# Python — Lurkmore

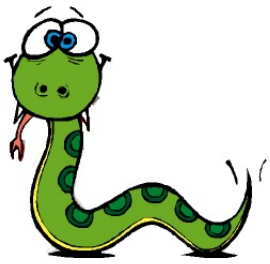
«Прикинь блин, сижу вчера на питон скрипт делаю, Мама подходит и говорит, харошь в крестики нолики играть к экзамену готовься)) »

*— бездна Баша*

«Сижу, разглядываю листинг на Питоне. Отец, глянув издалека на монитор, спрашивает: Маяковского читаешь? »

— *406082*

**Python** (*['paɪθən]*, рус. *Питон*, ласк. *Пихон*, хейт. *Питухон*,*Пыхтон* ,*Пистон*, *Пуфон*, *Гвидопых*, *Гвидобейсик*) — скриптовый язык программирования, созданный неким Гвидо ван Россумом. Назван так, по словам самого Гвидо, в честь шоу Monty Python. Наиболее знаменитая отличительная черта языка — использование отступов для выделения блоков кода и управляющих структур.



Удав же!

## Описание

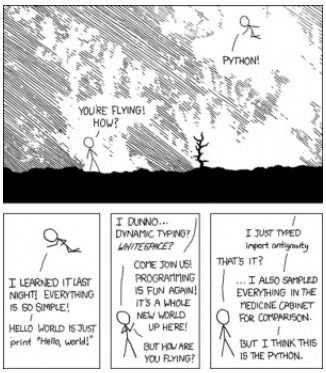
Автор языка — до недавних пор сотрудник Dropbox, а теперь — Microsoft, прежде ставшей спонсором некоторых конференций по Python). Поддерживает идеи SJW[1].

Чтобы его изучить, достаточно ознакомиться с туториалом(RTFM BITCH) и полистать код пары-тройки опенсорсных программ. Язык крайне простой и скромный на выразительные средства сравнительно с Ruby или Perl (один из элементов идеологии — надо программы писать, а не самовыражаться затейливо). Но на самом деле предоставляемые возможности достаточно широки, чтобы надолго увлечься данным языком. При достаточном желании и отсутствии перманентного отрицания (в духе «всё хуйня кроме пчёл») есть вероятность научиться писать годный, сопровождабельный и продаваемый код. В силу своей примитивности и нулевого порога входа, крайне популярен среди школия и быдлокодеров.

Основной причиной вышеупомянутого отрицания обычно бывает «магия отступов»: в отличие от многих языков, в которых *можно*, но *не обязательно* индентить код, в питоне написать код лопатой невозможно *в принципе* — отступы прямо влияют на вложенность выражений. Сторонники считают, что это удобнее, чем фигурные скобки (не говоря уже о begin…end), а заодно положительно влияет на читаемость кода, дисциплинирует программистов и отбрасывает кодирующих обезьян. Противники обзывают его «пиздоном» и пилят код на PHP.

Вызывает ненависть многих ЯП-нёрдов, ибо лишь ограниченно поддерживает некоторые продвинутые технологии программирования:

- анонимные функции (лямбды) могут состоять только из одного выражения;
- «reduce()» (аналог foldl из Haskell) убран в закрома стандартной библиотеки и не рекомендуется к применению из-за «плохой читаемости кода»;
- отсутствует Pattern Matching: всё-таки, это императивный язык;
- Гвидо считает ненужной поддержку хвостовой рекурсии, ибо уже есть циклы;
- невозможно потоками параллелить программу по нескольким процессорам/ядрам: вместо потоков надо использовать модуль multiprocessing или другие интерпретаторы питона (например, PyPy-STM).
- многострочные лямбды могли бы с извращениями помочь выправить положение, но их тоже нет




Я попробовал разные препараты из фармшкафа, но думаю, что это питон. (xkcd)

Бесит взрослое поколение, ибо невозбранно позволяет оттяпать себе пальцы, пройтись по граблям или прострелить ногу:

- объявление переменной при первом присваивании ей значения: если опечатался и создал не то — радуйся спиду;
- глобальное состояние: даже можно менять поля в объектах снаружи (коварный питон вначале бездействует);
- ссылки на функции: он не мешает создать переменную «len» и подорваться при попытке вызвать «len()»;
- отступы согласно стандарту PEP8 (см. ниже) должны быть в 4 пробела — а питон, даже третий, принимает любое число, и обязательно найдётся уебан, который смешает табы с пробелами.

