top

КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Основы Python

Тема «Циклы: for и while. Исключения и обработка ошибок»

Сегодня мы познакомимся с **циклами и исключениями в Python.**

Представьте, что нужно выполнить одно и то же действие много раз. Например, если мы решили посчитать количество яблок в каждой корзине на огромном складе. Без циклов нам пришлось бы писать одну и ту же команду снова и снова для каждой корзины. Но с циклами мы можем сказать программе: "Повторяй это действие для каждой корзины!", и она выполнит всю рутинную работу за нас.

Сегодня мы:

- **узнаем,** как автоматизировать повторяющиеся задачи с помощью циклов for и while. Это поможет сделать код более кратким и избежать лишних повторений.

- **научимся** использовать конструкции try, except для обработки ошибок в коде. Это важно для создания надежных программ, которые могут продолжать работу даже при возникновении проблем.

Глоссарий ко второму занятию

For (Цикл for) — Цикл, который проходит по каждому элементу последовательности, такой как список или строка.

While (Цикл while) — Цикл, который продолжает выполнение, пока выполняется заданное условие.

Exception (Исключение) — Специальный механизм в программировании, предназначенный для обработки ошибок или необычных условий без прерывания всей программы.

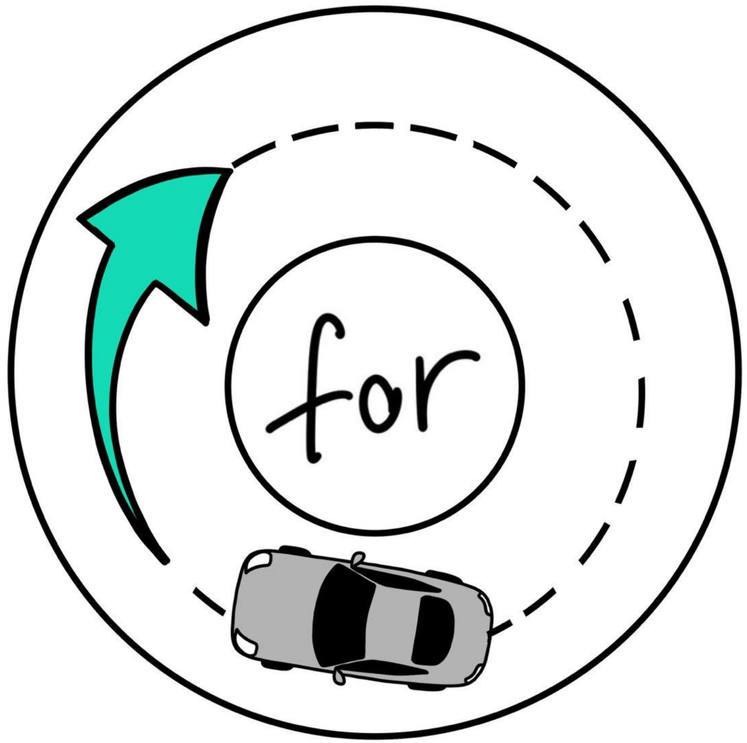
Что такое циклы?

Циклы — это инструмент, который помогает выполнять один и тот же набор действий несколько раз. Например, если мы хотим вывести числа от 1 до 10 или перебрать элементы списка, нам не нужно писать одну и ту же команду 10 раз. Достаточно написать цикл, и Python сделает всё за нас.

Цикл for

Представьте, что решили выпить 3 чашки чая. Мы точно знаем, сколько раз повторится наше действие — 3 раза. Это похоже на цикл for, где действие выполняется заранее известное количество раз.

Цикл for используется, когда нужно выполнить однотипное действие для каждого элемента списка, строки или другого набора данных.



Цикл while

Представьте, что пьёте чай, пока не почувствуете, что утолили жажду. Мы не знаем заранее, сколько чашек нам потребуется. Это похоже на цикл while, где действие продолжается, пока выполняется условие (мы еще хотим пить).

Таким образом, **цикл while** повторяет действия, пока выполняется определённое условие.

