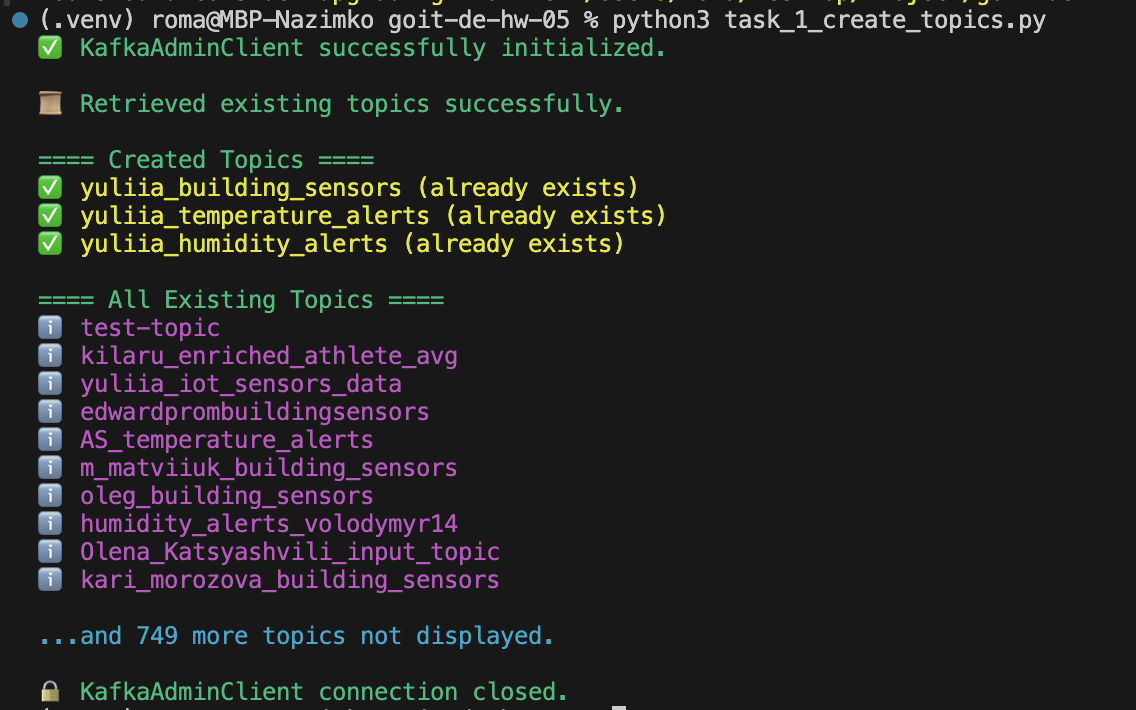
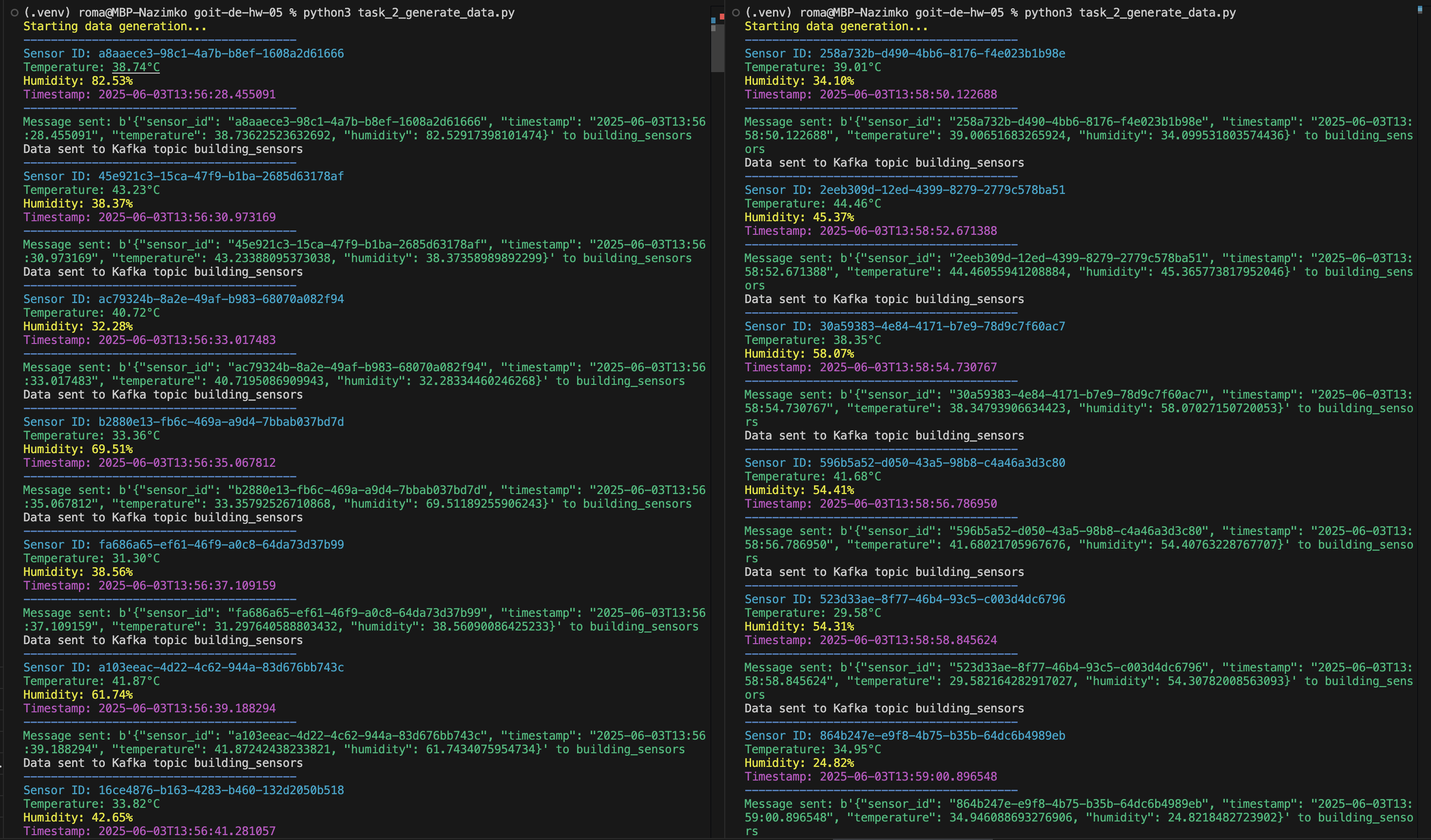