Алсо, синтаксис с отступами и строчными буквами ни разу не совместим с традиционными шрифтами для программистов (такими, как Terminus), но любивен на «Маках», где используются сжатые шрифты (такие, как Lucida Grande). Отсюда мораль: сильнее сжимайте шрифты.

Питон прост. Он очень прост. Он феерически прост. Например, поиск простых чисел:

```
def f(x):
    for y in xrange(2, x):
        if not x % y:
            return False
    return True

U = filter(f, xrange(2, 40))
```

И версия для настоящих ЯП военов:

```
# Я могу найти все простые числа от 0 до x. А ты?
U = lambda x: filter(lambda y: all(map(lambda z: y % z, range(2, 1 + int(y ** 0.5)))), range(2, x))
```

Православная версия 3-го уровня, без всяких задрот-лямбд:

```
#скорость в питоне не главное ;), хотя дядя Лутц говорит, что генераторы быстрее в данном случае
def optimus_prime(val):
    return [x for x in range(2,val+1) if all((x%y for y in range(2,1+int(x**0.5))))]
```

И очень забавные бегущие цифры на вашем мониторе:

```
import time # тут библиотека времени
import math # а вот это считает корни и прочий матан

with open('outfile.txt', 'w') as out: # ясно же, что с файлом работать будем
    out.write('Первый нах посчитал от 1 and 10 000\n-----\n А чего добился ты ?\n')
    start = time.clock()
    print("2\n3\n5\n7\n11\n13\n17\n19\n23\n")
    out.write( "2\n3\n5\n7\n11\n13\n17\n19\n23\n") ## Это выкидывает перечисленные числа сразу, это круто, экономит нам 0.04 секунды при расчёте до 1000000
    for i in range(29, 1000000, 2):
        if (
            i % 2 ==  0 or # |(                                                  )
            i % 3 ==  0 or # |(                                                  )
            i % 5 ==  0 or # |(                                                  )
            i % 7 ==  0 or # |(   быстренько проверим на первые простые,        )
            i % 11 == 0 or # |(   массив можно продолжать до усрачки            )
            i % 13 == 0 or # |(                                                  )
            i % 17 == 0 or # |(                                                  )
            i % 19 == 0 or # |(                                                  )
```

```
            i % 23 == 0    # |(                                                    )
        ):
            continue
        # если реально похоже на простое, подозрительно щуримся и проверяем его решетом ->
        else:
            for j in range(29, int(math.sqrt(i) + 1), 2): # нафиг все чётные, да и считать можно с 29
                if i % j == 0:
                    break
                else:
                    print(i)
                    out.write(str(i) + "\n")

finish = time.clock()
print(finish - start)
out.write('Program work at ' + str(finish - start) + ' second') # тут запишем время выполнения
```

Шах и мат.

## Python 3 Fail

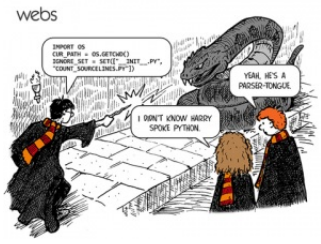
**Python 3** = новая версия того же языка от того же автора. Исправляет некоторые недочёты предыдущей версии, например:

