Логирование в Python

Колодяжный Иван,
http://blog.e0ne.info/
http://kharkivpy.org.ua/
@e0ne

http://www.devconf.ru

О чем доклад?

- Что такое логирование и зачем нужно
- Основные ошибки и best practicies
- Коротко о документации
- Логирование в реальных условиях
- Вопросы, ответы и комментарии

- **Внимание!** Мнение автора может не совпадать с мнением других участников конференции. Все сказанное основаное на личном опыте разработки, отладки и чтении документации

Логирование: что это такое и зачем нужно?

- Файл регистрации, протокол, журнал или лог (англ. log) — файл (в большинстве случаев) с записями о событиях в хронологическом порядке. Различают регистрацию внешних событий и протоколирование работы самой программы - источника записей (хотя часто всё записывается в единый файл).

Логирование: что это такое и зачем нужно?

- Текущее и прошлое состояние приложение
- Часто - единстенный возможный способ отладки в production окружении
- Хорошие логи в большинстве случаев избавляет от необходимости запуска отладчика

Как это бывает

- Python: всего два* способа

Как это бывает

- Python: всего два* способа:

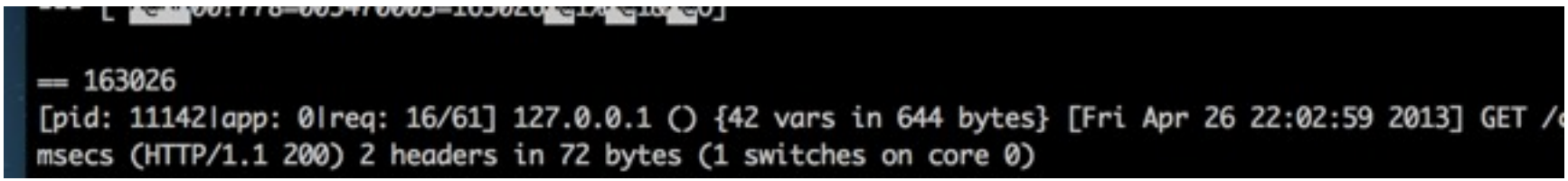
- Функция print() - быстро, просто, неудобно
- “copy & paste” паттерн

Как это бывает: функция print()

- За:
  - удостота
  - “удобство”

Как это бывает: функция print()

- За:
  - удостота
  - “удобство”
- Против
  - гибкость
  - поддерживаемость
  - “понятность”



Распространенные ошибки

- “copy & paste” паттерн
- “Один за всех и все за одного”
- Неправильное использование уровней логирования
- Неправильная настройка

Распространенные ошибки

- “copy & paste” паттерн: проблема

```
class SampleHandler(BaseHandler):

    def PUT(self, id):
        logger.debug('SampleHandler: PUT request with id %s' % id)
        # ...

    def POST(self, id):
        logger.debug('SampleHandler: POST request with id %s' % id)
        # ...
```

Распространенные ошибки

- “copy & paste” паттерн: решение

- декораторы (@log_request, @log_action, etc.)
- метаклассы
- middleware для web-приложений

Распространенные ошибки: “Один за всех и все за одного”

- Один логгер для всего
  - трудно читать (grep и прочее не предлагать)
  - невозможно поддерживать и настраивать

Распространенные ошибки

- Неправильное использование уровней логирования
- Неправильная настройка

- Дебаг сообщения с пометкой CRITIAL на production

Как нужно

- Стандартные подходы
- Централизованное хранение и обработка
- import logging

Как нужно: стандартные подходы

- EventLog (*nix)
- NT EventLog (Windows)
- Формат сообщений
- Общепринятое хранение (например, /var/log/devconf.log)

Документация

- 1 пакет
- 3 модуля

```
(.venv)bash-3.2$ ls logging/*.py
logging/__init__.py     logging/config.py       logging/handlers.py
```

Документация

- PEP 282
- 1 пакет
- 3 модуля
- 2 страницы на docs.python.org
- 3 страницы описания стандартных модулей

Документация: исходники

- конфигурация
- хэндлеры (handlers)
- логгеры (loggers)

Документация: исходники

- конфигурация
  - во время выполения (runtime)
  - конфиг-файлы
    - .py
    - .ini
    - dictconf(json, yaml, python 2.7+)
- хэндлеры (handlers)
- логгеры (loggers)

Документация: исходники

- конфигурация
- хэндлеры (handlers)
  - ~10 штук ()
  - *FileHandler
  - SocketHandler
  - SysLogHandler / NTEventLogHanlder
  - HttpHandler
  - etc
- логгеры (loggers)

Немного практики

- http://github.com/e0ne/devconf

Спасибо! Вопросы?

- E-mail: e0ne@e0ne.info
- @e0ne

- http://blog.e0ne.info
- http://kharkivpy.org.ua

- http://github.com/e0ne/devconf