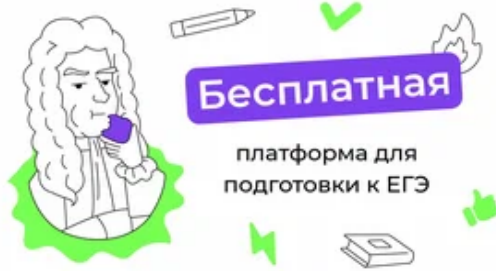



ХОЧУ ПОМОЧЬ  
ПРОЕКТУ



Бесплатная  
платформа для  
подготовки к ЕГЭ



РЕКЛАМА

Бесплатная платформа для подготовки к ЕГЭ

Бесплатно. Более 23 000 отборных заданий формата ЕГЭ с подробными пояснениями.

Узнать больше

practicum.yandex.ru

Курс «1С-аналитик»  
для начинающих.

5,0 ★ Рейтинг организации ⓘ

Научитесь собирать требования к ПО и моделировать бизнес-процессы.

О курсе >

Как учим >

Программа >

Узнать больше

установка Python3 / PyPy3, полноценная замена классического Python3

д работал быстрее,  
пользовать PyPy"Гвидо ван Россум (создатель Python).

ая замена CPython. Он построен с использованием языка RPython, который был разработан  
а использовать его вместо CPython - скорость: обычно он работает быстрее.

о при этом поддерживать (в идеале) любую программу Python.

к PyPy 7.3, реализует три максимальные версии - это Python 3.8 и Python 3.9. Он  
е CPython (с небольшими изменениями) и проходит набор тестов Python. Выпуск PyPy 7.3.9  
емых модулей стандартной библиотеки Python. Известные различия с CPython [смотрите в](#)

- интерпретатор PyPy3.8 поддерживает синтаксис и функции Python 3.8, включая стандартную библиотеку для CPython 3.8.
- интерпретатор PyPy3.9 поддерживает синтаксис и функции Python 3.9, включая стандартную библиотеку для CPython 3.9.

Интерпретаторы основаны на одной и той же кодовой базе, поэтому они имеют несколько выпусков. Все API совместимы с другими выпусками PyPy7.3. Основные моменты, начиная с выпуска 7.3.8 включают:

- Исправлены некоторые неудачные тесты стандартной библиотеки на PyPy3.9.
- Обновлен связанная системная библиотека libexpat до 2.4.6 и [sqlite3](#) до версии 3.38.2.

Поддерживаются и обслуживаются следующие архитектуры ЦП:

- x86 (IA-32) и x86\_64,
- платформа ARM (ARMv6 or ARMv7, with VFPv3),
- AArch64,
- PowerPC 64bit как с прямым, так и с обратным порядком байтов,
- System Z (s390x),

## Скорость исполнения кода компилятором PyPy.

Основной исполняемый файл руру поставляется с компилятором Just-in-Time. Он действительно быстро запускает большинство тестов, включая очень большие и сложные приложения Python, а не только 10-строчные.

Два случая, когда PyPy не сможет ускорить код:

- **Кратковременные процессы:** если PyPy запускается со скриптами работающими меньше 2-х секунд, JIT-компилятору не хватит времени для разгона.
- **JIT-компилятор не поможет,** если все время исполнения программы тратится в подключаемых C-библиотеках, а не на выполнение кода, написанного на Python.

Таким образом, лучше всего PyPy работает при выполнении длительно выполняющихся программ, когда значительная часть времени тратится на выполнение кода Python. Это случай, охватываемый большинством проводимых тестов.

## Установка PyPy3 на ОС Windows:

Установка PyPy3 ни чем не отличается от установки классического Python3. Загрузить исходники PyPy3 для ОС Windows можно с [официальной страницы](#). Дистрибутив PyPy3 Windows 32 bit совместим с любыми 32- или 64-битными ОС Windows.


Вверх



```
$ source env-pypy/bin/activate
# (PYPY3) $
```

Обратите внимание, что env-pypy/bin/python теперь является символической ссылкой на env-pypy/bin/pypy, следовательно можно запускать pypy, просто набрав python.

