Лекция 1: Введение в язык Python

Сергей Мыц

кафедра Информатики, БГУИР

предмет “ИГИ”, ИиТП, третий курс, осенний семестр 2015

Содержание лекции

Язык программирования Python

Источник названия

Создателю нравилась британская комедийная передача Monty Python's flying circus.

Fun to write

Основная идея: на языке должно быть удобно и приятно разрабатывать программы.

В каком году появился?

В каком году появился?

1991 год

Создатель

Guido van Rossum

BDFL

Benevolent dictator for life
Великодушный пожизненный диктатор

О развитии языка

- Python имеет длинную историю развития
- Более подробно можно почитать самостоятельно в интернетах
- Отдельные интересные статьи http://python-history.blogspot.com
- Сейчас активно используются версии языка 2.7.x и 3.x

Что такое Python?

Высокоуровневый динамический строго типизированный интерпретируемый язык программирования общего назначения.

Высокоуровневый

Высокий уровень абстракции от деталей исполняющей системы.

Динамический

Типы выводятся и проверяются во время выполнения (run-time).
С версии 3.5 можно также использовать compile-time проверки.

Строго типизированный

Сильно ограничены неявные приведения типов (строка + число и т.п.).

Общего назначения

Нет ограниченной области применения
(в противоположность domain-specific языкам).

О чём могли слышать

- Отступы – определяют структуру программы
- Duck-typing – утиная типизация, характеризует обращение с объектами и вызов методов

Различные языки

Есть множество различных языков программирования.
Почему?

Различные классы задач

Отдельные языки помогают проще решать отдельные классы задач.

Примеры языков

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

У питона также есть своя ниша.

Содержание

- Кроссплатформенный
- Мультипарадигменный (процедурное, ООП, функциональное, метапрограммирование и др.)
- Автоматическое управление памятью (refcounting + cycle-detecting garbage collector)

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.

Содержание

Интеграция с другими языками

- Многие модули написаны на Си и C++
- Проработанные механизмы для создания обёрток к модулям на других языках
- Механизмы использования Python из других языков

Python и платформы

- JVM – Jython
- .NET – Iron Python

Содержание

Расширяемость языка

Язык был изначально задуман легко расширямемым:
небольшое ядро + библиотеки.

Библиотеки

- Стандартная библиотека
- Внешние модули

Богатая стандартная библиотека

- регулярные выражения
- юнит-тесты
- логгирование
- поддержка различных стандартных форматов и протоколов
- работа с базами данных
- многопоточность и многопроцессность
- и многое другое

Огромное количество внешних модулей

- web-фрэймворки
- web-сервера
- поддержка различных сетевых протоколов
- фрэймворки тестирования, автоматизации, документирования, системного администрирования
- web-crawler-ы
- научные вычисления, обработка текста и изображений

Содержание

Дзен

Набор основополагающих тезисов в основе развития и использования языка

The Zen of Python, by Tim Peters

- 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.

The Zen of Python, by Tim Peters

- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.

The Zen of Python, by Tim Peters

- In the face of ambiguity, refuse the temptation to guess.
- There should be one– and preferably only one –obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.

The Zen of Python, by Tim Peters

- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!

PEP – Python Enchancement 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/

Pythonic

Понятие pythonic используется для характеристики решений, элементов и практик, которые соответствуют идеям языка Python.

Содержание

Когда функциональность важнее эффектиности

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

Вспомогательный инструмент

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

Веб и сервера

Позволяет просто реализовать весь набор элементов веб-приложения любой сложности.

Где используется

В значительном количестве крупных компаний разрабатывающих ПО или обрабатывающих данные так или иначе используется Python.

Где используется

Пруфы

- [ru.wiki] http://bit.ly/12ycKDm
- [en.wiki] http://bit.ly/1iAj5Gy.

Содержание

Интерпретаторы

- CPython – стандартная реализация
- Jython – JVM
- Iron Python – .NET
- PyPy – JIT-компиляция и написан на языке с python-подобным синтаксисом
- Stackless Python – вместо стека вызовов функций из C использует собственную реализацию
- и др.

Версии языка

- Основной выбор: 2.7.x или 3.x
- Почитать про различия можно тут http://bit.ly/1gcF18X
- Мы в этом курсе будем пользоваться 2.7
- https://www.python.org/downloads/release/python-2710/

Мотивация выбора 2.7

- Всё ещё широко используется
- Будет меньше проблем с совместимостью туториалов, документации, описаний найденных в интернете
- Не будет проблем с совместимостью библиотек
- При желании, несложно самостоятельно перейти на использование версии 3.x

Содержание

Интерактивная оболочка IPython

Расширенная по сравнению со стандартной оболочка для интерактивного выполнения программ:

- улучшенная интроспекция
- дополнительные команды
- подсветка кода
- автодополнение

Редакторы

- Расширяемые редакторы Emacs или Vim с множеством различных плагинов
- PyCharm – хорошая IDE для Python
- Sublime Text 3 – популярный расширяемый редактор для тех, кто не осилил Vim :)

PyPi

- Центральный репозиторий модулей для Python
- Позволяет очень просто распространять их
- Подавляющее большинство библиотек устанавливается одной короткой командой
- pip install <pkg_name>

Awesome Python

https://github.com/vinta/awesome-python

Содержание

Официальная документация

- https://docs.python.org/
- Можно выбрать интересующую версию языка (нам 2.7, пожалуйста)
- Практически все детали от базовых до глубоких и редко используемых можно найти здесь или начиная отсюда

Книги

- Марк Пилгрим. Погружение в Python – быстрая вводная в python в виде книги [ru]
- http://www.diveintopython.net/toc/index.html – обновляемая онлайн версия предыдущей книги [en]
- Марк Лутц. Программирование на Python – подробное руководство в виде книги [ru]

Туториалы

- http://learnpythonthehardway.org/book/ – подробный пошаговый онлайн туториал для начинающих [en]
- http://www.swaroopch.com/notes/python/ – онлайн введение в python для начинающих [en]

Продвинутый уровень

- http://docs.python-guide.org/en/latest/ – продвинутое руководство по использованию языка
- https://speakerdeck.com/pyconslides/transforming-code-into-beautiful-idiomatic-python-by-raymond-hettinger-1
  – продвинутый уровень: советы по написанию кода