Основы языка Python. Интерактивный курс

Настройка среды для разработки на Python

План

- Установка интерпретатора Python.
- Проверка версии интерпретатора.
- Установка PyCharm IDE.
- Создание и запуск проектов в PyCharm.

Установка интерпретатора Python

- для Windows - скачать с официального сайта, установить.
- в Linux обычно уже есть 2 версии интерпретатора,
- будем использовать интерпретатор Python 3 версии.

Проверка версии интерпретатора

- в cmd (Windows) или в terminal (Linux),
- python --version,
- python3 --version (если установлено 2 версии: 2-я и 3-я).

Установка Pycharm IDE

- IDLE — встроенный редактор. Не рекомендуется.
- **Pycharm IDE.**
- Sublime Text, Atom, Notepad++, Vim.

Создание и запуск проекта в PyCharm

- Создать новый проект.
- Выбрать интерпретатор Python.
- Создать модуль, в котором будем писать код.
- Запустить из PyCharm.

Python

Переменные. Типы данных. Преобразование типов

План

- Зачем нужны переменные.
- Как объявить переменную.
- Как правильно называть переменные.
- Как определить тип переменной.
- Какие бывают типы.
- Как привести один тип к другому.

Зачем нужны переменные?

Переменные используются для хранения данных.

Переменные можно использовать несколько раз.

Можно менять значение и тип переменной.

Как объявить переменную?

имя_переменной = значение переменной

name = ‘Кеша’

Как правильно называть переменные?

Переменная должна называться так, чтобы по названию можно было понять её предназначение.

Верно

name, age, person_name, request, report

Неверно

a, b, c, ae, cp

Стиль имен переменных

маленькие буквы и знаки подчеркивания:

person_age

Тип переменной

Тип переменной определяет множество значений, которые могут быть ей присвоены и операции, которые могут быть с нею произведены. Он либо фиксирован в момент объявления переменной и соответствует одному из типов данных, предоставляемых языком программирования (статическая типизация), либо в каждый момент соответствует типу тех данных, что содержит переменная (динамическая типизация).

Самые простые типы:

- целое число — int
- число с плавающей точкой — float
- логический тип (истина/ложь) — bool
- ничего (пустой тип) — None
- строка — str (более сложный тип, будет рассмотрен отдельно)

Приведение типов:

- число к строке str(number)
- строка к числу int(word)
- любые другие преобразования аналогично

Python

Ввод, вывод

План

- Куда можно выводить информацию.
- Как можно использовать функцию print.
- Как вводить данные.
- Какой тип у введенных данных.

Куда можно выводить информацию

- GUI.
- WEB.
- Мобильное приложение.
- Хранилище, например файл или база данных.
- Консоль (Терминал).

print - дополнительные возможности

- вывод разных типов данных через ,
- использование разных разделителей слов (sep=)
- использование разных разделителей строк (end=)

Ввод данных. input()

- result = input()
- name = input(‘Как тебя зовут?’)

Тип введенных данных

- Что бы мы ни спрашивали у пользователя, для программы результатом ввода всегда будет строка (тип данных str).

Python

Арифметические и логические операции

План

- Арифметические операции.
- Приоритет арифметических операций.
- Логические операции.
- Сложные логические выражения.
- Приоритет логических операций.

Стандартные математические операции

- +
- -
- *
- /
- тип результата

Другие математические операции

- // - целая часть от деления
- % - остаток от деления
- ** - возведение в степень

Приоритет математических операции

- работает как в математике (умножение главнее сложения)
- круглые скобки () помогают управлять приоритетами

Логические операции

- == - равно
- != - не равно
- > - больше
- >= больше или равно
- < - меньше
- <= - меньше или равно

Сложные логические выражения

- and - и (ИСТИНА когда все ИСТИНА иначе ЛОЖЬ)
- or - или (ЛОЖЬ когда все ЛОЖЬ иначе ИСТИНА)
- not - не (ИСТИНА когда ЛОЖЬ, ЛОЖЬ когда ИСТИНА)

Python

Условные операторы

План

- Зачем нужны условные операторы.
- Оператор if в Python.
- Зачем нужны блоки и отступы в коде.
- Какие разновидности условных операторов есть в Python.
- Как применять условные операторы.

Зачем нужны условные операторы

- Начало сказки

  - Налево пойдешь — коня потеряешь.
  - Направо пойдешь — жизнь потеряешь.
  - Прямо пойдешь — счастье найдешь.

- Конец сказки

Оператор if

- Начало сказки
- if Налево пойдешь:

  коня потеряешь
- Конец сказки

Блоки и отступы в коде

- Блок кода — логически сгруппированный набор команд.
- В python нет {}, разделяющих блоки кода.
- Поэтому обязательно делать отступы для каждого блока кода.
- Отступ — 4 пробела (клавиша tab в pycharm).

if - elif - else

- if условие:
  - код1
- elif другое условие:
  - код2
- else:
  - код3

Вложенный if

- Внутри if-elif-else может быть другой if.
- Таких вложений может быть сколько угодно много.

Python

Понятие циклов. Цикл while

План

- Определение цикла.
- Примеры циклов в программировании и жизни.
- Цикл while.
- Примеры использования.

Цикл

Разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации **многократного** исполнения набора инструкций (из Википедии).

Цикл

Много раз выполняем один и тот же набор действий

Примеры циклов

- Ходить каждую неделю в университет до получения диплома.
- Отправлять запросы, пока есть адреса в списке.
- Качать героя в игре до получения максимального уровня.
- Ждать соединения клиента до его подключения.
- Мыть посуду, пока есть грязная.
- Задавать вопросы, пока пользователь не введет правильный ответ.

Цикл while

Пока условие выполняется (True): делать определенный набор действий.

while условие:

    действие1

    действие2

    ...

Использование

- Вывод чисел от 0 до 100
- Вывод чисел от 0 до n, n - вводит пользователь
- Вывод четных чисел от 0 до n

Python

break
continue
while - else

План

- break
- continue
- while - else
- Примеры использования

break

- Выход из цикла (не важно, выполнилось условие или нет).

continue

- Переход на следующую итерацию цикла (команды в цикле после continue не выполняются).

while-else

- В блоке else (после while) мы выполняем действия после того, как вышли из цикла while, когда условие цикла стало неверным (False)