ЯЗЫКИ ВЫШКИ
№ 3 (03), апрель, 2018
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ОКНА АКАДЕМИЧЕСКОГО РОСТА
ПРИЛОЖЕНИЕ К ИНФОРМАЦИОННОМУ БЮЛЛЕТЕНЮ
МОСКВА • НИЖНИЙ НОВГОРОД • ПЕРМЬ • САНКТ-ПЕТЕРБУРГ

PRINT("NON SCHOLAE SED VITAE DISCIMUS")

PYTHON

В нашем приложении мы рассказываем также и об искусственных языках. В этом выпуске, посвященном активно развивающемуся языку программирования Python, о причинах его изучения, достоинствах и недостатках рассказывают Алексей Зобнин, Борис Орехов, Михаил Густокашин и Андрей Паринов.

Почему вы стали изучать Python?

Алексей Зобнин, доцент базовой кафедры «Яндекса» факультета компьютерных наук

По очень необычной причине. В 2008 году в недавно открывшейся Школе анализа данных «Яндекса» я читал курс по программированию. Мы изучали со студентами язык C++. И вдруг от Ильи Сегаловича, технического директора «Яндекса», поступила просьба посвятить несколько занятий на курсе языку Python. Я до того момента на «Питоне» не программировал и вообще мало что о нем слышал, хотя язык этот довольно старый, он появился в самом начале 1990-х. Я прочитал книжку про «Питон», и мне этот язык очень понравился.

Через год, когда я стал разработчиком в «Яндексе», меня приятно удивило, как широко этот язык распространен там внутри. Помню еще такой случай. Как-то я присутствовал со своим руководителем на одном собеседовании, и мой начальник задал кандидату такой вопрос. Пусть требуется выполнить какие-то действия по обработке или преобразованию текстовых файлов – извлечь информацию, подсчитать статистики, записать в структурированном виде ответ и т.д., причем эти действия не настолько тривиальны, что их можно записать как комбинацию стандартных консольных утилит Unix. Какой язык программирования вы бы стали использовать? Кто-то мог бы ответить «Perl» (да, он был достаточно популярен в то время), кто-то воспользовался бы старым добрым «Паскалем» или тем же C++, но стремительно набиравший тогда популярность «Питон» подходил для этой цели идеально.

Борис Орехов, доцент школы лингвистики факультета гуманитарных наук

Я начал изучать Python благодаря тому, что в моей предметной области – лингвистике и анализе данных – используют именно этот язык. Сейчас на многих языках программирования можно написать программу, которая делает что угодно. Языки в базовой комплектации почти равны по возможностям и выразительности. Выбор определяют дополнительные модули, подключаемые библиотеки, которые облегчают выполнение какой-либо задачи. Python в этом смысле очень заботливый язык: он берет на себя большую часть рутины. Есть даже известная шутка, уже встроенная в язык: если специальным образом написать и выполнить в Python несложный код, то откроется браузер с веселой картинкой, на которой человек летит в небе, а другой удивленно спрашивает его, как ему удается летать. Первый отвечает, что точно не знает, как так вышло, он просто подключил в Python специальный модуль для полетов, а остальное получилось само.

Такие «магические» модули в Python есть и для многих стандартных лингвистических задач, для машинного обучения и многого другого, что постоянно нужно и мне, и нашим студентам-лингвистам. Так что, если ты знаешь Python, ты во многих смыслах становишься как Гарри Поттер. Во-первых, потому, что тебе доступна магия программирования, во-вторых, ты теперь «змееуст», если вспомнить, к какому подотряду чешуйчатых относятся питоны. Хотя название языка не биологического происхождения, оно восходит к британской комик-группе Monty Python, так что Python – это еще и весело.

Когда я начал учить Python, я уже неплохо знал Perl, R, SQL, основы веб-программирования, немного PHP. После свободного стиля, который свойствен всем этим языкам, дисциплина Python показалась мне волнующей и прекрасной. Сразу захотелось переписать на нем все то, что я уже находил в Perl и PHP.

Кроме того, начало занятий Python совпало для меня с тем, что я поступил на работу в Вышку и оказался в среде людей, которые тоже писали на Python. Это важно: когда я работал с Perl, вокруг меня не было единомышленников, и даже довольно приличное Perl-комьюнити в интернете этого не меняет. Когда оказываешься в особенной среде, это сильно стимулирует личностный рост и позволяет быстрее находить верные решения.