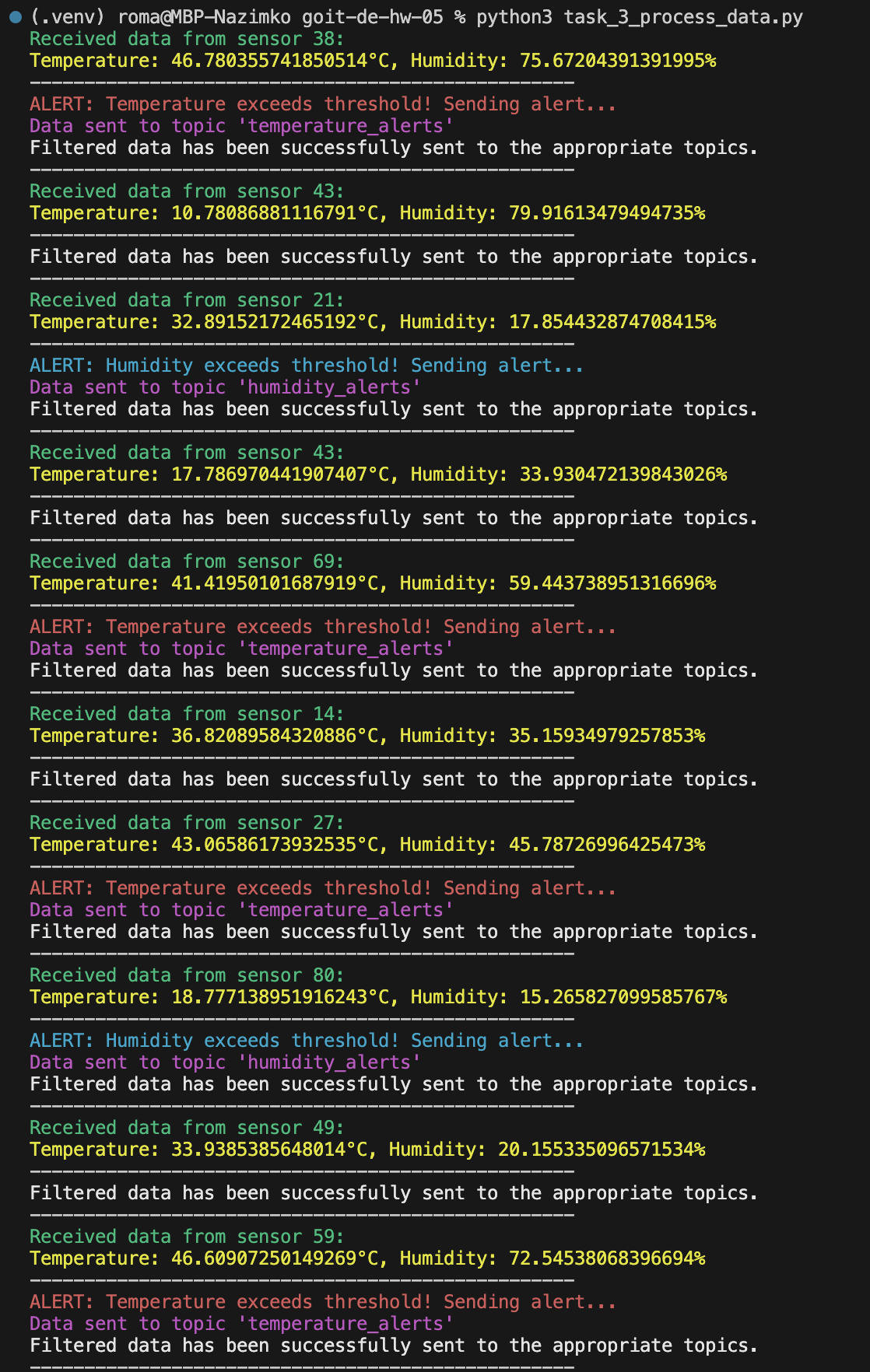
Домашнє завдання до теми «Apache Kafka»

