1 Выбираем интерпретатор

Python 2 против Python 3

При выборе интерпретатора для Python всегда возникает вопрос, что выбрать — Python 2 или Python 3? Ответ не всегда очевиден, однако третья версия с каждым днем становится все более популярной.

Дела обстоят следующим образом:

- долгое время стандартом был Python 2.7;
- в Python 3 произошли значительные изменения в языке (этим немного недовольны некоторые разработчики[1]);
- Python 2.7 будет получать обновления безопасности до 2020 года (https://www.python.org/dev/peps/pep-0373/);
- Python 3 продолжает развиваться, как это происходило с Python 2 в последние годы.

Теперь вы понимаете, почему этот выбор не из простых.

---

[1] Если вы не работаете с сетями на низком уровне, то практически не заметите изменений. Увидите только то, что выражение print стало функцией. В противном случае фраза «немного недовольны» является преуменьшением: разработчики, ответственные за крупные и популярные библиотеки для работы с Интернетом, сокетами или сетями, которые взаимодействуют со строками, представленными в формате Unicode и байтовом формате, должны были (и все еще должны) внести значительные изменения. Подробнее об изменениях, начавшихся, когда Python 3 явился миру, можно узнать из статьи под названием Everything you thought you knew about binary data and Unicode has changed (http://bit.ly/text-vs-data).

Рекомендации

Как нам кажется, опытный разработчик скорее выберет Python 3. Но, если вы можете работать только с Python 2, вы все равно будете считаться пользователем Python. Вот наши рекомендации.

Используйте Python 3, если...

- вам нравится Python 3;
- вы не знаете, какую версию выбрать;
- вы новатор.

Используйте Python 2, если...

- вам нравится Python 2 и вас печалит мысль, что в будущем вас ждет лишь Python 3;
- это повлияет на требования к стабильности вашего приложения[1];
- этого требует программное обеспечение, от которого вы зависите.

То есть... Python 3?

Если вы выбираете интерпретатор для Python и не имеете предубеждений, вам следует использовать новейший Python 3.x (в каждой версии появляются новые и улучшенные модули стандартных библиотек, а также исправления ошибок и пробелов в безопасности). Прогресс не стоит на месте. Выбирайте Python 2, только если на то есть серьезная причина, например вы работаете с библиотекой, которая использует Python 2 и не имеет адекватной альтернативы в Python 3, если вам нужна конкретная реализация (см. раздел «Реализации» далее) или если вам просто нравится работать с Python 2 (как некоторым из нас).

С помощью ресурса Can I Use Python 3? (https://caniusepython3.com/) вы можете проверить, блокируют ли используемые вами проекты возможность работать с Python 3.

Для получения информации обратитесь к Python2orPython3 (http://bit.ly/python2-or-python3), где указываются причины обратной несовместимости в спецификации языков, а также приводятся ссылки на подробные описания различий.

Если вы только начинаете работу с Python, кое о чем вам следует волноваться больше, чем о кросс-совместимости всех версий Python. Просто пишите работающий код для своей системы, а остальное наверстаете позже.

---

[1] Высокоуровневый список изменений вы найдете по адресу http://python3porting.com/stdlib.html в стандартной библиотеке Python (Python's Standard Library).

Реализации

Когда говорят о Python, зачастую имеют в виду не только сам язык, но и реализацию CPython. По сути, Python — это спецификация для языка, которая может быть реализована множеством разных способов.

Разные реализации могут пригодиться для совместимости с другими библиотеками или же для небольшого ускорения. Библиотеки Python должны работать независимо от вашей реализации Python, но работоспособность библиотек, основанных на языке C (например, NumPy), не гарантируется. В этом разделе мы рассмотрим наиболее популярные реализации Python.

В этом руководстве подразумевается, что вы работаете со стандартной реализацией CPython для Python 3, однако мы часто будем делать пометки для Python 2.

CPython

CPython (http://www.python.org/) — это базовая реализация[1] Python, написанная на языке C. Она компилирует код Python в промежуточный байт-код, который затем интерпретируется виртуальной машиной. CPython предоставляет высший уровень совместимости пакетов Python с модулями расширения, написанными на C[2].

Если вы пишете программу с открытым исходным кодом и хотите охватить наиболее широкую аудиторию, обратитесь к CPython. При использовании пакетов, функционирование которых зависит от расширений, написанных на C, CPython — ваш единственный вариант реализации.

Все версии языка Python созданы на языке C, поскольку CPython является базовой реализацией.

Stackless

Stackless Python (https://bitbucket.org/stackless-dev/stackless/wiki/Home) — это обычный вариант CPython (поэтому данная реализация будет работать со всеми библиотеками, которые может использовать CPython). Эта версия языка имеет патч, отвязывающий интерпретатор Python от стека вызовов, что позволяет изменять порядок выполнения кода. Stackless вводит концепцию *тасклетов*, которые могут оборачивать функции и превращать их в «микропотоки» (они могут быть сериа-

[1] Термин «базовая реализация» точно отражает определение языка. Его поведение указывает на то, как должны вести себя другие реализации.