РЕКЛАМА · 16+ · Я



Практикум

Курс «1С-аналитик»

Работает на технологиях Яндекс

practicum.yandex.ru

Курс «1С-аналитик» для начинающих.

5,0 ★ Рейтинг организации ⓘ

Научитесь собирать требования к ПО и моделировать бизнес-процессы.

О курсе >

Как учим >

Программа >

Узнать больше

В виртуальную среду, все равно необходимо обновить pip и wheel до последних версий через:

```
python3 -m pip install --upgrade pip wheel
```

## Совместимость с классическим Python3.

Несмотря на то, что PyPy3 имеет несколько отличий в управлении временем жизни объекта. Модули, использующие CPython C API, могут не достигнут ускорения за счет JIT. Авторам библиотек, команда разработчиков PyPy.

При использовании PyPy3 с научной экосистемой Python, то рекомендуется использовать компилятор Python conda, библиотеки, такие как scikit-learn и SciPy для PyPy.

Многие модули Python перекомпилированы для PyPy, чтобы они работали. В зависимости от системы сборки они могут быть более сложными.

Стандартные библиотечные модули. Обратите внимание, что многие модули python3 реализованы на C, поэтому они точно будут работать. Просто нужно проверить, сможет ли PyPy3 на вашей системе.

Многие модули Python перекомпилированы для PyPy, чтобы они работали. В зависимости от системы сборки они могут быть более сложными. Стандартные библиотечные модули. Обратите внимание, что многие модули python3 реализованы на C, поэтому они точно будут работать. Просто нужно проверить, сможет ли PyPy3 на вашей системе. select, signal, symbol, sys, termios, thread, time, token, unicodedata, zipimport, zlib.

Если PyPy3 импортирует вышеуказанные модули без ошибок, то он полностью совместим с вашим Python3 и должен работать без каких-либо ошибок.

Поддерживается и написано на чистом Python: cPickle, ctypes, datetime, dbm, \_functools, grp, readline, resource, sqlite3, syslog.

Все сторонние модули, которые написаны на чистом python в CPython, конечно будут работать после успешной установки.

## Различия, связанные со стратегиями сбора мусора.

Сборщики мусора, используемые или реализованные PyPy3, не основаны на подсчете ссылок, поэтому объекты не освобождаются мгновенно, когда они больше недоступны. Наиболее очевидный эффект от этого заключается в том, что файлы (и сокеты и т. д.) не закрываются сразу после выхода за пределы области видимости. Это отличие от классического Python3, не будет изменяться командой разработчиков.

Следующий код заполнит файл не сразу, а только через определенный промежуток времени, когда GC выполнит сборку мусора и очистит вывод:

```
open("filename", "w").write("stuff")
```

Правильное использование заключается в следующем:

```
with open("filename", "w") as f:
    f.write("stuff")
```

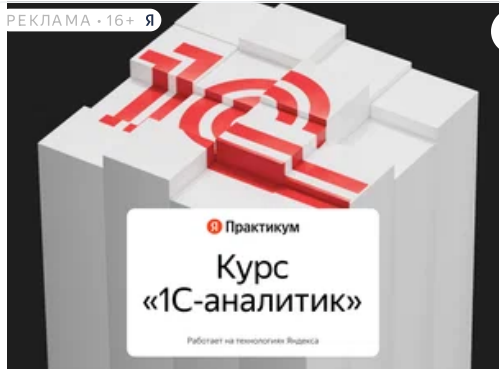
Та же проблема - не закрытие файлов - может также возникнуть, если программа открывает большое количество файлов, не закрывая их явно. В этом случае можно легко достичь системного ограничения на количество файловых дескрипторов, которые разрешено открывать одновременно.

PyPy3 можно запустить с параметром командной строки -X track-resources (например, pypy -X track-resources myprogram.py). Это вызывает ResourceWarning, когда GC закрывает незакрытый файл или сокет. Также дается трассировка для места, где был открыт файл или сокет, что помогает найти места, где отсутствует метод .close().

