Управляющие конструкции и коллекции (часть 1)

Видео 1:
Введение в типы данных

Цели занятия

- Познакомиться с простыми типами данных и увидеть методы работы с ними на практике в коде.
- Узнать, как работать со строками с помощью индексации и срезов и как форматировать строки.
- Рассмотреть понятие списков, их особенности и принципы работы.
- Разобрать применение различных операций со списками на практике в коде.
- Выяснить, как совмещать работу строк и списков, разобрать понятие кортежей и особенности их работы.
- Познакомиться с циклами while и for и узнать, как работать с ними в коде.

Базовые типы данных

1. **Integer** — целые числа. Обозначаются типом **int**.
2. **Float** — вещественные числа (с плавающей точкой).
3. **String** — строка/текст.
4. **Boolean** — булевый/логический тип. Частный случай целых чисел. Значения **true** и **false** ведут себя как целые числа 1 и 0.

Разные типы данных требуют разного количества оперативной памяти. К каждому типу данных применяется свой набор функций для их обработки. Соответственно, каждый тип данных служит своей цели.

Тип объекта можно узнать при помощи функции **type ()**.
Тип данных можно принудительно изменить функциями **int ()**, **float ()**, **bool ()**, **str ()** и др.

Важно!
Число, записанное через точку, будет автоматически считаться типом **float** и применять соответствующие методы.

Операции со строками

1. Конкатенация (объединение) строк возможно при помощи **+**.

2. Умножение строки на число позволит повторить её нужное количество раз.
3. **.upper()** приводит строку к верхнему регистру.
4. **.lower()** приводит строку к нижнему регистру.
5. **.capitalize()** приводит первую букву к верхнему регистру.
6. **.replace('что заменить', 'на что заменить')** заменяет элемент в строке на указанный.
7. **len(my_string)** позволяет определить длину строки (количество символов в ней.
8. Доступ по индексу.

Выводы

Основные типы данных в Python — целые числа типа int, вещественные числа типа float, булево значение и строки типа str. Строки — это особый тип данных, и при работе с ними используются специальные методы.

Видео 2:
Индексация и срезы строк

Индекс

Индекс — номер элемента.

Индексация строк — это доступ к элементам объекта по их порядковому номеру в нём. Индексация элементов начинается с нуля.

Получить значение элемента по индексу можно при помощи **[ ]** . Номер элемента можно отсчитывать слева направо по номеру индекса.

Отрицательный индекс — индекс при отсчёте с конца. Последний элемент справа будет иметь индекс [-1].

Срезы строк

Можно «доставать» из строки несколько элементов при помощи **срезов (slicing)**. Для указания интервала среза используется двоеточие **:** .

Через дополнительное двоеточие указывается шаг.

Важно!
При указании срезов крайний правый элемент не будет входить в результат.

Один из диапазонов можно не указывать:
- если указан только левый диапазон, система вернёт все варианты от указанного индекса до конца;
- если не указан первый индекс, система возьмёт по умолчанию первый индекс 0.

Операторы проверки вхождения

При работе с числами и строками используются следующие операторы:

1. **IN** — возвращает True, если элемент входит в объект.
2. **NOT IN** — возвращает True, если элемент не входит в объект.

Форматирование строк (f-строки)

Добавляя префикс **f** к строке, можно встраивать в неё произвольные выражения при помощи фигурных скобок **{ }** .

Выводы

Индексация — универсальный инструмент, позволяющий перебирать элементы, фильтровать их и выполнять прочие операции. Индексация и срезы работают в строках, списках и других типах данных.

Видео 3:
Списки

Списки (list) — структура данных для упорядоченного хранения объектов различных типов. Является изменяемым типом данных, в отличие от предыдущих. Каждый элемент списка имеет свой индекс.

Список инициализируется при помощи **[ ]** , элементы в списке разделяются запятыми. В одном списке могут быть одновременно элементы разных типов и даже другие списки.

Как работать с элементами списка

1. Перебор элемента в цикле.
2. Работа с индексом элемента.

Принципы работы с индексами списка аналогичны принципам работы с индексами строк: переборы по индексам, срезы и т. д.

Важно помнить, что первый элемент списка имеет индекс 0.