Михаил Густокашин,
приглашенный преподаватель департамента больших данных и информационного поиска факультета компьютерных наук

В 2009 году я занимался подготовкой школьников к олимпиадам по информатике и собственно проведением олимпиад: составлением задач, подготовкой тестов и т.п.

Тогда большинство учителей использовало Pascal для обучения программированию (в принципе, это верно и сейчас), но для олимпиадных задач он подходил плохо: подводило отсутствие стандартной библиотеки с удобно реализованными алгоритмами и структурами данных. Для олимпиад больше подходил язык C++, но в нем некоторые даже простые операции выполнялись с помощью довольно монструозных конструкций. Например, чтобы просто пройти по последовательности чисел и напечатать ее, нужно было написать что-то в стиле:

```
for (std::vector<int>::iterator it = v.begin(); it != v.end(); ++it)
        std::cout << *it << std::endl;
```

Такая сложность означала, что есть всего два варианта на ранних этапах обучения программированию: либо просить учеников запомнить это наизусть без понимания, либо долго и нудно рассказывать теорию и только после этого перейти к решению практических задач. Оба способа были неидеальны.

В этот момент я увидел, как просто и изящно можно реализовать полностью аналогичную конструкцию на Python:

```
for elem in v:
        print(elem)
```

А когда я увидел более простой способ:

```
print(*v),
```

то интерес к языку возрос многократно.

В те времена, как и сейчас, было очень много разных языков, в том числе с очень красивыми конструкциями. Почему именно Python? Уже в то время он входил в пятерку самых популярных языков по индексу Tiobe (*https://www.tiobe.com/tiobe-index/*), причем находился там несколько лет. Это давало надежду на то, что язык будет жить, будет поддержка, обновления, библиотеки, документация и сообщество.

В его пригодности для обучения я в тот момент сомневался, но опыт MIT (Massachusetts Institute of Technology), который уже тогда использовал Python как основной язык для преподавания программистских курсов, убедил меня хотя бы попробовать. Язык оказался очень простым, и уже на второй день я достаточно уверенно решал школьные задачи на нем. Более глубокое освоение языка тоже не заняло много времени, но, справедливости ради, это был далеко не первый мой язык.

Возвращаясь к C++ – в какой-то момент после изучения Python мне хотелось забыть его как страшный сон, но стандарт C++11 (2011 года) сделал язык приятнее, в том числе для простых учебных задач. Например, задача по выводу последовательности стала выглядеть так:

```
for (auto elem : v)
        std::cout << elem << std::endl;
```

Почти как на Python, но все равно немного сложнее.

Андрей Паринов, старший преподаватель департамента анализа данных и искусственного интеллекта факультета компьютерных наук

На сегодняшний день Python поддерживает сплоченное сообщество программистов. Существует обширная библиотека классов для вычислений, работы с файловой системой, реляционными базами данных, NoSQL-хранилищами данных, сбором и предобработкой данных. Активно развиваются работающие, как отлаженный механизм, веб-фреймворки.

Мое знакомство с Python началось в 2008 году. Меня заинтересовали алгоритмы кластеризации и классификации, построение рекомендаций, генетические алгоритмы и т.д. Большую часть алгоритмов сейчас рассматривают как часть машинного обучения. На глубоком уровне задачи и алгоритмы я изучил в книгах и работах известных ученых – Бориса Григорьевича Миркина, Сергея Олеговича Кузнецова, Константина Вячеславовича Воронцова.

Стоит ли начинать обучаться программированию именно на Python?

Алексей Зобнин

«Священный» спор о том, каким должен быть первый язык программирования, не прекращается уже десятки лет. У меня первыми языками в середине 90-х были BASIC и Pascal. В середине двухтысячных я вел занятия по информатике в школе, и мы программировали там на PascalABC (тогда еще без .NET). Знаю многих людей, которые предлагают в качестве первого языка изучать JavaScript. Но теперь я за Python! Трудно придумать язык, так похожий на «псевдокод» (что бы мы ни обозначали этим словом). Создатели «Питона» руководствовались «принципом наименьшего удивления»: конструкции языка должны означать именно то, что в основном ожидали бы от них программисты, впервые их увидевшие. Не на последнем месте среди аргументов этого спора и моя любимая фишка языка, объявляющая отступы строк программы обязательной частью синтаксиса. Знаю, насколько тяжело бывает научить школьников расставлять отступы в программе, чтобы обозначить ее структуру. А тут отступы – это необходимая часть языка.

