**Стиль программирования**

**Стиль программирования** - дополнительные ограничения, накладываемые на структуру и вид программного кода группой совместно работающих программистов с целью получения удобных для применения, легко читаемых и эффективных программ. Основные ограничения на вид программы дает синтаксис языка программирования, и его нарушения вызывают синтаксические ошибки. Нарушение стиля не приводит к синтаксическим ошибкам, однако, как отдельные программисты, так и целые коллективы сознательно ограничивают себя в средствах выражения ради упрощения совместной разработки, отладки и сопровождения программного продукта.

Стиль программирования затрагивает практически все аспекты написания кода:

* именование объектов в зависимости от типа, назначения, области видимости;
* оформление функций, методов, классов, модулей и их документирование в коде программы;
* декомпозиция программы на модули с определенными характеристиками;
* способ включения отладочной информации;
* применение тех или иных функций (методов) в зависимости от предполагаемого уровня совместимости разрабатываемой программы с различными компьютерными платформами;
* ограничение используемых функций из соображений безопасности.

Для языка Python Гвидо ван Россум разработал официальный стиль. С оригинальным текстом "Python Style Guide" можно ознакомиться по адресу [http://www.python.org/doc/essays/styleguide](http://www.python.org/doc/essays/styleguide" \t "_blank).

Наиболее существенные положения этого стиля перечислены ниже. В случае сомнений хорошим образцом стиля являются модули стандартной библиотеки.

* Рекомендуется использовать отступы в 4 пробела.
* Длина физической строки не должна превышать 79 символов.
* Длинные логические строки лучше разбивать неявно (внутри скобок), но и явные методы вполне уместны. Отступы строк продолжения рекомендуется выравнивать по скобкам или по первому операнду в предыдущей строке. Текстовый редактор Emacs в режиме python-mode и некоторые интегрированные оболочки (IDE) автоматически делают необходимые отступы в Python-программах:
* def draw(figure, color="White", border\_color="Black",
* size=5):
* if color == border\_color or \
* size == 0:
* raise "Bad figure"
* else:
* \_draw(size, size, (color,

border\_color))

* Не рекомендуется ставить пробелы сразу после открывающей скобки или перед закрывающей, перед запятой, точкой с запятой, перед открывающей скобкой при записи вызова функции или индексного выражения. Также не рекомендуется ставить более одного пробела вокруг знака равенства в присваиваниях. Пробелы вокруг знака равенства не ставятся в случае, когда он применяется для указания значения по умолчанию в определении параметров функции или при задании именованных аргументов.
* Также рекомендуется применение одиночных пробелов вокруг низкоприоритетных операций сравнения и оператора присваивания. Пробелы вокруг более приоритетных операций ставятся в равном количестве слева и справа от знака операции.

Несколько рекомендаций касаются написания комментариев.

* Комментарии должны точно отражать актуальное состояние кода. (Поддержание актуальных комментариев должно быть приоритетной задачей!) После коротких комментариев можно не ставить точку, тогда как длинные лучше писать по правилам написания текста. Автор Python обращается к неанглоязычным программистам с просьбой писать комментарии на английском, если есть хотя бы небольшая вероятность того, что код будут читать специалисты, говорящие на других языках.
* Комментарии к фрагменту кода следует писать с тем же отступом, что и комментируемый код. После " # " должен идти одиночный пробел. Абзацы можно отделять строкой с " # " на том же уровне. Блочный комментарий можно отделить пустыми строками от окружающего кода.
* Комментарии, относящиеся к конкретной строке, не следует использовать часто. Символ " # " должен отстоять от комментируемого оператора как минимум на два пробела.
* Хороший комментарий не перефразирует программу, а содержит дополнительную информацию о действии программы в терминах предметной области.

Все модули, классы, функции и методы, предназначенные для использования за пределами модуля, должны иметь строки документации, описывающие способ их применения, входные и выходные параметры.

* Строка документации для отдельной программы должна объяснять используемые ею ключи, назначение аргументов и переменных среды и другую подобную информацию.
* Для строк документации рекомендуется везде использовать утроенные кавычки ( """ ).
* Однострочная документация пишется в императиве, как команда: "делай это", "возвращай то".
* Многострочная документация содержит расширенное описание модуля, функции, класса. Она будет смотреться лучше, если текст будет написан с тем же отступом, что и начало строки документации.
* Документация для модуля должна перечислять экспортируемые функции, классы, исключения и другие объекты, по одной строке на объект.
* Строка документации для функции или метода должна кратко описывать действия функции, ее входные параметры и возвращаемое значение, побочные эффекты и возможные исключения (если таковые есть). Должны быть обозначены необязательные аргументы и аргументы, не являющиеся частью интерфейса.
* Документация для класса должна перечислять общедоступные методы и атрибуты, содержать рекомендации по применению класса в качестве базового для других классов. Если класс является подклассом, необходимо указать, какие методы полностью заменяют, перегружают, а какие используют, но расширяют соответствующие методы надкласса. Необходимо указать и другие изменения по сравнению с надклассом.
* Контроль версий повышает качество процесса создания программного обеспечения. Для этих целей часто используются RCS или CVS. "Python Style Guide" рекомендует записывать $Revision: 1.31 $ в переменную с именем \_\_version\_\_, а другие данные заключать в комментарии " # ".

Сегодня сосуществуют несколько более или менее широко распространенных правил именования объектов. Программисты вольны выбрать тот, который принят в их организации или конкретном проекте. Автор Python рекомендует придерживаться нижеследующих правил для именования различных объектов, с тем чтобы это было понятно любому программисту, использующему Python.

* Имена модулей лучше давать строчными буквами, например, shelve, string, либо делать первые буквы слов заглавными, StringIO, UserDict. Имена написанных на C модулей расширения обычно начинаются с подчеркивания " \_ ", а соответствующие им высокоуровневые обертки - с прописных букв: \_tkinter и Tkinter.
* Ключевые слова нельзя использовать в качестве имен, однако, если все-таки необходимо воспользоваться этим именем, стоит добавить одиночное подчеркивание в конце имени. Например: class\_.
* Классы обычно называют, выделяя первые буквы слов прописными, как в Tag или HTTPServer.
* Имена исключений обычно содержат в своем составе слово "error" (или "warning"). Встроенные модули пишут это слово со строчной буквы (как os.error ) (но могут писать и с прописной): distutils.DistutilsModuleError.
* Функции, экспортируемые модулем, могут именоваться по-разному. Можно давать с прописных букв имена наиболее важных функций, а вспомогательные писать строчными.
* Имена глобальных переменных (если таковые используются) лучше начинать с подчеркивания, чтобы они не импортировались из модуля оператором from-import со звездочкой.
* Имена методов записываются по тем же правилам, что и имена функций.
* Имена констант (имен, которые не должны переопределяться) лучше записывать прописными буквами, например: RED, GREEN, BLUE.
* При работе с языком Python необходимо учитывать, что интерпретатор считает некоторые классы имен специальными (обычно такие имена начинаются с подчеркивания).