Домашнее задание
по теме «Работа с декораторами, контекстными менеджерами»

Формулировка задания

Решить задачи на python на применение декораторов и контекстных менеджеров. Результирующий код должен быть читаемым, с единой системой отступов и адекватными названиями переменных.

Описание плана работы

Решить задачи на python на применение декораторов и контекстных менеджеров. Желательна реализация в файлах py.
Сохранить задачи (сделать коммиты для каждой) в локальном git и опубликовать в удаленном репозитории.

Для отчета по работе выполнить задание в файле.py или .ipynb. Сделать снимки экрана корректного выполнения программы в IDE.

Задача 1 Права администратора

Довольно часто в веб-разработке приходится иметь дело с проблемой ограничения доступа. Некоторые авторизованные пользователи должны иметь доступ к ресурсе, другие же нет. Для решения этой проблемы используются роли: администратор, менеджер, пользователь и т.д. и т.п.

Легче всего решить эту проблему при помощи декоратора **role_required(role: str)**, который будет разрешать или запрещать определенные действия, выполняемые над ресурсом, в нашем примере это будет функция **secret_resource() -> str**.

Исходные условия:

- Определена функция **secret_resource() -> str**, которая должна быть доступна только пользователям с ролью **'admin'**.
- Роль очередного пользователя записана в глобальной переменной **user_role**.

Порядок выполнения:

- Напишите код декоратора **role_required(role: str)**, который в случае, если пользователь является админом предоставляет доступ к функции **secret_resource**, а иначе возвращает строку **'Permission denied'**.

Вход 1:

admin

Выход 1:

Permission accepted

Вход 2:

manager

Выход 2:

Permission denied

Задача 2 Кэширование

Кэширование помогает сохранять результат запроса (например, из БД) и выдавать его сразу, не совершая новых запросов в БД. Важно определить политику кэширования - время жизни данных в секундах или количество раз, сколько их можно запросить (после чего они будут стёрты из кэша). Мы будем использовать вторую политику для простоты.

В этом задании вам придётся реализовать декоратор, который будет принимать несколько аргументов. Так, нужно написать декоратор **cache(db: str)**, который принимает в качестве параметра **db** - название базы данных, где будет кэшироваться информация. Затем подумайте, как можно передать второй параметр - **expiration**, количество раз, когда данные будут браться из кэша, а затем будут стёрты.

При первом запросе необходимо кэшировать результат и возвращать строку вида:

Info about: <thing> from <db>, now cached with expire=<expire_time>

Где:

- **users** - параметр функции **get_info** (возвращающая информацию о предмете);
- **db** - название БД, где будет кэшироваться информация;
- **expire_time** - количество раз, когда данные будут браться из кэша, а затем будут стёрты.

При последующих запросах необходимо выдавать следующие строки:

Info about: <thing> cached in <db>, expire=<expiration_time-1>

Info about: <thing>cached in <db>, expire=<expiration_time-2>

Info about: <thing>cached in <db>, expire=<expiration_time-3>

Вплоть до:

Info about: <thing>cached in <db>, expire=0

Исходные условия:

- Определена функция **get_info(thing: str) -> str**, которая возвращает информацию о предмете **thing**.

Порядок выполнения:

- Напишите декоратор, который будет будет принимать название базы данных и количество раз, когда данные будут браться из кэша. После того, как количество станет равным нулю кэш необходимо обновить актуальными данными (см. примеры).

Вход 1:

bike_store

Выход 1:

Info about: bike_store from postgresql, now cached with expire=5

Info about: bike_store cached in postgresql, expire=4

Info about: bike_store cached in postgresql, expire=3

Info about: bike_store cached in postgresql, expire=2

Info about: bike_store cached in postgresql, expire=1

Info about: bike_store cached in postgresql, expire=0

Info about: bike_store from postgresql, now cached with expire=5

Info about: bike_store from sqlite, now cached with expire=3

Info about: bike_store cached in sqlite, expire=2

Info about: bike_store cached in sqlite, expire=1

Info about: bike_store cached in sqlite, expire=0

Info about: bike_store from sqlite, now cached with expire=3

Вход 2:

users

Выход 2:

Info about: users from postgresql, now cached with expire=5

Info about: users cached in postgresql, expire=4

Info about: users cached in postgresql, expire=3

Info about: users cached in postgresql, expire=2

Info about: users cached in postgresql, expire=1

Info about: users cached in postgresql, expire=0

Info about: users from postgresql, now cached with expire=5

Info about: users from sqlite, now cached with expire=3

Info about: users cached in sqlite, expire=2

Info about: users cached in sqlite, expire=1

Info about: users cached in sqlite, expire=0

Info about: users from sqlite, now cached with expire=3

Задача 3 Контекстный менеджер safe_write

Реализуйте контекстный менеджер **safe_write**, который принимает один аргумент:
filename — имя файла
Контекстный менеджер должен позволять записывать информацию в файл с именем *filename*.
Причем если во время записи в файл было возбуждено какое-либо исключение, контекстный менеджер должен поглотить его, отменить все выполненные ранее записи в файл, если они были, вернуть файл в исходное состояние и проинформировать о возбужденном исключении выводом следующего текста:

Во время записи в файл было возбуждено исключение <тип исключения>

Дополнительная проверка данных на корректность не требуется. Гарантируется, что реализованный контекстный менеджер используется только с корректными данными.

Вход 1:

```
with safe_write('undertale.txt') as file:
    file.write('Я знаю, что ничего не знаю, но другие не знают и этого.')

with open('undertale.txt') as file:
    print(file.read())
```

Выход 1:

Я знаю, что ничего не знаю, но другие не знают и этого.

Вход 2:

```
with safe_write('under_tale.txt') as file:
    file.write('Я знаю, что ничего не знаю, но другие не знают и этого. \n')

with safe_write('under_tale.txt') as file:
    print(
      'Если ты будешь любознательным, то будешь много знающим.',
      file=file,
      flush=True
    )
    raise ValueError

with open('under_tale.txt') as file:
    print(file.read())
```

Выход 2:

Во время записи в файл было возбуждено исключение ValueError

Я знаю, что ничего не знаю, но другие не знают и этого.

Перечень необходимых инструментов

- Python
- venv

- Jupiter Notebook
- IDE VS Code
- GigaIDE

Форма предоставления результата

1. В поле ссылки загрузить ссылку на удаленный репозиторий с доступом для наставника.
2. В поле файла загрузить архив с папкой, в которой разместить отчет со скриншотами по заданиям и решение задач 1-3. Решения должны быть представлены в формате .ipynb или .py.

Шкала оценивания

- **1.0 – отлично**

Учтены все требования, код структурирован и отрабатывает корректно. Задание выполнено на высоком уровне.

- **0.7–0.9 – хорошо**

Код соответствует большинству требований, но может содержать небольшие ошибки или недочеты. Продемонстрирован навык работы с декораторами и с контекстным менеджером. Есть к чему стремиться в изучении темы.

- **0.5–0.6 – удовлетворительно**

Код соответствует большинству требований, но может содержать значительные ошибки или недочеты. Требуется дополнительное изучение темы и исправление программы.

- **Менее 0.5 – задание не выполнено**

Задание выполнено на очень низком уровне или решено 0 задач. Требуется дополнительное изучение темы и решение заданий для практического изучения темы.