.) — диапазон.

Функция range умеет создавать списки из целых чисел, используется для перебора списка и экономии оперативной памяти.

На выходе создаёт генератор — объект, который пошагово выдаёт все элементы по одному. Расхода оперативной памяти при этом практически нет.

При переводе диапазона в список все элементы списка появляются в оперативной памяти.

Для перебора объектов используют цикл for. При обращении к генератору range в каждом шаге цикла будет одно значение. Это удобно для перебора больших структур.

Можно указывать:

- правую границу, если параметр один;
- левую и правую границы — в этом случае правая граница в результат не входит;
- шаг — перечисление всех указанных элементов через заданный шаг.

Функция enumerate

Функция enumerate может перебирать списки и к каждому элементу автоматически добавлять номер строки.

Это нужно для перебора длинного списка.

При проверке алгоритмов с использованием оператора break можно прервать цикл.

Выводы

Функции range и enumerate возвращают не готовый список значений, а генератор, позволяющий выводить из памяти каждое значение по одному. Функция enumerate подходит для перебора сложных вложенных структур. В решении практических задач рекомендуется писать программу последовательно, используя оператор break.

Видео 2

Списки. List comprehension. Множества

List comprehension

Структура, позволяющая сократить и упростить код.

Подходит для простых задач, но не подходит для сложных преобразований со множеством условий и вычислений.

Преобразование цикла в list comprehension:

1. записать цикл в [ ];
2. указать слева от цикла, как нужно преобразовать элементы цикла;
3. указать условия преобразования справа от названия последовательности.

Множества

Множества (sets) — «контейнер», содержащий неповторяющиеся элементы в случайном порядке.

Преимущества множеств:

1. Все элементы множества уникальны.
2. Элементы множества представляют собой хеш-таблицы — особую структуру данных, позволяющую быстро искать элементы по множеству.

Множества инициализируются при помощи **set()**. Как правило, создаются из списков.

Поиск по множеству чаще всего не зависит от числа элементов.
Часто множества используют, чтобы сравнивать и выделять разницу между списками.

Операции с множествами

- **.add(el)** добавляет элемент в множество.
- **.update(set)** соединяет множество с другим множеством/списком.
- **.discard(el)** удаляет элемент из множества по его значению.
- **.union(set)** объединяет множества (логическое ИЛИ).

- **.intersection(set)** — пересечение множеств (логическое И).
- **.difference(set)** возвращает элементы одного множества, которые не принадлежат другому множеству (разность множеств).
- **.symmetric_difference(set)** возвращает элементы, которые встречаются в одном множестве, но не встречаются в обоих.

Выводы

List comprehension — структура, позволяющая упростить код. Она не подходит для сложных задач со множеством условий. Множества — отдельный тип данных, набор неповторяющихся элементов в случайном порядке. Основные преимущества множеств — уникальность каждого элемента и быстрый поиск элементов.

Видео 3

Словари

Словари (dictionaries) — неупорядоченные коллекции произвольных объектов с доступом по ключу.

Ключи и значения

Словарь инициализируется при помощи **{ }**. В словаре элементом являются два значения: ключ (key) и значение (value). В словаре элементы хранятся в формате **key:value**.

Ключами словаря могут быть **strings, booleans, integers и floats**.

Ключ, по которому можно найти значение, пишется слева. После двоеточия пишется соответствующее ему значение. Значением словаря может быть что угодно: числа, списки, словари словарей и любые вложенные структуры.

Любое значение из словаря можно получить следующим образом: **my_dict[key]**

Все ключи в словаре должны быть уникальными. Это позволяет быстро искать нужные значения среди словарей.

Операции со словарями

Чтобы обратиться к элементу словаря, пишут название словаря и в [ ] указывают название ключа. Таким образом можно сразу получить значение, соответствующее ключу.

- **del(dict[key]**) удаляет элемент из списка по ключу.
- **.keys()** позволяет получить все ключи словаря.
- **.values()** позволяет получить все значения словаря.
- **.items()** позволяет получить ключи и значения словаря.

- **.get(key)** безопасно возвращает значение по ключу — при отсутствии ключа ошибка не возникает.

Два способа проверить наличие ключа в словаре:

1. Через операторы if и else.
2. Метод **setdefault** проверяет, есть ли в словаре ключ. Если ключа нет, он по умолчанию назначает ему значение, стоящее справа.

Способы проходить словари:

1. **dict.keys()** — перебор по ключам.
2. **dict.value()** — перебор по значениям.
3. **dict.items()** — получить информацию обо всём словаре: на выходе получится список из кортежей, каждый элемент которого — ключ и значение.

Функция zip в словарях

Функция zip совмещает списки.

При работе со словарями функцию zip используют для создания словаря.
Чтобы создать словарь, перед функцией zip добавляют **dict**.

Dict comprehension

Аналог list comprehension.

Необходимо указывать пары значений.

Чтобы получить словарь, ставят **{ }**
Для каждого значения пишут парное значение через **:**
Используется для преобразования структуры и быстрого формирования словаря.

Выводы

Словари — неупорядоченные коллекции произвольных объектов, состоят из пары значений: ключа и значения. Словари — сложная структура данных, предполагающая определённые методы работы с ними. Функция zip позволяет получить готовый словарь из двух списков. Dict comprehension — структура, позволяющая преобразовать сложный список в словарь.

Общие итоги

1. На практике увидели применение функций range и enumerate в решении заданий со списками и циклами.
2. Познакомились с работой структуры list comprehension и множествами, узнали их преимущества в работе с данными.
3. Рассмотрели словари как тип данных и методы работы с ними.