Домашнее задание
по теме «Многопоточность в Python»

Формулировка задания

Решить 2 задачи на python на применение многопоточного и многопроцессорного программирования и вспомогательных библиотек threading, multiprocessing. Результирующий код должен быть читаемым, с единой системой отступов и адекватными названиями переменных.

Описание плана работы

Решить 2 задачи на python на применение многопоточного и многопроцессорного программирования и вспомогательных библиотек threading, multiprocessing.

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 Удвоение чисел и получение первого результата

Напишите многопоточный код для обработки чисел из нескольких списков. Каждое число в списке должно быть умножено на 2, с имитацией задержки 0.2 сек на каждой операции. Используйте ThreadPoolExecutor и as_completed для управления потоками и отслеживания результатов.

Подробное описание задачи:

1. У вас есть список списков с числами, которые должны обрабатываться. Пример списка можно загрузить из файла test_list_numbers.txt (находится в материалах к заданию).
2. Реализовать функцию process_number(number), которая принимает число, умножает его на 2, имитирует задержку в 0.2 секунды (рекомендуется через time.sleep(0.2)) и возвращает результат.
3. Инициализировать ThreadPoolExecutor с определенным количеством рабочих потоков (рекомендуется, 10). Использовать метод submit() для отправки задачи обработки каждого числа из всех списков в пул потоков. Сохраните возвращаемые объекты Future в списке.
4. Итерируйтесь через объекты Future, используя as_completed(), чтобы получить результаты задач по мере их завершения.
5. После завершения всех задач, выведите сумму обработанных чисел списка который был обработан быстрее остальных. Вывод программы должен быть следующим:

Сумма чисел в первом обработанном списке: 11090

print(f"Сумма чисел в первом обработанном списке: {first_list_sum}")

Задача 2 Поиск и суммирование чисел через цепочку файлов

Для решения используйте многопроцессность с помощью метода ***pool.starmap***.

Вам дан архив с 1000 текстовыми файлами, Архив с файлами 1 (path_8_8.zip). Задача заключается в том, чтобы написать код, который обрабатывает каждый из этих файлов в многопроцессном режиме.

В каждом файле записан путь к файлу в другом архиве. Архив с файлами 2 (recursive_challenge_8_8.zip). Ваш код должен следовать этому пути, чтобы найти конечный файл, содержащий число. Это число необходимо прибавить к глобальному счётчику.

Требования к коду:

1. Код должен открыть каждый текстовый файл из первого архива, считать путь, указанный внутри.
2. Перейти по указанному пути к целевому файлу и извлечь из него число.
3. Найденные числа необходимо суммировать.

Перечень необходимых инструментов

- Python
- venv
- multiprocessing
- 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 1-2. Решения должны быть представлены в формате .ipynb или .py.

Шкала оценивания

- **1.0 – отлично**

Решены 2 задачи. Учтены все требования, код структурирован. В работе продемонстрировано умение решать сложные задачи параллельного программирования.

- **0.7–0.9 – хорошо**

Решены 2 задачи или 1 задача решена на высоком уровне. Код соответствует большинству требований, но может содержать небольшие ошибки или недочеты. В работе продемонстрирован навык решения сложных задач параллельного программирования. Есть к чему стремиться в изучении темы.

- **0.5–0.6 – удовлетворительно**

Решены 2 задачи или 1 задача с множеством замечаний. 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.