Если говорить о самых маленьких школьниках, то для них в «Питоне» есть прекрасный модуль turtle, реализующий «черепашью графику» из языка «Лого».

Борис Орехов

Дидактический аспект занимает многих теоретиков и профессиональных программистов. Тому, с какого языка стоит начинать обучение, посвящен довольно широкий круг материалов. И многие из них действительно рекомендуют именно Python. Он не только простой, он еще и предоставляет вам максимально короткий путь от первой учебной программы (у программистов это традиционно вывод на экран приветствия мира “Hello World!”; мы, конечно, предлагаем студентам-лингвистам приветствовать мир на башкирском или на исландском: “һаумы, донъя!” или “Sæll, heimur!”) до программы, которая делает что-то полезное. Хотя лично мой путь в программирование был другим и начинался с PHP, в школе у меня не складывалось со многими предметами, которые предполагали решение каких-то общих задач, условия которых оторваны от практического, наблюдаемого. Поэтому я до сих пор не люблю игры: выигрыш в них ни к чему не ведет,

это не созидательный процесс. Для людей моего темперамента Python очень подходит: можно начинать делать что-то полезное уже после первого занятия. А есть специальные учебные языки программирования вроде Logo, которые для такого практико-ориентированного пути не предназначены совсем.

Но есть и альтернативные точки зрения. Так, некоторые молодые программисты из провинции обиженно пишут, что в их регионе трудно найти первую работу, если ты знаешь только Python (в Москве и Санкт-Петербурге с этим проблем, кажется, нет). Но сам Python в этом не виноват.

Михаил Густокашин

Да! Более того, большинству людей, которые не собираются становиться программистами, можно на нем и закончить свое обучение.

Процесс обучения программированию можно, до некоторой степени, сравнивать с процессом обучения работе начальника: вы составляете инструкции, и исполнитель их выполняет. Некоторые языки программирования можно сравнить с требованием писать инструкции на жестком канцелярите, где исполнитель все воспримет максимально формально и превратно истолкует любую описку. С другой стороны, существуют учебные языки, где очень ограничен набор инструкций – этот процесс можно сравнить с руководством очень ограниченным исполнителем, который плохого не сделает, но и хорошего делать умеет мало.

Python достаточно изящно размещается между двумя этими крайностями, позволяя «писать инструкции» почти человеческим языком и в то же время умеренно ограничивая набор возможностей.

Поэтому он не вызывает отвращения на начальном этапе обучения своей сложностью, а на более позднем этапе – своей ограниченностью.

Андрей Паринов

На сегодняшний день Python де-факто является основным языком для анализа данных и машинного обучения. Он поддерживает различные библиотеки для анализа данных. Библиотеки NumPy и SciPy содержат классы и функции для быстрых и точных вычислений. Библиотека ScikitLearn содержит алгоритмы машинного обучения. Библиотека NLTK – функции для работы с текстами. Библиотека PyTorch позволяет обучать нейронные сети с использованием GPU.

Долгое время единственным вариантом для анализа данных было использование дорогих больших программных пакетов, таких как SAS, SPSS и STATA. Несмотря на удобство и высокий уровень качества, стоимость мешала их широкому распространению. Начиная с 80-х годов разрабатывались свободные пакеты для анализа данных. В 2000-х различные инициативы по разработке свободных пакетов для анализа данных были поддержаны Google, Microsoft, Boeing и другими. К 2010 году наиболее функциональными стали платформы на основе языка R и языка Python.

Однако, по версии авторитетного рейтинга TIOBE, Python занимает в большинстве рейтингов 4-е место после Java и C/C++.

На сегодня существует более десятка популярных методологий сравнения. Основные источники для сравнения: запросы в Google, проекты на Github, объявления о поиске сотрудников, научные статьи, авторские методики, опросы программистов. Область сравнений – год, страна, целевая платформа.

