Основы языка Python. Интерактивный курс

Тернарный оператор

План

- Определение тернарного оператора.
- Применение.
- Синтаксис.
- Примеры использования.

Определение

- тернарный (ternarius — «тройной») оператор — операция, возвращающая свой первый или третий операнд в зависимости от значения логического выражения, заданного вторым операндом
- **результат 1** если **выражение_истинно,** иначе **результат 2**

Применение

- вместо конструкции if … else, в которой нет elif
- позволяет писать компактный и читаемый код

Синтаксис

- name = ‘Max’ if is_has_name else ‘Empty’
- number = 1 if is_one else 2
- print(‘Привет’ if is_russian else ‘Hello’)
- в общем виде: **результат1** если **условие** иначе **результат2**

Примеры использования

- От if к тернарному оператору

  слово -> СлОвО

- сразу пишем тернарный оператор

  проверка пароля пользователя

Python

Генераторы списков и словарей

План

- Определение генератора.
- Применение генераторов.
- Синтаксис.
- Преимущества и недостатки.
- Примеры.

Определение

- В Python существует специальная синтаксическая конструкция, которая позволяет по определенным правилам создавать заполненные списки. Такие конструкции называются генераторами списков.
- Генераторы словарей можно определить по аналогии.

Применение

- вместо цикла for
- позволяет писать компактный и читаемый код
- работают быстрее

Синтаксис

- [number for number in numbers if number > 0]

Преимущества

- компактный и читаемый код
- скорость

Недостатки

- Нельзя заменить очень сложные конструкции.
- При неправильном использовании могут ухудшить читаемость.

Примеры

- Создать список из случайных чисел от 1 до 100.
- Создать список квадратов чисел.
- Создать список имен на букву А.

Python

Принципы работы операторов and и or

План

- Приведение типов к bool в Python.
- Python — стиль записи логических выражений
- Как работает and.
- Как работает or.
- Примеры применения.

Приведение типов к bool в Python

Все встроенные типы данных в Python приводятся к логическому типу bool по определенным правилам:

- Истина: ‘abc’, [1], (1,), {1:’a’}, 10, 1.1, ...
- Ложь: ‘’, [], (), {}, 0, None, ...

Стиль записи логических выражений

Из-за данного преобразования типов в Python желательно использовать лаконичный стиль записи логических выражений:

- Вместо if len(str_var) > 0: ...
- Пишем: if str_var: ...

Это ускоряет разработку и делает код более читаемым.

Как работает and

- Оператор and не проверяет следующее логическое выражение если текущее False (ленивый).
- Оператор and возвращает первый ложный элемент или последний истинный.

Как работает or

- Оператор or не проверяет следующее логическое выражение если текущее True (ленивый).
- Оператор or возвращает первый истинный элемент или последний ложный.

Примеры применения

- and: извлечение квадратного корня из отрицательного числа
- or: сохранение в переменную одного из 2-х значений

Python

Модуль copy

План

- Хранение списков в памяти.
- Изменение элементов списка в функции.
- Копирование списка.
- Модуль copy.

Хранение списков в памяти

При работе со списками стоит помнить, что если мы переприсваиваем список в другую переменную a = b и меняем значения внутри нового списка b[1], значения изменятся и внутри старого списка a[1], т.к. ссылки на элементы списка остаются на своих местах в памяти и каждый список использует одни и те же элементы.

Изменение элементов списка в функции

При передаче списка параметром в функцию нужно быть особенно внимательными: функция может изменить элемент списка внутри основной программы.

Методы копирования списка

- Создание среза от начала и до конца списка my_list[:].
- Метод copy у самого списка.

Модуль copy

- Применяется для полного (глубокого) копирования списка.
- Используется функция deepcopy.
- b = copy.deepcopy(a).

Python

Обработка исключений

План

- Определение исключительной ситуации.
- Обработка исключений.
- Примеры.
- Генерация исключений.

Исключительная ситуация

Во время выполнения программы могут возникать ситуации, когда состояние внешних данных, устройств ввода-вывода или компьютерной системы в целом делает дальнейшие вычисления в соответствии с базовым алгоритмом невозможными или бессмысленными.

Классические примеры

- Деление на 0
- Ошибка чтения данных при отсутствии доступа к ресурсу
- ...

Обработка исключений

- Что делать при возникновении исключительной ситуации?
- Как определить, произошла исключительная ситуация или программа работает в нормальном режиме?

Python имеет встроенный механизм обработки исключений.

Обработка исключений

- try:
- Блок с возможной исключительной ситуацией.
- except:
- Код, который выполняется при возникновении исключительной ситуации.

Перехват конкретных исключений

- В Python каждая исключительная ситуация имеет свой тип.
- TypeError, ValueError, ...
- Самое общее исключение имеет тип Exception.
- Рекомендуется обрабатывать конкретные исключительные ситуации и реагировать на разные исключения по-разному.

Информация об ошибке

- Можно получить дополнительную информацию об исключении,

  Если использовать конструкцию except Исключение as e:

- Тогда в переменную e будет сохранен объект исключения.

try - except - else - finally

- Блок try — код, который может вызвать исключение.
- Блок except — что делать при возникновении исключения.
- Блок else — что делать, если исключения не произошло.
- Блок finally — выполняется всегда.

Генерация исключений

- Иногда требуется не обработать а, наоборот, создать исключительную ситуацию.
- Это можно сделать с помощью команды raise:
  raise Exception(‘Что то пошло не так’).