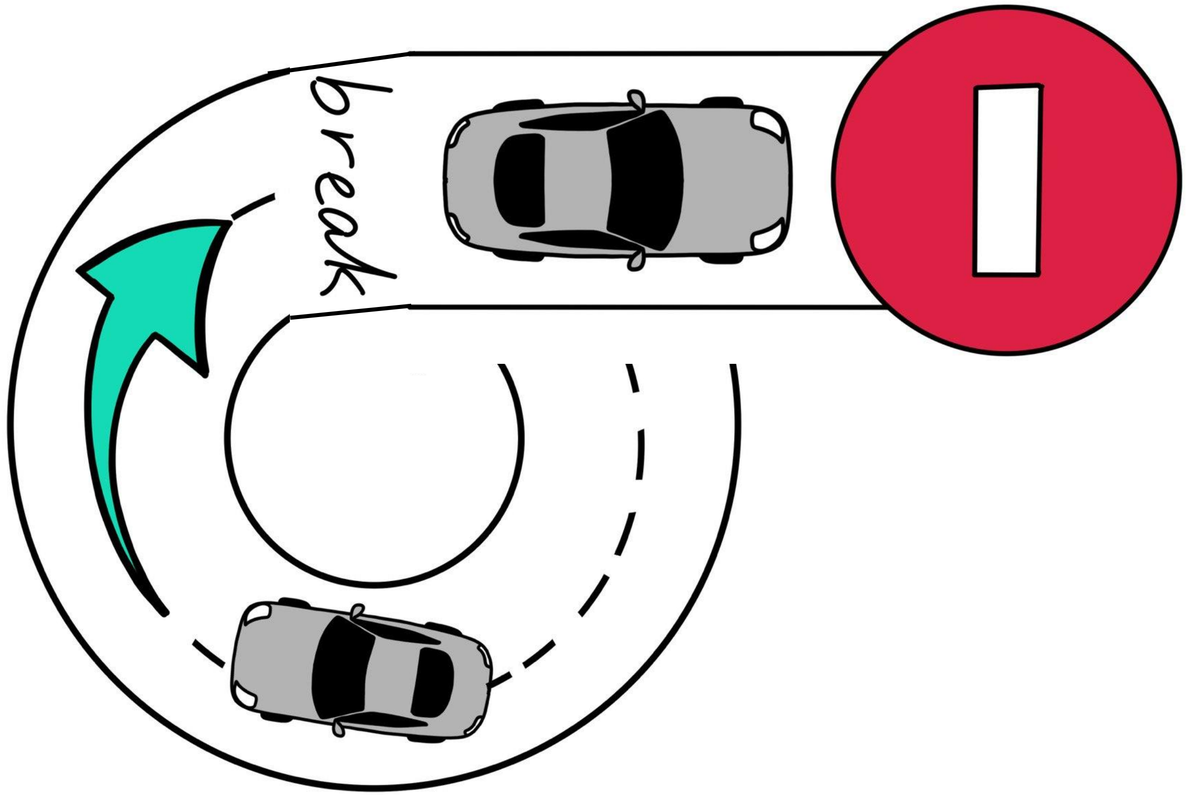
Управление циклами: команды break и continue

Иногда нужно досрочно остановить цикл или пропустить выполнение части кода. Для этого используются команды **break** и **continue**.

Команда break

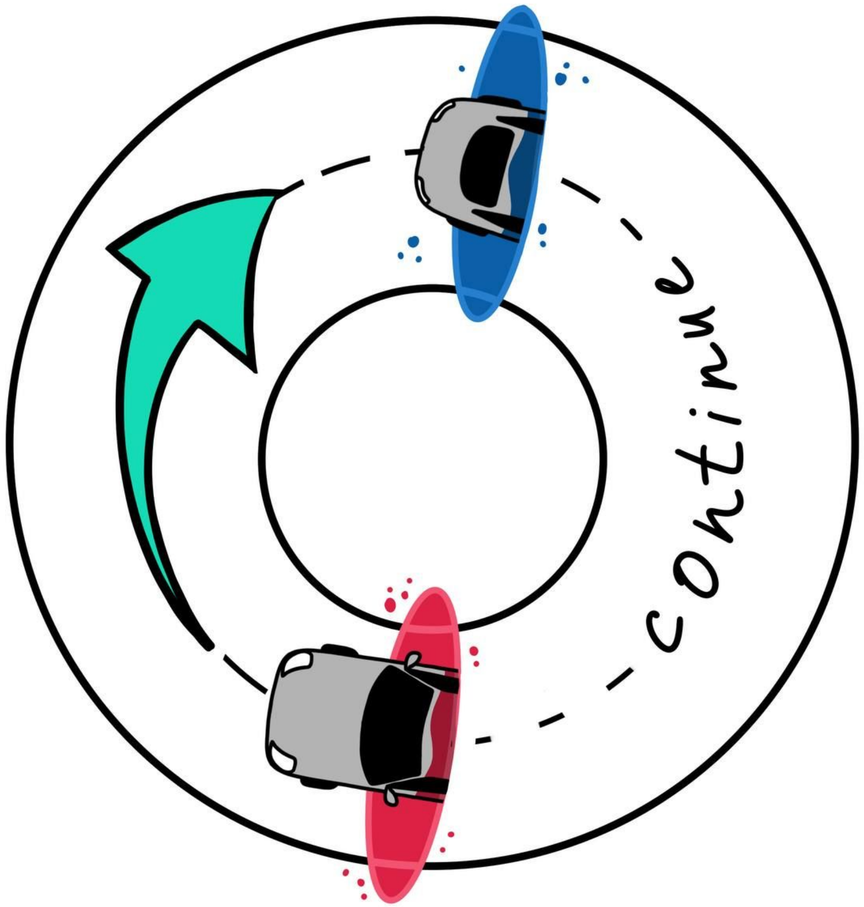
Останавливает выполнение цикла,

даже если условие еще истинно.