Если много уровней вложенности, номера элементов перечисляются подряд через **[ ]** .

Изменение списков

В отличие от кортежей и множеств, значения списков можно изменять.

При обращении к элементу списка и назначении ему нового значения это значение в списке сразу поменяется.

Неочевидные способы изменения списков:

- в знаке равенства слева и справа расположить сразу несколько значений. Таким образом в одной строчке можно поменять сразу несколько значений;
- слева записать переменные через запятые, а справа — список желаемых значений;
- сложение списков — это операция конкатенации, а не сложение значений списков, таким образом в конец списка добавляются элементы другого списка.

Выводы

Списки состоят из произвольного набора элементов (чисел, строк или вложенных списков). У каждого элемента списка есть свой индекс. Правила перебора индексов полностью совпадают с принципами работы индексов в строках. При сложении списков элементы одного списка добавляются к другому списку.

Видео 4:
Операции со списками

Распаковка списков

Если нужные значения содержатся в списке и каждой переменной нужно назначить значение:

- можно каждой переменной назначить значение в отдельной строчке кода;
- слева перечислить переменные, а справа — их значения. Python автоматически назначит эти переменные;
- если столбцов много, сначала пишутся все переменные, которые нужны, и через ***_** указывается любое название переменной. Все остальные элементы будут автоматически записаны в эту переменную. Последние переменные могут быть назначены таким же образом.

Операции со списками

- Списки можно складывать.
- **del(list[index])** удаляет элемент из списка по индексу.
- **.remove(el)** удаляет указанный элемент из списка.
- **.append(el)** позволяет добавить элемент в список.
- **.count(el)** считает количество вхождений элемента в список.
- **.index(el)** позволяет узнать индекс элемента в списке.
- **.reverse()** разворачивает список.
- **sorted(list)** сортирует список.

Важно! В Python все структуры по умолчанию сортируются по возрастанию. Если нужно поменять порядок сортировки, нужно написать параметр **reverse= True**. В этом случае будет применена сортировка по убыванию.

Сортировка списков со строками и с числами будет различаться:

- с числами все элементы сортируются по убыванию или по возрастанию числового значения;
- элементы строк сортируются по алфавиту.

Математические операции

- **Функция len** возвращает количество элементов в списке.
- **sum** — арифметическая сумма всех значений.
- **max** — максимальный элемент элементов.
- **min** — минимальный элемент элементов.

Изменение списков: модуль copy

Чтобы приравнять список к другой переменной или что-то с ним сделать, необходимо использовать **модуль copy**. Он позволяет создать копию списка. Таким образом при внесении изменений в копию списка оригинал останется неизменным.

Выводы

Существует большое количество возможных операций со списками. В случае, когда необходимо сделать копию списка и провести над ней вычисления, необходимо использовать модуль copy.

Видео 5:
Списки и строки. Кортежи

Как совмещать работу списков и строк

- Метод **split** — преобразование строки в список. Список дополнительно можно разделять. В качестве разделителя может выступать запятая, пробел или другой разделитель.
- Метод **join** — преобразование списка в строку. В преобразовании сначала идёт разделитель, к которому применяется метод join. В качестве параметров указывается определённый список.
- Проверка вхождения элемента в список возможна через операторы **in** или **not in**.

Кортежи

Кортежи (tuples) — неизменяемые списки. Нельзя добавлять или удалять элементы из уже созданного кортежа.

Кортежи инициализируют при помощи **( )** . При работе с кортежами требуется меньше оперативной памяти по сравнению со списками.

Кортежи часто используются в качестве ключей словарей.

Основное отличие списка от кортежа в том, как его задают:

- список — в **[ ]** ;
- кортеж — в **( )** .

Кроме того, элементы кортежа нельзя изменить.

Кортеж из одного элемента задают в **( )** , после элемента обязательно ставят запятую. Без запятой получится просто строка.

Функция zip

Функция **zip(list_1, list_2, ...)** берёт на вход несколько списков и создаёт из них специальный zip-объект, состоящий из кортежей.

Первый элемент полученного объекта содержит кортеж из первых элементов всех списков-аргументов.

В случае, когда сразу необходимо получить результат, можно использовать функцию **list**. Весь результат вернётся в оперативную память. Каждый элемент списка будет представлять собой кортеж.

