

Содержание

Введение	9
Принятые условные обозначения	10
От издательства	12
Запуск программ на Python из командной строки	13
Параметры командной строки в Python	13
Указание программ в командной строке	15
Параметры командной строки в версии Python 2.X	17
Переменные окружения Python	18
Операционные переменные	18
Переменные, аналоги параметров командной строки в Python	20
Запуск программ на Python в Windows	21
Директивы запуска файлов	22
Командные строки для запуска	22
Переменные окружения для запуска	23
Встроенные типы и операторы	24
Операторы и их предшествование	24
Примечания к применению операторов	26
Категории операций	28
Конкретные встроенные типы	34
Числа	34
Символьные строки	37
Символьные строки в уникоде	59
Списки	64
Словари	72
Кортежи	77
Файлы	78
Множества	85
Другие типы и преобразования	88
Операторы и синтаксис	90
Правила синтаксиса	90
Правила именования	92

Конкретные операторы	95
Оператор присваивания	96
Оператор выражения	100
Оператор <code>print</code>	102
Условный оператор <code>if</code>	105
Оператор цикла <code>while</code>	106
Оператор цикла <code>for</code>	106
Оператор <code>pass</code>	107
Оператор <code>break</code>	107
Оператор <code>continue</code>	107
Оператор <code>del</code>	108
Оператор <code>def</code>	108
Оператор <code>return</code>	113
Оператор <code>yield</code>	114
Оператор <code>global</code>	116
Оператор <code>nonlocal</code>	116
Оператор <code>import</code>	117
Оператор <code>from</code>	121
Оператор <code>class</code>	123
Оператор <code>try</code>	126
Оператор <code>raise</code>	129
Оператор <code>assert</code>	132
Оператор <code>with</code>	132
Операторы в версии Python 2.X	134
Правила обозначения пространств имен и областей действия	135
Уточненные имена: пространства имен объектов	135
Неуточненные имена: лексические области действия	136
Вложенные области действия и замыкания	138
Объектно-ориентированное программирование	140
Классы и экземпляры	140
Псевдозакрытые атрибуты	141
Классы нового стиля	142
Формальные правила наследования	143
Методы перегрузки операторов	149
Методы для всех видов операций	150
Методы для операций над коллекциями (последовательностями и отображениями)	158

Методы для числовых операций в двоичной форме	160
Методы для других операций над числами	164
Методы для операций с дескрипторами	165
Методы для операций с диспетчерами контекста	166
Методы перегрузки операторов в версии Python 2.X	167
Встроенные функции	171
Встроенные функции в версии Python 2.X	201
Встроенные исключения	209
Суперклассы категорий исключений	210
Конкретные исключения	212
Конкретные исключения типа <code>OSError</code>	217
Исключения категории предупреждений	219
Каркас предупреждений	220
Встроенные исключения в версии Python 3.2	221
Встроенные исключения в версии Python 2.X	222
Встроенные атрибуты	223
Стандартные библиотечные модули	224
Модуль <code>sys</code>	225
Модуль <code>string</code>	237
Функции и классы	237
Константы	238
Модуль <code>os</code>	239
Административные средства	241
Константы переносимости	242
Средства командной оболочки	243
Средства среды исполнения	245
Средства дескрипторов файлов	247
Средства имен путей к файлам	251
Управление процессами	256
Модуль <code>os.path</code>	260
Модуль <code>re</code> сопоставления по шаблону	263
Функции из модуля <code>re</code>	263
Шаблонные объекты регулярных выражений	266
Совпадающие объекты	267
Синтаксис шаблонов	269