Точно так же помните, что необходимо закрывать неизрасходованный генератор, чтобы ожидающие завершения операторы finally или with выполнялись немедленно:

Вверх

```
def mygen():
    with foo:
        yield 42
```



practicum.yandex.ru

## Курс «1С-аналитик» для начинающих.

5,0 ★ Рейтинг организации ⓘ

Научитесь собирать требования к ПО и моделировать бизнес-процессы.

О курсе

Как учим

Программа

Узнать больше

ется не сразу!

и образом

`del()` не выполняются так же предсказуемо, как на CPython: в PyPy3 они запускаются *через* тся вообще, если программа тем временем завершает работу).

возвращает неиспользуемую память операционной системе, если есть системный вызов `madvise()` (BSD) или в Windows. Важно понимать, что можно не увидеть этого в выводе команды терминала

помечаются `MADV_FREE`, который сообщает системе: *Если вам понадобится больше памяти в раницу*. Пока памяти достаточно, столбец `RES` вверху может оставаться высоким. Исключением ы без `MADV_FREE`, где PyPy3 использует `MADV_DONTNEED`, который принудительно снижает `RES`

## Почему PyPy3 жрет так много памяти?

PyPy3 возвращает неиспользуемую память операционной системе только после системного вызова `madvise()` (по крайней мере, в Linux, OS X, BSD) или в Windows. Важно понимать, что такое поведение может HE показываться топе утилиты `bash htop`. Неиспользуемые страницы помечаются `MADV_FREE`, что говорит системе: *"Если в какой-то момент понадобится больше памяти, то возьмите эту страницу"*. Пока памяти много, верхний столбец `RES` остается высоким.

Исключением из этого правила являются системы без `MADV_FREE`, где PyPy3 использует `MADV_DONTNEED`, что принудительно снижает `RES`. Это включает Linux = 4.4.

## Подклассы встроенных типов

Официально в CPython вообще нет правила, когда точно переопределенный метод подклассов встроенных типов вызывается неявно или нет. В качестве приближения эти методы никогда не вызываются другими встроенными методами того же объекта. Например, переопределенный `__getitem__()` в подклассе `dict` не будет вызываться, например, встроенный метод `dict.get()`.

Сказанное выше верно как для CPython, так и для PyPy. Могут возникнуть различия в том, будет ли встроенная функция или метод вызывать переопределенный метод другого объекта, кроме `self`. В PyPy они часто вызываются в тех случаях, когда CPython этого не делает.

Два примера:

```
class D(dict):
    def __getitem__(self, key):
        return "%r from D" % (key,)

class A(object):
    pass

a = A()
a.__dict__ = D()
a.foo = "a's own foo"
print(a.foo)
# CPython => a's own foo
# PyPy => 'foo' from D

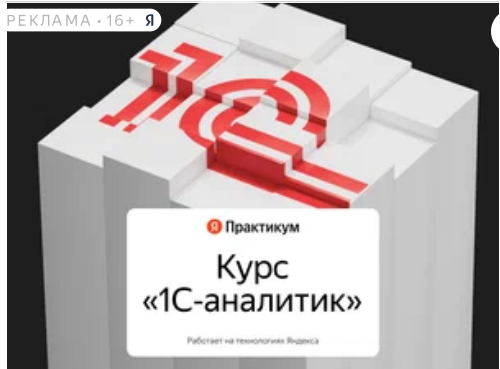
glob = D(foo="base item")
loc = D(foo="local item")
exec("print foo" in glob, loc)
```

Вверх



```
# CPython => base item
# PyPy => 'foo' from D
```

## Игнорируемые исключения.



practicum.yandex.ru

**Курс «1С-аналитик» для начинающих.**

5,0 ★ Рейтинг организации ⓘ

Научитесь собирать требования к ПО и моделировать бизнес-процессы.

О курсе

Как учим

Программа

Узнать больше

молча проглатывать исключения. Точный список случаев, когда это происходит, довольно з очень редки. Наиболее известные места - это настраиваемые расширенные методы сравнения зарю; вызовы некоторых встроенных функций, таких как `isinstance()`.

смотрено конструкцией и не задокументировано как таковое (например, для `hasattr()`), в однимать исключения.