Выводы

Совместить работу списков и строк возможно с помощью методов split и join. Кортежи — это неизменяемые списки. Их преимущества в том, что они занимают меньше оперативной памяти. Функция zip позволяет склеивать два списка и более в отдельную структуру. В этом случае каждый элемент будет представлять из себя кортеж.

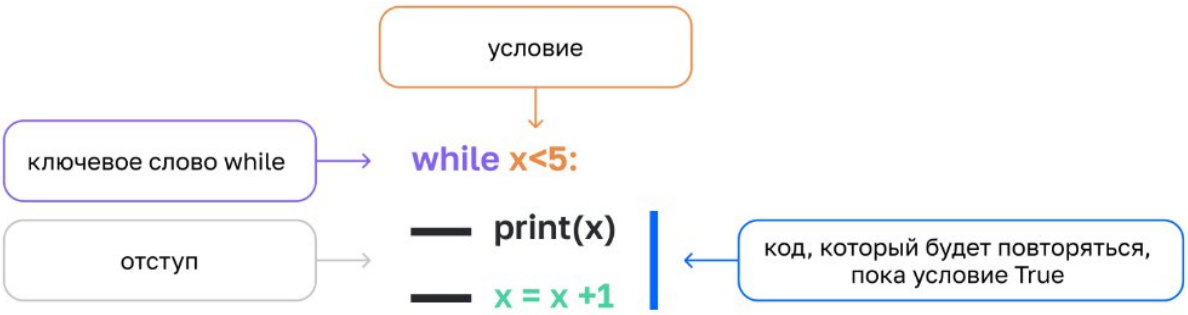
Видео 6:
Циклы while и for

Циклы позволяют организовать повторение выполнения участков кода.

В Python существует два типа циклов: цикл **while** и цикл **for**.

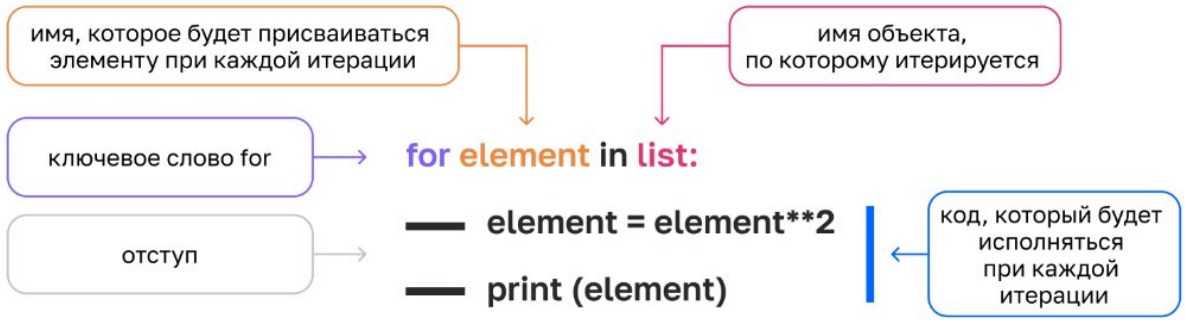
Цикл while

Позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Как правило, цикл **while** используют, когда невозможно заранее определить точное значение количества проходов исполнения цикла:



Цикл for

Цикл **for** проходится по элементам любого итерируемого объекта (строки, списка и т. д.) и во время каждого прохода выполняет заданную последовательность действий:



Ключевые слова

- **break** прерывает исполнение цикла.
- **continue** завершает исполнение текущей итерации цикла и переходит к следующей итерации.
- **pass** игнорирует условие и продолжает исполнение цикла.

Выводы

Основные циклы в Python — циклы while и for. Их используют для решения различных задач. Чтобы создать особые способы изменения работы цикла, используют операторы break, pass, continue.

Общие итоги

1. Познакомились с простыми типами данных и посмотрели методы работы с ними на практике в коде.
2. Узнали, как работать со строками с помощью индексации и срезов и как форматировать строки.
3. Рассмотрели понятие списков, их особенности и принципы работы.
4. Разобрали применение различных операций со списками на практике в коде.
5. Познакомились с циклами while и for и узнали, как работать с ними в коде.