BIGDATA TEAM

# #04: Logging, Mock. Workshop.

# 1. Цель занятия

Научиться конфигурировать логирование приложений на Python.

*DISCLAIMER*: Не надо беспокоиться, если у Вы что-либо не успели. У всех разная скорость потребления учебного контента, все материалы остаются доступны и Вы сможете продолжить погружение дома в комфортной обстановке.

# 2. Общие рекомендации

Рекомендуется установить miniconda (или anaconda), чтобы был доступен менеджер пакетов conda. Windows-пользователям рекомендуется установить командную строку git-bash или пользоваться настроенным Linux-контейнером.

Рекомендации по настройке виртуального окружения, а также датасеты для тестирования описаны здесь:

- MAIL-PY-2019-Q4 | Testing, configuration instructions

# 3. Unicode Recap

**Задание**: проверить по таблицам “koi8-r” и “utf-8” соответствие Unicode-символу “ч”

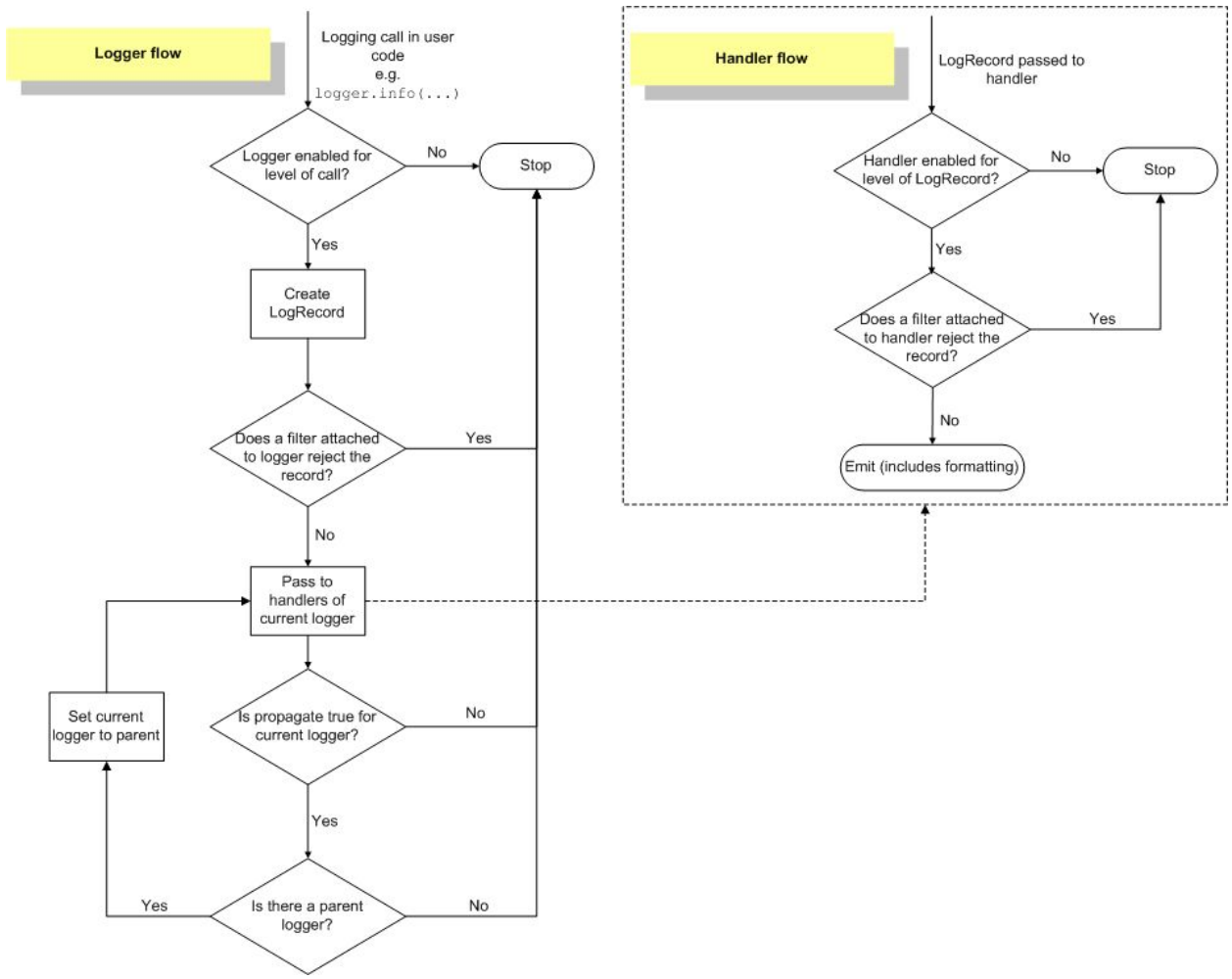
- b'\xd1\x87' (кодировка “utf-8”)
- для преобразования из 16-ричной системы счисления используйте: int(str, 16)
- для получения бинарного представления числа используйте: bin(number)
- таблицы преобразования доступны в слайдах занятия.

# 4. Logging

~~В прошлой серии~~ На прошлом занятии мы остановились на:

- MAIL-PY-2019-Q4 | Logging and Mock, StarterCode, InvertedIndex

Как Вы могли уже догадаться отлаживать приложения с помощью “print” далеко не лучшее решение. Как минимум, всю журнальную информацию (историю поведения программы) следует логировать с помощью специализированных инструментов:




**Задания (basic / intermediate)**:

1. Определим базовый logger приложения (root logger vs application logger);
2. Разметим журнальные записи разным уровнем логов (debug, info, …);
3. Определяем базовую конфигурацию logger, чтобы увидеть результат в логах:

```
logging.basicConfig(
    filename='inverted_index.log', level=logging.DEBUG,
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S",
)
```

4. Добавляем caplog fixture и изучаем содержимое с помощью pytest и pdb (задаем уровень сборщика: --log-level=DEBUG и уровень вывода на консоль --log-cli-level=DEBUG);
5. Напишите тест, где проверяется, что все записи имеют уровень ниже “WARNING” (см. logging._levelToName и обратно);

**Задания (intermediate / advanced)**:

1. Создадим конфигурацию логгера самостоятельно с помощью указания handler и formatter;
2. Создадим конфигурацию логгера с помощью конфига в формате YAML (см. logging.config.dictConfig);
3. Создайте логирование в 2 потока: в один будут приходить все сообщения, в другой - сообщения уровня INFO и выше;

**Задание** на YAML:

- documentation: https://pyyaml.org/wiki/PyYAMLDocumentation
- multiline: https://yaml-multiline.info/

Сохраните следующий YAML в файл, укажите недостающие спецификаторы и прочитайте с помощью Python. Выведите на экран Python объекты, чтобы убедиться, что данные прочитаны правильно (спецификаторы указаны верно).

```
image: <укажите правильные спецификаторы>
  by hjw                 ___
     __                 /.-.\
    /  )_______________\\  Y
   /_ /=== == === === =\ _\_
  ( /)=== == === === == Y   \
   `--------------------(  o  )
                        \___/
sentence: <укажите правильные спецификаторы>
  It will
  become
  one sentence
```

Полезные материалы по Logging:

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

Pytest logging integration:

- caplog (fixture)
- https://docs.pytest.org/en/latest/logging.html

## 5. Обратная связь

Просьба потратить 1-2 минут Вашего времени, чтобы поделиться впечатлением, описать что было понятно, а что непонятно. Мы учитываем рекомендации и имеем возможность переформатируем учебную программу, формат и скорость подачи материалы под Ваши запросы.