### К ТИПОВ (is и id).

К типов работает по равенству значений, а не по идентичности id. Это означает, что `x + 1` ольных целых чисел `x`. Правило распространяется на следующие встроенные типы:

```
имвольные строки)`;
дносимвольные строки)`;
)`;
zenset)`.
```

горых изменений в `id`. `id` выполняет следующее условие: `x is y <=> id(x) == id(y)`. Поэтому возвращать значение, которое вычисляется из аргумента, и, таким образом, может быть кет быть произвольно длинным).

Обратите внимание, что `строки` длиной 2 или более могут быть равны, не будучи идентичными. Аналогично, `x is (2,)` не обязательно истинно, даже если `x` содержит кортеж и `x == (2,)`. Правила уникальности применимы только к частным случаям, описанным выше. Правила `str`, `unicode`, `tuple` и `frozenset` были добавлены в PyPy выпуск 5.4; до этого тест типа `if x is "?"` или `if x is ()` мог потерпеть неудачу, даже если `x` был равен `"?"` или `()`. Новое поведение, добавленное в PyPy выпуск 5.4, ближе к CPython, который кэширует именно пустой `tuple/frozenset` и (как правило, но не всегда) `str` и `unicode` длиной `<= 1`.

Обратите внимание, что для `float` существует только один объект на “битовый шаблон” `float`. Таким образом, `float('nan') is float('nan')` истинно на PyPy3, но не на CPython, потому что они являются двумя объектами; но `0.0 is -0.0` всегда `False`, так как битовые шаблоны различны. Как обычно, `float('nan') == float('nan')` всегда ложно. При использовании в контейнерах (например, в виде элементов `list` или в `set`) точное правило равенства используется так: “`if x is y or x == y`” (как на CPython, так и на PyPy); как следствие, поскольку все `nan` идентичны в PyPy3, вы не можете иметь несколько из них в множестве `set`, в отличие от CPython.

Другим следствием является то, что `cmp(float('nan'), float('nan')) == 0`, потому что `cmp()` сначала проверяет `is`, идентичны ли аргументы (нет хорошего значения для возврата из этого `cmp()`, так как функция `cmp()` делает вид, что существует полный порядок для чисел с плавающей запятой, но это неверно для `NaN`).

## Различия в производительности.

CPython имеет оптимизацию, которая может сделать повторную конкатенацию строк неквадратичной. Например, такой код выполняется за время `O(n)`:

```
s = ''
for string in mylist:
    s += string
```

В PyPy3 этот код всегда будет иметь квадратичную сложность. Также обратите внимание, что оптимизация CPython хрупкая и в любом случае может сломаться из-за небольших изменений в коде. Так что все равно необходимо заменить код выше на:

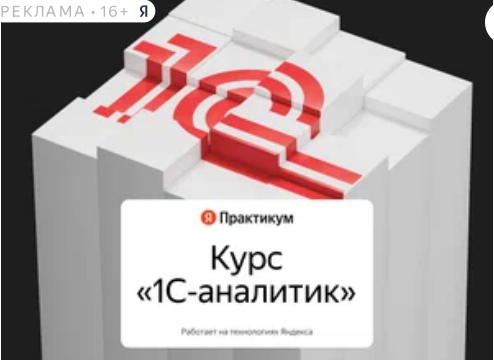
```
parts = []
for string in mylist:
    parts.append(string)
s = "".join(parts)
```


В принципе это основные отличия с которыми сталкивается 80% разработчиков.

Вверх



Содержание раздела:

- [ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА](#)
- [Выбираем разрядность Python3 для Windows](#)
- [Установка Python3 на ОС Windows](#)



 practicum.yandex.ru

**Курс «1С-аналитик»**  
**DOCS-Python.ru™, 2023 г.**  
**для начинающих.**

5,0  Рейтинг организации 

Научитесь собирать требования к ПО и моделировать бизнес-процессы.

- О курсе >
- Как учим >
- Программа >

[Узнать больше](#)

[ан\) из репозиториев](#)  
[Linux](#)  
[з на ОС Linux](#)  
[тора Python](#)  
[совместимость с Python3](#)  
[версию Python](#)

(Внимание! При копировании материала ссылка на источник обязательна) [@docs\\_python\\_ru](#)