Модули сохраняемости объектов	272
Модули <code>shelve</code> и <code>dbm</code>	273
Модуль <code>pickle</code>	276
Модуль <code>tkinter</code> для построения ГПИ	280
Пример применения модуля <code>tkinter</code>	280
Базовые виджеты в модуле <code>tkinter</code>	281
Типичные средства создания диалоговых окон	282
Дополнительные классы и средства в модуле <code>tkinter</code>	283
Сопоставление модуля <code>tkinter</code> с библиотекой Tk на языке Tcl	284
Модули и средства доступа к Интернету	285
Другие стандартные библиотечные модули	288
Модуль <code>math</code>	289
Модуль <code>time</code>	289
Модуль <code>timeit</code>	291
Модуль <code>datetime</code>	293
Модуль <code>random</code>	293
Модуль <code>json</code>	294
Модуль <code>subprocess</code>	294
Модуль <code>enum</code>	295
Модуль <code>struct</code>	296
Модули многопоточной обработки	297
Прикладной интерфейс API базы данных SQL в Python	299
Пример применения прикладного интерфейса API базы данных SQL	300
Интерфейсный модуль	301
Объекты подключения к базе данных	301
Объекты курсоров	302
Объекты типов и конструкторы	304
Дополнительные рекомендации и идиомы	304
Общие рекомендации по языку	304
Рекомендации по среде исполнения	306
Рекомендации по применению	308
Разные рекомендации	311
Предметный указатель	313

Карманный справочник по Python

Введение

Python — это язык общего назначения с открытым исходным кодом, применением множества парадигм, поддержкой структур объектно-ориентированного, функционального и процедурного программирования. Как правило, он применяется для написания как автономных программ, так и сценариев в самых разных областях и обычно считается одним из самых широко употребляемых языков программирования во всем мире.

К характерным особенностям Python относятся акцент на удобочитаемости исходного кода и функциональных возможностях библиотек, а также конструкция, оптимизирующая производительность труда разработчика, качество программного обеспечения, переносимость программ и интеграцию их компонентов. Программы на Python выполняются на большинстве общеупотребительных платформ, включая Unix и Linux, Windows, Mac OS и .NET, Android, iOS и пр.

В этом карманном справочнике вкратце рассматриваются типы данных и операторы языка Python, имена специальных методов, встроенные функции и исключения, общеупотребительные стандартные библиотечные модули и прочие примечательные языковые средства Python. Этот справочник предназначен в качестве краткого руководства для разработчиков в дополнение к другой литературе с упражнениями, примерами исходного кода и прочим учебным материалом.

В настоящем, *пятом* издании рассматриваются обе версии — Python2.X и 3.X. И хотя основное внимание в нем уделяется версии 3.X, по ходу изложения материала отмечаются отличия, имеющиеся в версии 2.X. В частности, настоящее издание обновлено по версиям Python 2.7 и 3.3, а также наиболее примечательным усовершенствованиям в версии 3.4, хотя большая часть содержания

этого справочника распространяется как на прежние, так и на последующие выпуски в рамках версий 2.X и 3.X.

Настоящее издание охватывает все основные реализации Python, в том числе CPython, PyPy, Jython, IronPython и Stackless. Оно обновлено и дополнено с учетом самых последних изменений в данном языке, его библиотеках и практике программирования на нем. К этим изменениям относятся порядок разрешения методов (MRO) и функция `super()`, формальные алгоритмы наследования, импорт, диспетчеры контекста, расположение кодовых блоков с отступами, наиболее употребительные библиотечные модули и инструментальные средства, в том числе `json`, `timeit`, `random`, `subprocess`, `enum`, а также новое средство запуска программ в Windows.

Принятые условные обозначения

В этой книге приняты следующие условные обозначения:

[]

В форматах синтаксиса элементы в квадратных скобках являются дополнительными, но *не* обязательными. Квадратные скобки используются буквально в некоторых частях синтаксиса Python, например, для обозначения списков, как отмечается там, где это уместно.

*

В форматах синтаксиса элементы, следующие после знака звездочки, могут повторяться многократно или вообще не повторяться. Знак звездочки используется буквально в некоторых частях синтаксиса Python, например, для обозначения арифметической операции умножения.

a | b

В форматах синтаксиса элементы, разделяемые знаком прямой черты, являются альтернативными. Знак прямой черты используется буквально в некоторых частях синтаксиса Python, например, для обозначения операции логического сложения.

Курсив

Служит для обозначения компонентов Python, а также для выделения новых и важных терминов.

Моноширинный шрифт прямой

Служит для обозначения исходного кода, имен файлов, модулей, операторов, функций, атрибутов, переменных и методов.

Моноширинный шрифт наклонный

Служит для обозначения имен заменяемых параметров в синтаксисе команд, вводимых из командной строки, выражений, функций и методов.

Моноширинный шрифт полужирный

Служит для обозначения параметров и аргументов командной строки, а также команд Python, операторов, знаков и цифр там, где это уместно.