- любые попытки сконвертировать байты в юникод и обратно без указания кодировки будут вызывать исключение, да и вообще работа с юникодом стала намного удобнее;
- ненавистный всем любителям многопоточности GIL (Global Interpreter Lock) остался, но в версии 3.2 его малехо подфиксили, так что теперь он работает адекватнее, хотя склонность давать зеленый свет жирным долго работающим процессам в ущерб легковесным все еще осталась;
- появилась возможность указывать хинты для типов аргументов функций, а с версии 3.6 - для обычных переменных и аттрибутов класса;
- улучшена библиотека коллекций. Теперь питон настолько *гибкий*, что из рук выскальзывает!
- строковые шаблоны используют модный синтаксис в стиле C# вместо старого C-подобного синтаксиса;
- добавлен новый слой I/O, позволяющий работать с файлами так же, как в Java и C#;
- прямо в стандартную либу запилили фреймворк asyncio, так что теперь можно пилить асинхронный код, не притягивая в проект таких гигантов, как Twisted или Tornado;
- и просто все патчи делаются для тройки, и только треть бэкпортится на двойку (можно спокойно наткнуться на баг 2007 года, который на 3 запатчен, а на 2 — не нужно).

Однако лулз в том, что хотя Python3 отрелизили в 2008 году, многие продолжают лабать на ветке 2.*. А суть тут в нарушении обратной совместимости между 2.* и 3.*, особенно из-за строк. Под 3.* надо писать кучу новых либ, или переделывать старые и танцевать с бубном. А под 2.* всё уже и так есть, и по сравнению с этим геморроем по переносу всех наработок сообщества профит от фикса нескольких костылей старой версии — неочевиден. На самом деле, переход между версиями не так уж сложен, большинство старых библиотек уже успешно перепилены под 3-ку, а многие новые только под нее и выходят. Вишенкой на торте и последним гвоздем в гроб 2-ой ветке является официальное заявление разработчиков о завершении поддержки последней в 2020 году. Goodnight, sweet prince. Так что новые проекты (в смысле, крупные, а не админские скрипты) на 2-ке никто не пишет, а старые на тройку не переводят разве что ленивые, да еще ребята, у которых все завязано на Twisted или какого-нибудь похожего монстра, не имеющего полной совместимости с Python3.

Кроме того, будучи простым в освоении интерпретатором, Python стал де-факто скриптовым языком во многих бизнес-приложениях, потеснив кастрированные версии BASIC и самопальные «скриптовые языки». Python вкалывает как Папа Карло и в CVS, и в Blender, и в Techlog, тихой сапой крадётся в Open/LibreOffice, и даже обложил со всех сторон приложения Microsoft. Однако, не у всех разработчиков хватает сотен нефти и смелости поменять в своём приложении милый сердцу 2.7 на троечку.

## Другие фичи



Гарька тоже шпрекает

- Любовь питонистов к подчеркиваниям вылилась в использование последних в названиях многих переменных: *_защищённых*, *__приватных*, *__магических__* и *просто_плохо_названных_очень_длинных_потому_что_так_получилось_*.
- PEP-0008 (рус): документ, описывающий стилистические особенности кода на питоне, которые должен соблюдать любой программист, чтобы не быть анально покаранным. Для проверки кода на соответствие есть специальная утилита.
- Dive into Python: универсальная книга, позволяющая любому выучить питон и получать такую зарплату, шо все бабы будут теч.
- Будучи скриптовым языком, Питон медлителен. Для борьбы с этим используются: специальный транслятор для написания шустрых модулей к питону Cython, JIT-компилятор PyPy и прочее, тысячи их. Да и CPython на месте не стоит.
- На самом деле, CPython является виртуальной машиной со встроенным интерпретатором ЯП Python. Никто не мешает делать компиляторы других языков в байткод питона, хоть это и не нужно. Надо сказать, что родной компилятор CPython'а туп до невозможности и не выполняет практически никаких оптимизаций кода. Отличный пример: если присвоить некой локальной переменной метод объекта, а потом начать вызывать его через эту переменную, то Python ВНЕЗАПНО заработает шустрее, чем при явном вызове метода. А разгадка проста: разыменование ссылки на метод, лежащей в объекте, обходится дороже, чем дерганье локальной переменной. Случай не единичный, но всем насрать.
- Питон-фаги любят описывать свой стиль написания программ как Pythonic и говорить, что они следуют Python Way. Некоторые кодеры реагируют на это мемом «Python way? No way!».
- Python, наряду с C++, Java и Go — один из языков, принятых к использованию в Google. В последнее время, впрочем, на нем ничего нового не пишут, а то, что есть, постепенно перепиливается на жабу. А все из-за того, что Гвидо в свое время разосрался с руководством и свалил в Dropbox.