Пять наиболее авторитетных рейтингов сходятся во мнении, что Python входит в пятерку самых популярных языков программирования. Лишь рейтинг журнала IEEE Spectrum считает Python самым популярным языком программирования в 2017 году в США.

Все дело в том, что на данный момент уже существует множество промышленных проектов, которые реализованы не на Python. До сих пор C++ и Java занимают лидирующие позиции в рейтингах именно по причине необходимости поддержки кода существующих проектов. Существующие проекты надо поддерживать и развивать. Переписывать их полностью на любом другом языке – слишком дорогое занятие.

Таким образом, при выборе языка программирования стоит обратить внимание на задачи, которые планируется решать с его помощью. На сегодняшний день Python 3 – прекрасный выбор для разработки программ, связанных с анализом данных и машинным обучением.

Каковы сильные и слабые стороны Python в сравнении с другими языками программирования?

Алексей Зобнин

Для того чтобы новый язык стал популярным, недостаточно просто взять и придумать хороший синтаксис и реализовать транслятор. Необходима экосистема: наличие удобных библиотек, удобных средств разработки и, главное, большого сообщества активных пользователей. «Питон» набрал эту критическую массу где-то во второй половине двухтысячных. Он уверенно занимает свою нишу высокоуровневого скриптового языка общего пользования, где акцент сделан на производительности программиста и на читаемости кода. Язык особенно популярен у специалистов по машинному обучению и тех, кто работает с большими данными. Удобная интерактивная оболочка IPython Notebook позволяет прямо в браузере создавать документы с фрагментами кода на «Питоне» и результатами их выполнения с возможностью перезапустить каждый фрагмент независимо. Библиотеки numpy, scipy, matplotlib, pandas и т.д. позволяют обрабатывать и визуализировать научные и статистические данные. Упомяну и широко известные библиотеки для обучения нейронных сетей TensorFlow, Theano, Keras, также имеющие основную реализацию на «Питоне».

Я часто использую систему компьютерной алгебры SageMath. Этот замечательный проект оказался столь успешным еще и потому, что в качестве языка в этой системе был выбран «Питон» (правда, с немного расширенным синтаксисом). Это демонстрирует еще одну сферу применения «Питона» в качестве встраиваемого в другие системы языка программирования или как скриптового языка для расширения возможностей других приложений. Например, на нем можно писать макросы для LibreOffice.

Бессмысленно сравнивать этот язык с занимающими совсем другую нишу C/C++ – я люблю говорить, что эти языки ортогональны друг другу. Например, для больших распределенных серверных приложений «Питон» вряд ли подойдет.

Я в своей работе использую и C++, и «Питон», и у меня часто бывают курьезные машинальные ошибки, когда переключаешься с одного языка на другой. Например, мне надо было уменьшить переменную на единицу, и я написал в «Питоне» --x. А потом удивлялся, почему это программа запустилась и работает, но вовсе не так, как ожидается.

Пожалуй, неудачный момент в синтаксисе «Питона», который часто приводит к ошибкам, – это когда присваиваешь значение какой-то существующей переменной, ошибаешься при этом в ее имени, и значение вместо присваивания старой переменной молча записывается в новую переменную. Полагаю, на эти грабли наступал каждый программист на «Питоне».

Еще один недостаток «Питона», являющийся, на мой взгляд, отражением его достоинств, в том, что язык зачастую не заставляет программиста задумываться об эффективности операций и структур данных. Например, нужно нам вставить элемент в середину списка – пожалуйста, сделаем это через «срезки». А то, что в классической реализации списка эффективнее всего добавлять элементы именно в конец, задумываются не все.

Я оставлю в стороне обсуждение классического «недостатка» «Питона», связанного со скоростью работы программ. Считается, что для прототипирования сложных систем или для разовых вычислений это с лихвой компенсируется скоростью разработки.

Борис Орехов

Тема сильных и слабых сторон Python в профессиональной среде обсуждается давно и подробно. Тут нет секретов. Python ценят за скорость разработки: полезную и полнофункциональную программу на нем можно написать в среднем быстрее, чем на C или Java, но работать она будет в среднем медленнее. Эта «медленность», конечно, будет ощущаться только при обработке по-настоящему больших массивов данных, которые появились в распоряжении лингвистов не так давно. В то же время вопросом остается, что сейчас обходится дороже – машинное время или время программиста? Иными словами, для финального успеха важнее быстро написать программу, которая работает немного медленнее, или долго писать программу, которая потом все сделает быстро? Я считаю, что сейчас правильнее первый подход, и Python ему полностью отвечает, не говоря уже о том, что существуют способы «разогнать» программу на Python до нужных скоростей.

