Введение в язык Python

лекция 1

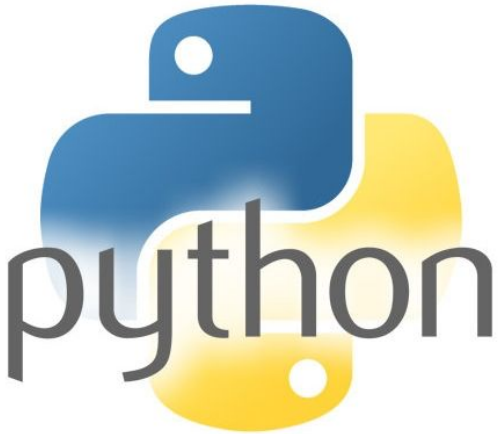
Козак А.В.
15.02.2020

Краткий план курса

- 14-16 лекций
- 3-4 лабораторные работы
- 1-2 контрольная работа
- зачет (письменный)
- экзамен (письменный и устный, 2 этапа)

Язык программирования Python

- появился в 1991 году
- создатель: Guido van Rossum
- Популярный версии: 2.7 и 3.X



Высокоуровневый динамический строго типизированный интерпретируемый язык программирования общего назначения.

Высокоуровневый

высокий уровень абстракции от деталей исполняющей системы

Динамический

позволяет определять типы данных и осуществлять синтаксический анализ и компиляцию «на лету», на этапе выполнения программы

Строго типизированный

язык не позволяет смешивать в выражениях различные типы и не выполняет автоматические неявные преобразования

Интерпретируемый

команды языка интерпретируются во время работы специальной программой-интерпретатором

Общего назначения

нет ограниченной области применения

(в противоположность domain-specific языкам)

Интерактивный

позволяет писать код прямо в оболочке интерпретатора и вводить новые команды по мере выполнения предыдущих

Интересные особенности

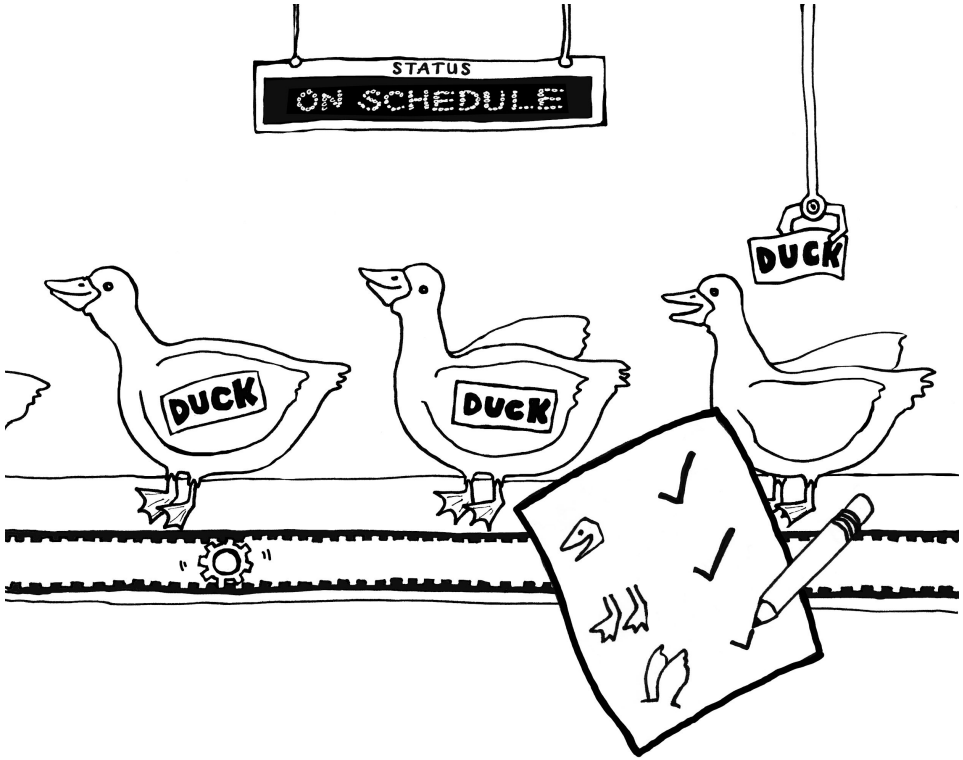
- отступы определяют структуру программы

```
a = 1
while a <= 10:
    if a==5:
        a += 1
        continue

    print(a ** 2)
    a += 1
else:
    print('Цикл завершен')
```

Интересные особенности

- Duck-typing – утиная типизация, характеризует обращение с объектами и вызов методов



Множество языков программирования. Зачем?

Отдельные языки помогают проще решать отдельные классы задач.

- C/C++
- C#, Java
- Python, Go

Зачем Python?

Python особенно популярен в задачах, где функциональность гораздо важнее эффективности: исследовательское программирование и написание прототипов.

Зачем Python?

Даже если нужно быстро, язык будет полезен просто как инструмент для написания небольших и средних серверных и вспомогательных прикладных программ для обработки данных.

Задачи, решаемые с помощью Python

- ➔ веб-разработка
- ➔ data science: машинное обучение, анализ данных и визуализация
- ➔ автоматизация процессов
- ➔ и другие

Хакартеристики

- ➔ Кроссплатформенный
- ➔ Мультипарадигменный (процедурное, ООП, функциональное, метапрограммирование и др.)
- ➔ Автоматическое управление памятью
- ➔ Возможность интеграции

Плюсы

- ➔ Легко разрабатывать – краткость и выразительность
- ➔ Легко читать – читабельность как одна из целей в основе дизайна языка
- ➔ Легко отлаживать – интерактивная проверка, можно всё динамически посмотреть

Минусы

Относительно медленно работает – частично компенсируется интеграцией с другими языками или использованием альтернативных интерпретаторов.