[2] Модули расширения написаны на C, но могут использоваться и в Python.

лизованы на диск для последующего выполнения и планирования, по умолчанию выполняются по методу циклического перебора).

Библиотека greenlet (http://greenlet.readthedocs.org/) реализует такую же функциональность по смене стека для пользователей CPython. Большая ее часть также реализована в PyPy.

PyPy

PyPy (http://pypy.org/) — это интерпретатор Python, реализованный в ограниченном подмножестве статически типизированных языков Python (которое называется RPython), что позволяет выполнить оптимизацию. Интерпретатор предоставляет функциональность компиляции на лету и поддерживает несколько бэкендов, например C, язык CIL (Common Intermediate Language) (http://bit.ly/standard-ecma-335) и байт-код виртуальной машины Java (Java Virtual Machine, JVM).

PyPy нацеливается на максимальную совместимость с базовой реализацией CPython (http://speed.pypy.org/), при этом улучшая производительность. Если вы хотите повысить производительность вашего кода Python, стоит опробовать в деле PyPy. Согласно тестам производительности (бенчмаркам), в данный момент PyPy быстрее CPython более чем в пять раз. Он поддерживает Python 2.7, а PyPy3 (http://pypy.org/compat.html) нацелен на Python 3. Обе версии можно найти на странице загрузки PyPy (http://pypy.org/download.html).

Jython

Jython (http://www.jython.org/) — это реализация интерпретатора Python, компилирующая код Python в байт-код Java, который затем выполняется JVM. Дополнительно он может импортировать и использовать любой класс Java в качестве модуля Python.

Если вам нужно создать интерфейс для существующей базы кода Java (или же у вас есть другие причины писать код Python для JVM), Jython будет лучшим выбором.

Jython в данный момент поддерживает версии Python вплоть до Python 2.7 (http://bit.ly/jython-supports-27).

IronPython

IronPython (http://ironpython.net/) — это реализация Python для фреймворка .NET. Она может использовать библиотеки, написанные как на Python, так и с помощью .NET, а также предоставлять доступ к коду Python другим языкам фреймворка .NET.

Надстройка Python Tools for Visual Studio (http://ironpython.net/tools/) интегрирует IronPython непосредственно в среду разработки Visual Studio, что делает эту реализацию идеальным выбором для разработчиков, использующих Windows.

IronPython поддерживает Python 2.7 (http://ironpython.codeplex.com/releases/view/81726).

PythonNet

Python for .NET (http://pythonnet.github.io/) — это пакет, который предоставляет практически бесшовную интеграцию нативно установленного Python и общеязыковой среды выполнения .NET (Common Language Runtime (CLR)). Такой подход противоположен подходу, которым пользуется IronPython. Это означает, что PythonNet и IronPython дополняют друг друга, а не конкурируют.

В совокупности с Mono (http://www.mono-project.com/) PythonNet позволяет нативным установкам Python в операционных системах, отличающихся от Windows, например OS X и Linux, работать с фреймворком .NET. Реализация может быть без конфликтов запущена вместе с IronPython.

PythonNet поддерживает версии от Python 2.3 до Python 2.7. Инструкции по установке вы найдете на странице http://pythonnet.github.io/readme.html.

Skulpt

Skulpt (http://www.skulpt.org/) — это реализация Python на JavaScript. Для нее еще не была полностью портирована стандартная библиотека CPython. Библиотека включает в себя модули math, random, turtle, image, unittest, а также части модулей time, urllib, DOM и re. Предназначена для использования при обучении. Кроме того, есть возможность добавить собственные модули (http://bit.ly/skulpt-adding-module).

Стоит упомянуть примеры ее применения — Interactive Python (http://interactivepython.org/) и CodeSkulptor (http://www.codeskulptor.org/demos.html).

Skulpt поддерживает большую часть функциональности версий Python 2.7 и Python 3.3. Для получения более подробной информации смотрите страницу реализации Skulpt на GitHub (https://github.com/skulpt/skulpt).

MicroPython

MicroPython (https://micropython.org/) — это реализация Python 3, оптимизированная для запуска на микроконтроллере. Поддерживает 32-битные процессоры ARM, имеющие набор инструкций Thumb v2, такие как Cortex-M, широко используемые в дешевых микроконтроллерах. Включает в себя модули стандартной библиотеки Python (http://bit.ly/micropython-library), а также несколько

библиотек, характерных для MicroPython, которые отвечают за подробную информацию о плате и памяти, доступ к сетям, а также модифицированную версию ctypes, оптимизированную для уменьшения размера. Эта реализация не похожа на Raspberry Pi (https://www.raspberrypi.org/) — та поддерживала Debian или другую операционную систему, основанную на C, на которой установлен Python. Плата pyboard (https://micropython.org/store/#/store) использует MicroPython как свою операционную систему.

Здесь и далее мы будем использовать CPython в Unix-подобной системе, OS X или Windows.

Итак, перейдем к установке.