Все остальные особенности Python – дело вкуса (нестрогая типизация, форматирование как часть синтаксиса, свойства реализации встроенных структур данных), и любой программист с опытом в зависимости от качества этого опыта может отнести их как к положительным, так и к отрицательным особенностям языка.

С позиций моей субъективности важно то, что Python оказался хорошо приспособлен для работы именно с текстовыми массивами. Скажем, если я захочу компьютерно обработать текст на русском языке, то я обнаружу, что у Fortran просто нет поддержки современного способа кодировать символы, стандарта Unicode. Этот язык, как и R, предназначен скорее для того, чтобы читать и считать с его помощью уже подготовленные для анализа данные. А Python в этом смысле ничем не ограничен.

Михаил Густокашин

Про простоту изучения я уже написал, это однозначно сильная сторона по сравнению со многими другими языками программирования. Также у Python есть большой набор библиотек на все случаи жизни, сообщество, умеющее отвечать на вопросы, и хорошие учебные курсы. Все это приводит к тому, что «набросать» работающий прототип почти любой программы на Python намного быстрее, чем на большинстве других языков.

К недостаткам можно отнести низкую скорость работы программ на Python; динамическую типизацию переменных, которая может привести к трудно отлавливаемым ошибкам; не очень хорошие возможности по распараллеливанию программ (сейчас число ядер процессора растет намного быстрее скорости отдельного ядра, и это действительно серьезная проблема). В сравнении с каким-либо конкретным языком недостатков можно найти больше, но все они, что радует, начинают проявляться только на больших и сложных проектах, в разработке которых участвуют люди того уровня профессионализма, когда знают уже больше одного языка.

Андрей Паринов

Основной причиной популярности Python стала совокупность нескольких факторов: удобный и понятный синтаксис; слабая типизация; открытость; возможность подключения модулей на языке C.

Совокупность этих обстоятельств привела к возможности быстро разрабатывать программы. Высокая скорость разработки средних по размеру программ – основное преимущество Python. В то же время в крупных промышленных проектах, где над одной и той же системой трудится множество программистов, слабая типизация затрудняет поддержку кода системы, усложняет отладку и предъявляет повышенные требования к уровню разработчиков. Поэтому многие выбирают для проектов язык программирования с сильной типизаций – Java, C++, C#.

Что ждет Python в будущем, на ваш взгляд?

Алексей Зобнин

В ближайшем будущем – завершение неизбежного перехода в индустрии с версий 2.x на версии 3.x. Десять лет назад создатели языка смело выпустили версию 3.0, которая не была обратно совместима со старыми версиями, но в которой, по мнению авторов, устранялись фундаментальные недостатки предыдущих версий. Какое-то время обе ветки версий развивались параллельно. Ветку 3.x сообщество программистов приняло не сразу, ей пришлось в некотором смысле заново завоевывать популярность. Мне тогда казалось, что версии 2.x будут сходиться к числу e, а версии 3.x – к числу π. В итоге ветка 2.x навечно застыла на версии 2.7.14, а вот в третьей ветке число π давно преодолено: текущая стабильная версия имеет номер 3.6.4.

Борис Орехов

Пока что будущее Python выглядит радужно: это совершенно живой язык, на нем пишут код в самых разных предметных областях, в том числе в самых актуальных, и это особенно важно. Большое место в современном мире IT занимают разного рода искусственные нейронные сети, и Python сразу же стал языком, на котором реализованы интерфейсы к библиотекам для таких сетей.

Продолжение читайте на сайте OKNA.HSE.RU

Выпускается НИУ ВШЭ. Редактор Сергей Матвеев. Материалы подготовлены при участии В. Селивёрстова.
Контакты: okna@hse.ru, (495) 772 9590, доб. 12-677.
Подписаться на рассылку можно здесь: www.okna.hse.ru. Дата выпуска: 12.04.2018.