Function()

Обозначает вызываемые функции и методы с завершающими круглыми скобками, чтобы отличать их от других типов атрибутов, кроме тех случаев, где на это указывается специально.

См. раздел “Заголовки раздела”.

Обозначает ссылку на другие разделы этой книги с указанием их заголовков в двойных кавычках.

НА ЗАМЕТКУ

Принятые в этой книге обозначения 2.X и 3.X указывают на то, что рассматриваемая тема относится ко всем общепотребительным выпускам Python в рамках версий 2.X и 3.X. А более конкретные номера версий указываются в темах с более ограниченными рамками (например, обозначение 2.7 указывает только на версию 2.7). Будущие изменения в Python могут сделать непригодными некоторые рассматриваемые здесь языковые средства в последующих

версиях, поэтому в дальнейшем рекомендуется обращаться к разделу “What’s New in Python” (Что нового в Python) документации на Python по адресу <http://docs.python.org/3/whatsnew/index.html>.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши электронные адреса:

E-mail: info@williamspublishing.com

WWW: <http://www.williamspublishing.com>

Наши почтовые адреса:

в России: 127055, г. Москва, ул. Лесная, д.43, стр. 1

в Украине: 03150, Киев, а/я 152

Запуск программ на Python из командной строки

Командные строки служат для запуска программ на Python из командной оболочки системы в следующем формате:

```
python [параметр*]  
[ файл_сценария | -с команда | -m модуль | - ] [arg*]
```

В этом формате *python* обозначает интерпретатор языка Python, исполняемый как по полностью указанному пути к каталогу, так и по слову *python*, зарезервированному в командной оболочке системы (например, в переменной окружения PATH). Параметры командной строки (*параметр*), определяющие режим работы интерпретатора Python, обычно указываются перед именем исполняемой программы, тогда как аргументы (*arg*), требующиеся для выполнения программы, — после ее имени.

Параметры командной строки в Python

Элементы *параметр* командной строки служат для обозначения режима работы самого интерпретатора Python. В версии Python 3.X допускаются приведенные ниже элементы *параметр* командной строки, а их отличия в версии 2.X см. далее, в разделе “Параметры командной строки в версии Python 2.X”.

-b

Выдать предупреждения об ошибках при вызове функции `str()` с объектом типа `bytes` или `bytearray`, но без аргумента, обозначающего способ кодирования символов, а также при сравнении данных типа `bytes` или `bytearray` с данными типа `str`. А при указании параметра **-bb** выдаются непосредственно ошибки.

-B

Не записывать байт-код в файлы с расширением **.рус** или **.руо** при импорте.

-d

Активизировать вывод результатов отладки синтаксического анализатора (служит для разработчиков ядра интерпретатора Python).

-E

Игнорировать описываемые далее переменные окружения Python (например, переменную PYTHONPATH).

-h

Вывести вспомогательное сообщение и выйти из программы.

-i

Войти в диалоговый режим работы после выполнения сценария. *Совет:* этот параметр удобен для отладки программ после отказов. См. также описание функции `pdb.pm()` в руководстве по библиотекам Python.

-O

Оптимизировать генерируемый байт-код, создавая и используя файлы с расширением **.pyc** для хранения байт-кода. В настоящее время этот параметр не дает заметных преимуществ в производительности.

-OO

Этот параметр действует подобно параметру **-O**, но при его указании удаляются также строки документации из байт-кода.

-q

Не выводить сообщение о версии и авторском праве при запуске программ в диалоговом режиме, начиная с версии Python 3.2.

-s

Не указывать местный каталог пользователя в пути поиска модулей в переменной `sys.path`.

-S

Не подразумевать импорт модулей из местного каталога пользователя при инициализации.

-u

Сделать принудительно небуферизованными и двоичными стандартные потоки вывода данных (*stdout*) и ошибок (*stderr*).

-v

Выводить сообщение всякий раз, когда инициализируется модуль, показывая место, из которого он загружен. Для более подробного вывода этот параметр можно повторить.

-V

Вывести номер версии Python и выйти из программы (этот параметр может быть также указан в виде **--version**).

