BIGDATA TEAM

HW #04: Logging Asset CLI

1. Описание задания

В этом задании вам нужно настроить логирование консольной утилиты, позволяющей оценивать инвестиционную прибыльность активов. Цель задания:

1. Научиться настраивать логирование;
2. Научиться тестировать логирование (см. pytest:caplog);

1.1. Консольное приложение по работе с активами

Консольное приложение по работе с активами доступно на Github курса:

- https://github.com/big-data-team/python-course/blob/master/asset.py

Приложение предоставляет возможность оценить инвестиционную прибыль от актива за разные периоды времени (в годах). Пример использования:

```
$ python3 asset.py --logging-config task_*asset_log.conf.yml --filepath asset.txt --periods 1 2 5
```

Где asset.txt содержит информацию об активе:

```
<name> <capital> <interest>
```

- name - имя актива
- capital - стоимость актива (например в рублях)
- interest - годовая прибыльность актива, выраженная в процентах (0.1 - значит 10%)

Приложение уже осуществляет логирование разного уровня событий (debug, info, warn). Вам остается сконцентрироваться на настройке логирования, чтобы писать логи в файлы соответствующего уровня, и на самих тестах.

1.2. Требования к реализации

На выходе ожидаются два лог-файла:

1. `asset_log.debug` - содержит журнальную информацию со всеми событиями уровня DEBUG+:
2. `asset_log.warn` - содержит только предупреждения и ошибки (уровень WARN+).

В stderr пишутся все журнальные сообщения уровня INFO+.

Формат строк для Formatter:

```
"%(asctime)s %(name)s %(levelname)s %(message)s"
```

2. Рекомендации

Рекомендуемая последовательность реализации функциональности:

1. Добавьте тесты на имеющуюся функциональность вывода логов (TDD);
2. Добавьте логирование сначала в stderr, затем в файлы. Расширяйте конфигурацию логирования итеративно с помощью последовательного создания Python объектов logging.{Formatter,*Handler} и им подобных;
3. Научитесь правильно считывать и сохранять ascii картинки в YAML файлах. Для вдохновения посмотрите на ascii картинки на сайте https://www.asciiart.eu/;
4. Перенесите конфигурацию логирования в YAML файл.

Материалы для погружения:

1. https://docs.python.org/3/howto/logging.html
2. https://docs.python.org/3/library/logging.config.html
3. https://docs.python.org/3/howto/logging-cookbook.html

В период реализации ваших тестов вы можете натолкнуться на ошибку (см. stderr при записи сообщений из logger, пример issue на github pytest):

```
ValueError: I/O operation on closed file
```

К сожалению, если в период тестирования самостоятельно настраивать логирование в различные потоки (использовать кастомный конфиг, который требуется написать в рамках этого задания), то pytest может не отловить эту историю и подсунуть в captured stderr (capsys) уже закрытый поток после прошлого тестирования. Чтобы победить эту историю можно воспользоваться советами из issue (пример), либо организовать тесты следующим образом (тесты запускаются по очереди определения в тест-файле):

- сначала все тесты базовой функциональности
- затем тесты на проверку сообщений в логере
- затем (один) тест, который содержит изменение стримов для логирования, где можно использовать capsys

3. Критерии оценивания

Балл за задачу складывается из:

- **70%** - правильная реализация логирования
- **30%** - качество покрытия тестами
  - оценка качества проводится автоматически вызовом pytest:
    - `PYTHONPATH=. pytest -v --cov=asset test_*_asset_log.py`
    - уровень покрытия тестами должен быть выше 80%
    - проверяем код Python версии 3.7 с помощью `pytest==6.0.1`
    - точная формула: 20% x min([test_coverage / 0.8], 1.0)

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения в после soft deadline и до hard deadline
- **5%** за каждую посылку после 2й посылки в день (каждый день можно делать до 2х посылок без штрафа)

лучший балл с 1-й попытки: 100%
лучший балл со 2-й попытки: 100%
лучший балл с 3-й попытки: 95%
лучший балл с 4-й попытки: 90%

4. Инструкция по отправке задания

Оформление задания:

- Код задания (Short name): **HW04:Logging Asset CLI**
- Выполненное ДЗ запакуйте в архив **`PY-MADE-2021-Q4`**`_<Surname>_<Name>_HW#`**`.zip`**, пример -- **`PY-MADE-2021-Q4`**`_Dral_Alexey_`**`HW04`**`.zip`. (Проверяйте отсутствие пробелов и невидимых символов после копирования имени отсюда.[1]) Если ваше решение лежит в папке my_solution_folder, то для создания архива hw.zip на Linux и Mac OS выполните команду[2]:
  - `zip -r hw.zip my_solution_folder/*`
- На Windows 7/8/10: необходимо выделить все содержимое директории my_solution_folder/ нажать правую кнопку мыши на одном из выделенных

[1] Онлайн инструмент для проверки: https://www.soscisurvey.de/tools/view-chars.php
[2] Флаг -r значит, что будет совершен рекурсивный обход по структуре директории

объектов, выбрать в открывшемся меню "Отправить >", затем "Сжатая ZIP-папка". Теперь можно переименовать архив.