Минусы

Нет статических проверок компилятора – компенсируется качественным автоматизированным тестированием.

Интеграция с другими языками

- многие модули написаны на Си и C++
- проработанные механизмы для создания оберток к модулям на других языках
- механизмы использования Python из других языков

Суть Python

Язык был изначально задуман легко расширяемым:

небольшое ядро + библиотеки.

- стандартная библиотека
- внешние модули

Стандартная библиотека

- регулярные выражения
- юнит-тестирование
- логирование
- поддержка протоколов
- работа с БД
- многопоточность и многопроцессность
- системные модули
- и многое другое

Внешние модули

- веб-фреймворки и веб-сервера
- поддержка сетевых протоколов
- научные вычисления
- обработка текста
- обработка изображений

Zen of Python

- ➔ Beautiful is better than ugly.
- ➔ Explicit is better than implicit.
- ➔ Simple is better than complex.
- ➔ Complex is better than complicated.
- ➔ Flat is better than nested.
- ➔ Sparse is better than dense.
- ➔ Readability counts.

PEP – Python Enhancement Proposal

Документы стандартизированного формата, в которых содержатся предложения по развитию языка и описание уже имеющихся элементов.

Оформление и документирование кода

- PEP-8 – https://www.python.org/dev/peps/pep-0008/
- PEP-257 – https://www.python.org/dev/peps/pep-0257/

Anaconda

дистрибутив Python и R вместе с основными библиотеками для анализа данных и пакетным менеджером conda

Conda

- менеджер пакетов, позволяет устанавливать уже скомпилированные пакеты
- менеджер окружений системы, позволяет создавать окружения с разными версиями чего угодно (библиотеки C, низкоуровневые библиотеки и т.д.)

IPython и Jupyter Lab/Notebook

jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help | Python 3

Markdown CellToolbar

Simple spectral analysis

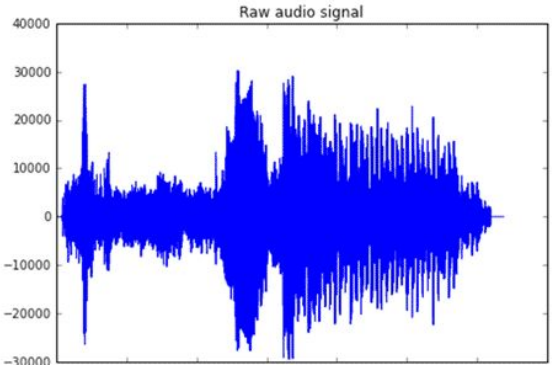
An illustration of the Discrete Fourier Transform

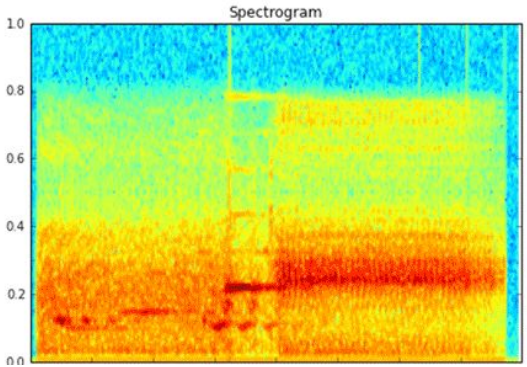
$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

In [2]:
```
from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

In [5]:
```
fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```





Оценка сложности алгоритмов

Рассмотрим модель, которая состоит из памяти и процессора, которые работают следующим образом:

- память состоит из ячеек, каждая из которых имеет адрес и может хранить один элемент данных;
- каждое обращение к памяти занимает одну единицу времени, независимо от номера адресуемой ячейки;
- количество памяти достаточно для выполнения любого алгоритма;
- процессор выполняет любую элементарную операцию (основные логические и арифметические операции, чтение из памяти, запись в память, вызов подпрограммы и т.п.) за один временной шаг;
- циклы и функции не считаются элементарными операциями.

Оценка сложности алгоритмов

```
s = 0
для каждого i от 1 до n делать:
    s = s + i
вывести s
```

Получаем оценку времени n или же O(n).
Оценка по памяти – 1+1 = O(1).

Алгоритмы и структуры данных

Основные структуры данных:

- списки (односвязные, двусвязные)
- массивы
- очередь
- хэш-таблицы
- множества
- кучи

Основные алгоритмы:

- сортировки
- бинарный поиск
- поиск порядковых статистик

Книги

1. Марк Пилгрим. Погружение в Python – быстрая вводная в python в виде книги [ru]
2. http://www.diveintopython.net/toc/index.html – обновляемая онлайн версия предыдущей книги [en]
3. Марк Лутц. Программирование на Python – подробное руководство в виде книги [ru]
4. http://docs.python-guide.org/en/latest/ – продвинутое руководство по использованию языка

Лабораторная работа 0

1. Установить Anaconda
2. Написать программу “Hello world!”
3. Запустить (с помощью командной строки)
4. “Поиграться” с IPython

Лабораторная работа 1

1. Изучить PEP8
2. Изучить, что такое git. Создать аккаунт на GitHub или GitLab (и использовать на протяжении всего семестра)
3. Изучить подробнее про оценку сложности алгоритмов
4. Написать тестовую работу по пунктам 1-3