-W arg

Этот параметр управляет выдачей предупреждений, где аргумент **arg** принимает вид *действие:сообщение:категория:модуль:номер_строки*. См. далее разделы “Каркас предупреждений” и “Исключения категории предупреждений”, а также документацию на модуль `warnings` в справочном руководстве по библиотеке Python (Python Library Reference), доступном по адресу <http://www.python.org/doc/>.

-x

Пропустить первую строку исходного кода, разрешая использовать формы директив `#!cmd`, отличающиеся от принятых в Unix.

-X параметр

Установить параметр в зависимости от реализации, начиная с версии Python 3.2. Поддерживаемые значения, которые принимает *параметр*, приведены в документации на конкретную реализацию.

Указание программ в командной строке

Исполняемый код программы на Python и передаваемые ей аргументы могут быть указаны в командной строке следующими способами:

файл_сценария

Обозначает имя файла сценария на языке Python, который должен выполняться как основной файл программы, находящийся на самом верхнем уровне ее иерархии (например, по команде **python main.py** выполняется код из файла `main.py`). Имя файла сценария может быть указано как по абсолютному, так и по относительному пути (.) и доступно в элементе `sys.argv[0]` списка аргументов. В командных строках на некоторых платформах элемент `python` может быть опущен, если они начинаются с имени файла сценария и не содержат параметры, определяющие режим работы самого интерпретатора Python.

-c команда

Обозначает (в виде символьной строки) исполняемый код Python (например, по команде **python -c "print('spam' * 8)"** в Python выполняется операция вывода на печать). Значение '-c' устанавливается в элементе `sys.argv[0]` списка аргументов.

-m модуль

Выполняет модуль в виде сценария. Поиск *модуля* осуществляется по пути в переменной `sys.path`, а его выполнение — в виде файла, находящегося на самом верхнем уровне иерархии (например, по команде **python -m pdb s.py** модуль `pdb` отладки программ на Python, находящийся в каталоге стандартной библиотеки, выполняется с аргументом `s.py`). Полный путь к модулю указывается в элементе `sys.argv[0]` списка аргументов.

-

Вводит команды Python из стандартного потока ввода (`stdin` — по умолчанию). Входит в диалоговый режим работы, если команды вводятся из стандартного потока ввода `tty` (интерактивного устройства). Значение '-' указывается в элементе `sys.argv[0]` списка аргументов.

arg*

Обозначает, что все остальное в командной строке передается файлу сценария или команде и доступно в элементе `sys.argv[1:]` списка аргументов.

Если ни один из параметров *файл_сценария*, *команда* или *модуль* не указан в командной строке, интерпретатор Python переходит в диалоговый режим работы, вводя команды из стандартного потока ввода *stdin* и используя для этой цели библиотеку *readline* из проекта GNU, при условии, что она установлена, а также задавая значение '-' (пустую строку) в элементе `sys.argv[0]` списка аргументов, если только интерпретатор не вызывается с упомянутым выше параметром `-`.

Помимо традиционных командных строк, вводимых по приглашению системной командной оболочки, программы на Python можно запускать щелчком кнопкой мыши на именах их файлов в графическом пользовательском интерфейсе (ГПИ) проводника по файлам, вызывая функции из стандартной библиотеки Python (например, функцию `os.popen()`), а также выбирая соответствующие команды запуска из меню интегрированных сред разработки (ИСП) вроде IDLE, Komodo, Eclipse или NetBeans.

Параметры командной строки в версии Python 2.X

В версии Python 2.X поддерживается тот же самый формат командной строки, что и в версии 3.X. Но в ней отсутствует поддержка параметра `-b`, что связано с изменениями в строковом типе данных, а также параметров `-q` и `-X`, введенных в версии 3.X. В то же время поддерживаются приведенные ниже дополнительные параметры, доступные в версиях 2.6 и 2.7, а возможно, и в прежних версиях Python.

-t и -tt

Выдают предупреждения о несогласованном совместном употреблении символов табуляции и пробелов при отступах. А при указании параметра `-tt` выдаются непосредственно ошибки. В версии 3.X такое совместное

употребление символов табуляции и пробелов всегда интерпретируется как синтаксические ошибки (дополнительно см. далее раздел “Правила синтаксиса”).

-Q

Это параметры разделения **-Qold** (по умолчанию), **-Qwarn**, **-Qwarnall** и **-Qnew**. Эти параметры относятся к новому режиму подлинного разделения в версии Python 3.X (см. далее раздел “Примечания к применению операторов”).