- Решение задания должно содержаться в одной папке.
- Перед проверкой убедитесь, что дерево вашего архива выглядит так:
  - `| PY-MADE-2021-Q4_<Surname>_<Name>_HW04.zip`
  - `| ---- task_<Surname>_<Name>_asset_log.conf.yml`
  - `| ---- test_<Surname>_<Name>_asset_log.py`
  - ~~`| ---- *.txt`~~[3]
  - При несовпадении дерева вашего архива с представленным деревом, ваше решение не будет возможным автоматически проверить, а значит, и оценить его.
- Для того, чтобы сдать задание, необходимо:
  - Зарегистрироваться и залогиниться в сервисе Everest
  - Перейти на страницу приложения: MADE Python Grader
  - Выбрать вкладку Submit Job (если отображается иная).
  - Выбрать в качестве “Task” значение: **HW04:Logging Asset CLI**[4]
  - Загрузить в качестве “Task solution” файл с решением
  - В качестве Access Token указать тот, который был выслан в личные сообщения в Telegram от @bdt_support
- **Перед отправкой задания**, оставьте, пожалуйста, отзыв о домашнем задании по ссылке: https://rebrand.ly/pymade2021q4_feedback_hw. Это позволит нам скорректировать учебную нагрузку по следующим заданиям (в зависимости от того, сколько часов уходит на решение ДЗ), а также ответить на интересующие вопросы.

Внимание: если до дедлайна остается меньше суток, и вы знаете (сами проверили или коллеги сообщили), что сдача решений сломана, обязательно сдайте свое решение, прислав нам ссылку на выполненное задание (Job) на почту с темой письма “Short name. ФИО.”. Например: “**HW04:Logging Asset CLI**. Иванов Иван Иванович.” Таким образом, мы сможем увидеть какое решение у вас было до дедлайна и сможем его оценить. Пример ссылки:

- https://everest.distcomp.org/jobs/67893456230000abc0123def

Любые вопросы / комментарии / предложения пишите согласно предложениям на портале.

Всем удачи!

[3] Все необходимые тестовые данные нужно генерировать с помощью pytest fixture tmpdir и объектов в памяти Python

[4] Сервисный ID: python.asset_log

5. FAQ (часто задаваемые вопросы)

"You are not allowed to run this application", что делать?
Если Вы видите надпись "You are not allowed to run this application" во вкладке Submit Job в Everest, то на данный момент сдача закрыта (нет доступных для сдачи домашних заданий, по техническим причинам или другое). Попробуйте, пожалуйста, еще раз через некоторое время. Если Вы еще ни разу не сдавали, у коллег сдача работает, но Вы видите такое сообщение, сообщите нам об этом.

Grader показывает 0 или < 0, а отчет (Grading report) не помогает решить проблему
Ситуации: • система оценивания показывает оценку (Grade) < 0, а отчет (Grading report) не помогает решить проблему. Пример: в случае неправильно указанного access token система вернет -401 и информацию о том, что его нужно поправить; • система показывает 0 и в отчете (Grading report) не указано, какие тесты не пройдены. Пример: вы отправили невалидный архив (rar вместо zip), не приложили нужные файлы (или наоборот приложили лишние - временные файлы от Mac OS и т.п.), рекомендуется проверить содержимое архива в консоли: `unzip -l your_solution.zip` Если Вы столкнулись с какой-то из них присылайте ссылку на выполненное задание (Job) в чат курса. Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def

Как правильно настроить окружение, чтобы оно совпадало с тестовым окружением?
1. Если еще не установлено, то установите conda https://docs.conda.io/projects/conda/en/latest/user-guide/install/ 2. Настройте окружение для разработки на основе README.md курса https://github.com/big-data-team/python-course 3. Скачайте необходимые датасеты для выполнения задания https://github.com/big-data-team/python-course#study-datasets

6. Дополнительные задания (не на оценку)

Добавьте возможность регулировать уровень общительности консольного приложения в терминале запуска. Добавьте возможность выводить логи на экран (в stderr) с помощью указания флага verbosity:

- -v - показывать на экране лог сообщения уровня WARN+
- -vv - показывать на экране лог сообщения уровня INFO+
- -vvv (и больше) - показывать на экране лог сообщения уровня DEBUG+

Для этого вам пригодится action="count" в argparse.