ПРАВИЛА ХОРОШЕГО ТОНА

План:

- Правила хорошего тона;
- Модуль Argparse;
- Конфигурационные файлы;
- Переменные окружения.

Правила хорошего тона

Приложение должно настраиваться одним или несколькими из трех путей:

- Использовать аргументы командной строки;
- Использовать конфигурационные файлы;
- Использовать переменные окружения.

Важно: в Python-приложениях почти всегда рекомендуется писать логи в структурном виде, используя для этого встроенный модуль logging. Одно из простых правил для современного приложения гласит «все логи пишутся на стандартный вывод», это обусловлено в том числе удобством дальнейшей сборки этих логов. Например, при запуске в Docker-контейнере все, что залогирует приложение, попадет в docker logs, где мы сможем увидеть это и поработать с ним дальше.

Одним из известных манифестов со списком рекомендаций является «Двенадцатифакторные приложения». Приложение существует как на английском языке, так и на русском. Следует отметить, что многие заявления спорны, однако с точки зрения стартовой позиции с правилами стоит ознакомиться.

Argparse

Если говорить про аргументы командной строки и конфиги, то здесь обычно проще всего не “изобретать велосипед”, а использовать встроенные модули.

Например, в Python встроен модуль Argparse. Если вы пишете скрипт, который конфигурироваться извне без изменения кода, то вам может помочь класс ArgumentParser, через который можно задать используемые скриптом аргументы и потом автоматически распарсить их для использования в коде. Наверное, сейчас это один из самых простых и удобных способов работы с аргументами.

Есть и альтернативные решения, например, проект click. Он использует несколько другой подход с применением декораторов, однако в условиях прицела на удобство поддержки встроенный вариант можно назвать более рекомендованным.

Конфигурационные файлы

В конфигурационном файле должны быть все настройки программы, которые мы можем захотеть поменять без модификации ее кода.

Формат конфига может быть любым. Чаще всего в проектах можно встретить YAML (для работы с ним используется модуль PyYAML), TOML (для версий Python <3.11 есть внешняя библиотека, начиная с 3.11 появилась встроенная), или же более классический формат INI (можно использовать встроенный ConfigParser).

Если приложение на Python запускается напрямую в системе (без Docker-контейнера), то помимо конфигурационного файла по умолчанию может существовать дополнительный глобальный файл конфигурации (или даже несколько), чаще всего располагающийся в каталоге /etc/<имя проекта>. При

такой схеме параметры в глобальном файле должны будут переопределять параметры по умолчанию.

Если же приложение запускается в контейнере, то иногда применяется схема, при которой конфигурационный файл просто монтируется извне внутрь контейнера при старте, подменяя собой конфиг по умолчанию.

Переменные окружения

Когда настраиваемых параметров не очень много, либо же приложение запускается в Docker-контейнере, часто используют конфигурацию через переменные окружения, задавая их извне. Это можно делать, например, в unit-файле systemd или в docker-compose.yml.

Как вам урок?

Далее