-3

Выдает предупреждения о любых признаках несовместимости с версией Python 3.X в исходном коде, которую не в состоянии заведомо устранить инструментальное средство *2to3* из стандартной установки Python.

-R

Активизирует псевдослучайную загрузку, чтобы сделать непредсказуемыми хеш-значения различных типов в промежутках между последовательными вызовами интерпретатора и тем самым предотвратить атаки типа отказа в обслуживании. Этот параметр появился в версии Python 2.6.8 и остался ради совместимости в версии 3.X, начиная с выпуска 3.2.3, хотя такого рода рандомизация активизируется по умолчанию, начиная с выпуска 3.3.

Переменные окружения Python

Переменные окружения (или так называемые переменные командной оболочки) устанавливаются на уровне системы. Они доступны в программах и применяются для их глобальной настройки.

Операционные переменные

Ниже перечислены основные, настраиваемые пользователем переменные окружения, связанные с режимом работы сценариев.

PYTHONPATH

Расширяет исходный путь поиска импортируемых файлов модулей. Формат значения этой переменной такой же, как и у значения, устанавливаемого в переменной окружения `PATH` командной оболочки, а именно: имена путей к каталогам разделяются двоеточиями (или точками с запятой в Windows). Если переменная `PYTHONPATH` установлена, то поиск импортируемых файлов или каталогов модулей осуществляется в каждом каталоге, перечисленном в этой переменной слева направо. Значение этой переменной включается в переменную окружения `sys.path`, предназначенную для хранения полного пути поиска импортируемых модулей в крайних слева составляющих абсолютного пути, после каталога со сценарием и перед каталогами стандартных библиотек. См. далее описание переменной окружения `sys.path` в разделах “Модуль `sys`” и “Оператор `import`”.

PYTHONSTARTUP

Если в этой переменной задано имя читаемого файла, то команды Python из этого файла выполняются до появления первого приглашения в диалоговом режиме работы, что удобно для определения часто используемых инструментальных средств.

PYTHONHOME

Если эта переменная установлена, то ее значение используется в качестве каталога библиотечных модулей с чередующимся префиксом (или `sys.prefix`, `sys.exec_prefix`). По умолчанию поиск модулей осуществляется по пути `sys.prefix/lib`.

PYTHONCASEOK

Если эта переменная установлена, то в операторах импорта регистр в именах файлов не учитывается (в настоящее время поддерживается только в Windows и Mac OS X).

PYTHONIOENCODING

Этой переменной присваивается символьная строка формата `наименование_кодировки[:обработчик_ошибок]`

для замены кодировки в уникоде (и дополнительно — обработчика ошибок) при вводе текста из стандартного потока *stdin* и его выводе в стандартные потоки *stdout* и *stderr*. Такая настройка может потребоваться в некоторых командных оболочках для обработки текста, не представленного в коде ASCII. Так, если вывод текста окажется неудачным, можно попробовать задать в этой переменной кодировку UTF-8 (т.е. значение "utf8").

PYTHONHASHSEED

Если в этой переменной задано случайное начальное значение, оно используется для хеширования объектов типа *str*, *bytes* и *datetime*. А для получения хеш-значений с предсказуемым начальным значением в этой переменной можно установить целое значение в пределах от 0 до **4294967295**. Такая возможность поддерживается в версиях Python 2.6.8 и 3.2.3.

PYTHONFAULTHANDLER

Если эта переменная установлена, то обработчики регистрируются при запуске программы на Python, чтобы вывести содержимое оперативной памяти и отследить снизу вверх причины фатальных ошибок. Начиная с версии Python 3.3, это равнозначно указанию параметра **-X** *обработчик_отказов* в командной строке.

Переменные, аналоги параметров командной строки в Python

Приведенные ниже переменные окружения являются аналогами некоторых параметров командной строки в Python (см. раздел “Параметры командной строки в Python”).

PYTHONDEBUG

Если эта переменная не пустая, то она действует аналогично параметру **-d**.

PYTHONDONTWRITEBYTECODE

Если эта переменная не пустая, то она действует аналогично параметру **-B**.

PYTHONINSPECT

Если эта переменная не пустая, то она действует аналогично параметру **-i**.