1. Cкрин трьох топіків з команди [print(topic) for topic in admin\_client.list\_topics() if "my\_name" in topic] .

На скриншоті показано успішне виконання програми:

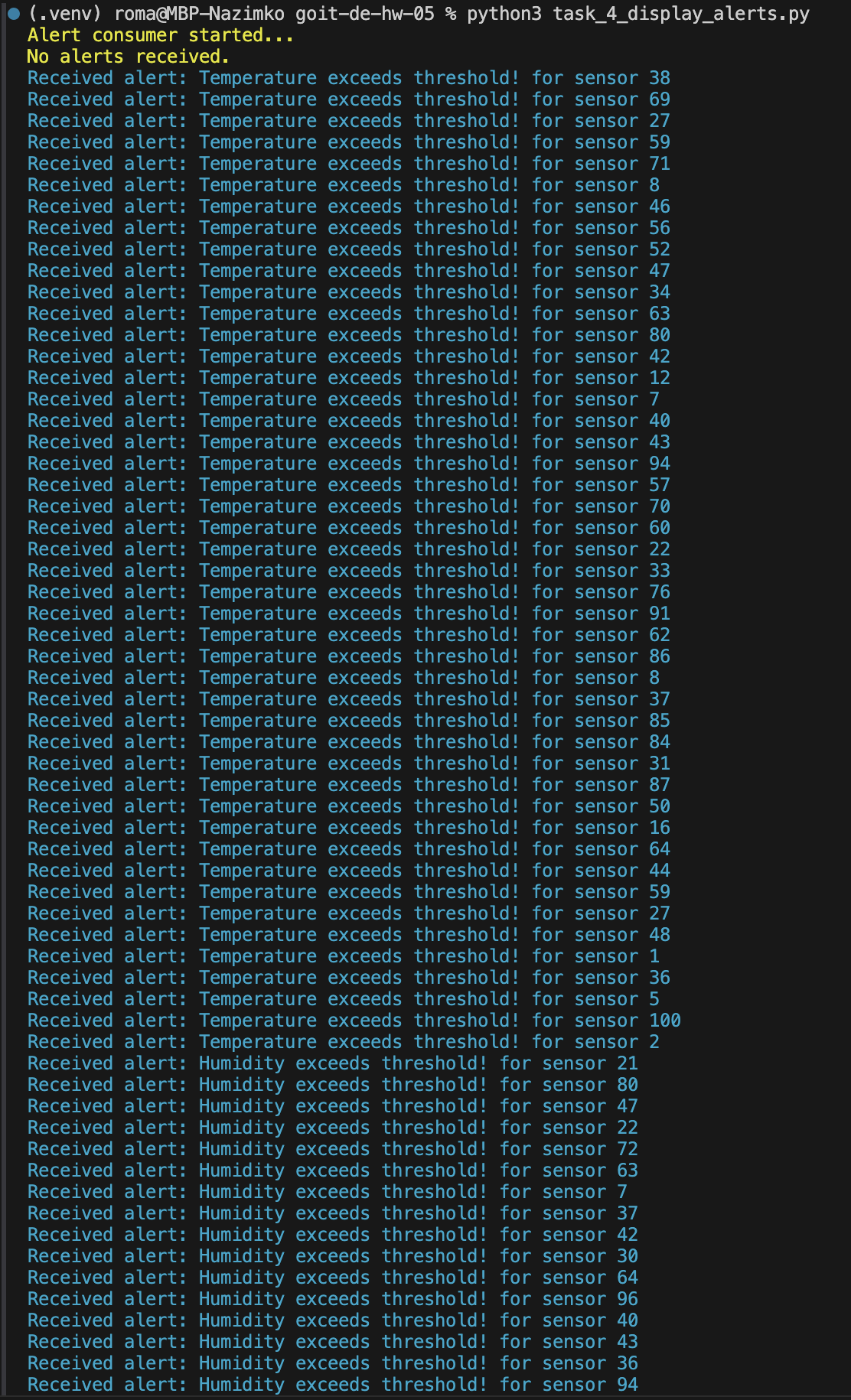
1. **Ініціалізація KafkaAdminClient**
   * Клієнт Kafka успішно ініціалізовано, про що свідчить повідомлення:  
     ✅ KafkaAdminClient successfully initialized.
2. **Перевірка та створення топіків**
   * Виведено три топіки з іменами:
     + yuliia\_building\_sensors
     + yuliia\_temperature\_alerts
     + yuliia\_humidity\_alerts  
       Усі три топіки вже існують у кластері, що підтверджується статусом:  
       (already exists).
3. **Список існуючих топіків**
   * У розділі ==== All Existing Topics ==== виведено частину списку всіх існуючих топіків у кластері (перші 10). Серед них присутні як загальні топіки (наприклад, iot\_alerts), так і топіки, створені іншими користувачами.
4. **Обмежений вивід**
   * В кінці виводу зазначено, що ще 246 топіків не показані:  
     ...and 246 more topics not displayed.
5. **Закриття з'єднання**
   * Після завершення операцій підключення до KafkaAdminClient було успішно закрито:  
     🔒 KafkaAdminClient connection closed.
6. Cкрин генерації даних сенсорів та відправки даних в building\_sensors з демонстрацією двох (або більше) одночасних роботи двох запусків програми.