Команда continue

Пропускает текущую итерацию цикла и переходит к следующей.



Зачем нужны циклы?

1. Они экономят время и делают код короче.
2. Позволяют работать с большими наборами данных, не повторяя одинаковые команды.
3. Помогают управлять повторяющимися действиями (например, обработкой списка или выполнения задачи несколько раз).

Исключения (Exceptions) и обработка ошибок

Когда дело доходит до программирования, ошибки неизбежны.

Однако способность правильно их обрабатывать может сделать наш код не только более надежным, но и более удобным для пользователя.

В Python для обработки исключений используются **конструкции try, except, а также finally и else.**

Исключение

Вот некоторые из наиболее часто встречающихся исключений:

- ValueError: Ошибка возникает, когда функция ожидает определенного типа или значения, но получает что-то неподходящее.
- ZeroDivisionError: Попытка деления на ноль.
- FileNotFoundError: Попытка открыть несуществующий файл.
- IndexError: Обращение к элементу списка за пределами его длины.
- TypeError: Попытка выполнить операцию с объектом неподходящего типа.

Как работает обработка исключений?

Блок try и except:

Основной способ обработки исключений в Python — это использование блоков try и except.

Код, который может вызвать ошибку, помещается в блок try, а обработка возможной ошибки — в блок except.

Практикум

Задача 1: Учёт финансовых операций

Ситуация: мы хотим начать вести учёт своих ежедневных расходов и доходов, чтобы лучше контролировать свои финансы. Нам нужна программа, которая позволит записывать каждую финансовую операцию, включая сумму и назначение траты или поступления.

Задача: создать программу для учёта наших ежедневных финансовых операций, которая поможет записывать, сколько денег мы потратили или получили и на что именно.

Задача 2: Учёт расходов на обеды

Ситуация: мы решили анализировать свои ежедневные расходы на обеды в течение недели, чтобы понять, сколько денег уходит на питание вне дома. Нам нужно создать простую программу, которая поможет нам собрать данные о расходах за каждый день.

Задача: написать программу для учёта расходов на обеды за неделю.

Итоги занятия

- **научились** использовать циклы for и while, которые являются основными инструментами для автоматизации повторяющихся задач.
  С их помощью мы можем эффективно обрабатывать данные, выполнять операции многократно без необходимости писать один и тот же код снова и снова. Это не только упрощает написание кода, но и делает его более читаемым и удобным для отладки.

- **изучили,** как использовать блоки try, except, else и finally для обработки исключений.
  Это позволяет нашим программам адекватно реагировать на ошибки, которые могут возникнуть во время их выполнения. Исключения помогают предотвратить аварийное завершение программ и обеспечивают более гладкую работу, даже когда что-то идет не так, как планировалось.

Домашняя работа

Описание задачи: разработайте программу для учёта времени учебы за неделю. Программа должна позволять пользователю ввести количество учебных дней (не более семи), а затем для каждого дня вводить количество часов, посвящённых учёбе. Программа должна обрабатывать ошибки ввода, такие как ввод нечисловых значений и отрицательных чисел, и запрашивать повторный ввод при необходимости.

Указания:

1. Используйте цикл while для запроса количества учебных дней, обеспечивая, чтобы количество дней не превышало семь и было положительным числом.
2. Используйте цикл for для запроса количества часов учебы в каждый из указанных дней.
3. Примените блоки try и except для обработки исключений при вводе количества часов, убедитесь, что введены только положительные числа.
4. В случае ошибки ввода запросите данные повторно, пока не будут получены корректные значения.

Ожидаемый результат:

Программа должна корректно принимать количество учебных дней и количество часов учебы для каждого дня, эффективно обрабатывать возможные ошибки ввода, и в конце вывести общее количество часов учебы за неделю.

Критерии оценивания:

3 балла — Корректность: Программа точно следует заданным условиям, правильно обрабатывает ввод и выводит итоговую информацию.

3 балла — Обработка ошибок: Программа должна умело обрабатывать все возможные ошибки ввода без прерывания своей работы.

3 балла — Полнота выполнения задачи: Все заданные функции должны быть реализованы и работать корректно, включая проверку и подтверждение ввода.

3 балла — Код должен быть организован, легко читаем и содержать комментарии, где это необходимо, для объяснения функций и обработки ошибок.

Максимальный балл за задание — 12 баллов.