PYTHONNOUSERSITE

Если эта переменная не пустая, то она действует аналогично параметру **-s**.

PYTHONOPTIMIZE

Если эта переменная не пустая, то она действует аналогично параметру **-O**.

PYTHONUNBUFFERED

Если эта переменная не пустая, то она действует аналогично параметру **-u**.

PYTHONVERBOSE

Если эта переменная не пустая, то она действует аналогично параметру **-v**.

PYTHONWARNINGS

Если эта переменная не пустая, то она действует аналогично параметру **-W** с тем же самым значением. Эта переменная принимает также разделенную запятыми символьную строку, равнозначную указанию нескольких параметров **-W**. Такая возможность имеется в версиях Python 2.7 и 3.2.

Запуск программ на Python в Windows

Начиная с версии Python 3.3 в Windows (и только в этой операционной системе) устанавливается средство запуска сценариев, которое было доступно отдельно в прежних версиях. Это средство состоит из исполняемых файлов `py.exe` (консольный вариант) и `pyw.exe` (бесконсольный вариант), которые можно вызывать без настроек переменной окружения `PATH`. Эти файлы

регистрируются для выполнения файлов Python через сопоставления типов файлов и позволяют выбирать версии Python с помощью:

- Unix-подобных директив `#!`, указываемых в самом начале сценариев;
- аргументов командной строки;
- параметров, настраиваемых по умолчанию.

Директивы запуска файлов

Средство запуска распознает строки кода с директивами `#!` в самом начале файлов сценариев. В этих директивах указываются версии Python в одной из приведенных ниже форм, где `*` обозначает использование версии *по умолчанию* (в настоящее время — версии **2**, если она установлена, что равнозначно пропуску директивы `#!`); номера *основной* версии (например, **3**) для запуска самой последней установленной версии или же *полной* спецификации в форме *основная_версия.упрощенная_версия* с дополнительным суффиксом `-32`, обозначающим предпочтительный вариант установки 32-разрядной версии (например, `3.1-32`).

```
#!/usr/bin/env python*
#!/usr/bin/python*
#!/usr/local/bin/python*
#!python*
```

Любые аргументы команд Python (`python.exe`) могут быть заданы в конце строки, а в Python 3.4 и более поздней версии можно обратиться к переменной окружения `PATH` для получения строк с директивами `#!`, предоставляющими только обозначение `python` без явного указания номера версии.

Командные строки для запуска

Средство запуска может быть также вызвано из командных строк по приглашению системной командной оболочки в следующей форме:

```
py [pyarg] [pythonarg*] script.py [scriptarg*]
```

В более общем случае все, что может появиться в команде `python` после составляющей `python`, может также появиться после дополнительного аргумента `pyarg` в команде `py` и быть дословно передано средству запуска программ на Python. К этому относятся параметры `-m`, `-c` и `-` из форм спецификации программ (см. раздел “Запуск программ на Python из командной строки”).

Средство запуска принимает приведенные ниже формы своего дополнительного, но не обязательного аргумента `pyarg`, отражающие составляющую `*` из строки с директивой `#!` в конце файла сценария.

```
-2          Запустить последнюю установленную версию 2.X
-3          Запустить последнюю установленную версию 3.X
-X.Y        Запустить указанную версию, где X равно 2 или 3
-X.Y-32     Запустить указанную 32-разрядную версию
```

Если присутствует и то и другое, то предпочтение отдается аргументам командной строки над значениями в строках с директивой `#!`. При соответствующей установке строки с директивой `#!` могут применяться и в более широком контексте (например, при выборе пиктограмм щелчком мышью).

Переменные окружения для запуска

В средстве запуска распознаются также настройки дополнительных переменных окружения, которые могут быть использованы для специальной настройки выбора версии в стандартных или частных случаях (например, пропуск версии, ее указание только в основной директиве `#!` или в аргументе командной строки `py`), как показано ниже. Все эти настройки используются только в исполняемых файлах средства запуска, но не при непосредственном вызове команды `python`.

```
PY_PYTHON   Версия для стандартных случаев (иначе версия 2)
PY_PYTHON3  Версия для 3 частных случаев (например, 3.2)
PY_PYTHON2  Версия для 2 частных случаев (например, 2.6)
```