Деталі:

1. **Генерація даних**
   * Кожна програма генерує дані з ідентифікатором сенсора (**Sensor ID**), температурою (**Temperature**), вологістю (**Humidity**) та часовою міткою (**Timestamp**).
   * Для кожного сенсора створюються унікальні значення, які включають:
     + sensor\_id (унікальний ідентифікатор сенсора),
     + температуру у градусах Цельсія,
     + вологість у відсотках,
     + часову мітку формату YYYY-MM-DDTHH:MM:SS.
2. **Відправка даних до Kafka**
   * Згенеровані дані перетворюються в JSON-формат і відправляються до топіка **building\_sensors**.
   * Для кожного відправлення у виводі відображено:
     + Повідомлення, що було надіслано (Message sent),
     + Назву топіка (Data sent to Kafka topic building\_sensors).
3. **Одночасний запуск**
   * Ліва і права частини скриншота демонструють незалежні запуски двох екземплярів програми.
   * Обидва процеси успішно працюють одночасно, генеруючи унікальні дані та відправляючи їх до одного топіка.
4. **Унікальність даних**
   * Кожен запуск генерує окремі записи даних, які не дублюються між собою. Наприклад:
     + Ліва програма: Sensor ID: 1add87bb-..., Temperature: 32.60°C.
     + Права програма: Sensor ID: e3aac1b9-..., Temperature: 34.05°C.
5. Скрин отримання даних та фільтрації саме тих даних, що будуть далі використані та з демонстрацією того, що відфільтровані дані були послані у відповідні топіки.

Скріншот демонструє процес обробки даних із сенсорів, включаючи перевірку температури й вологості на відповідність пороговим значенням. У разі перевищення порогу генеруються тривожні повідомлення, які надсилаються у відповідні топіки, такі як temperature\_alerts або humidity\_alerts. Усі відфільтровані дані успішно відправляються за призначенням.

**4. Скрин з результатом запису відфільтрованих даних**

Скріншот демонструє результати обробки даних, де вказано, які сенсори перевищили порогові значення температури, і для них було отримано відповідні сповіщення. Для інших сенсорів повідомлень про перевищення не було зафіксовано, про що свідчать рядки з текстом "No alerts received».

**5. Скрин одночасної генерацї декількох завдань**