**]**

## def __cult__(self, self, self, self, self, self, self, cls):

Существует культ Питона, называемый «Дзеном Питона» (The Zen of Python). Основные постулаты:

> Красивое лучше, чем уродливое.
>
> Явное лучше, чем неявное. Простое лучше, чем сложное. Сложное лучше, чем запутанное. Плоское лучше, чем вложенное. Разреженное лучше, чем плотное. Читаемость имеет значение. Особые случаи не настолько особые, чтобы нарушать правила. При этом практичность важнее безупречности. Ошибки никогда не должны замалчиваться. Если не замалчиваются явно. Встретив двусмысленность, отбрось искушение угадать. Должен существовать один — и, желательно, только один — очевидный способ сделать это. Хотя он поначалу может быть и не очевиден, если вы не голландец.[1] Сейчас лучше, чем никогда. Хотя никогда зачастую лучше, чем прямо сейчас. Если реализацию сложно объяснить — идея плоха. Если реализацию легко объяснить — идея, возможно, хороша. Пространства имён — отличная штука! Будем делать их побольше.

BTW, эти строки можно увидеть, введя в интерпретаторе «import this»


Табы лучше, чем пробелы

## def compare(Perl):

Как видно из философии, Питон является практически полной антитезой Перлу, утверждая, что делает нормально все то, что в Перле сделано через жопу. В результате, питонисты смотрят на верблюжьих как на говно, а за упоминание Питона на перловском форуме можно схватить бан.

## import sjw

this.py

Python, начиная с версии 3.8 избавится от слов slave и master. Это произошло благодаря известному еврейскому педерасту Витьке Штайнеру. Он создал в баг-трекере тикет с просьбой выпилить слова, которые напоминали нигерам о наболевшем. Волевым решением тогда ещё пожизненного диктатора Гвидо Ван Россума это идиотское требование было утверждено.

## См. также

- Perl
- Копипаста:Python
- Русский фан-клуб Гвидо ван Россума
- Вариант реализации Python на хостинге
- EVE Online сделана на Python

## Примечания

Шаблон:Reflist


Языки программирования

++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер Языки программирования

Животные

12 oz. Mouse Advice Dog Amazing Horse Anacondaz Angry Birds Bad Taxidermy Badger Battletoads BonziBuddy Charlie the Unicorn Crazy Frog Doge Dopefish Dramatic Prairie Dog Duckroll Earthworm Jim EDonkey2000 Everyone else has had more sex than me Fluffy Frog FUCK YEAH SEAKING Giant Enemy Crab Happy Tree Friends Ika Musume Internet Explorer Lolifox Mr. Hands Mudkip My Little Pony Nomad Frog O RLY? Pepe the Frog Python Renard Queenston Рыбка Дебилариус Sheep.exe Sonic the Hedgehog Surfin' Bird Winged Doom Worms Ёж Ёж ненависти Ёжик в тумане А то! Абаснуй Анаконда Аэрофлотовская курица Бабруйск Бармаглот Белая акула Белочка Битва слона с китом Бобёр-извращенец Боброудав Боевые животные Боклан Большой Пиздец Борьба Бобра с Ослом Бэтмен В мире животных Веганы Винни-Пух Волк Волшебный кролик Вонни Вуглускр Гаечка Газетная утка Гипножаба ГМО Гоблин Годзилла Горящий медведь Гринпис Гуидак Дельфин День Йожа Динозавры Дойная корова Донки-хот Дракон Драконофаги Ебала жаба гадюку Еби гусей Жаба Жаба душит Жук-антисемит Журнал «Крокодил» Жывотнайе Зайчатки разума Заяц Заяц и медведь Заяц несудьбы Заяц ПЦ Зелёный слоник Зомби Зоофилия И животноводство! Инерциальная гомойотермия ЙАААААЗЬ!

w:Python en.w:Python (programming language) ae:Python

1. ↑ Намёк на национальность создателя языка, а не на то, что вы подумали.