```
Task 1.b Event Producer 2
        hotspot_AQUA_streaming.csv
       from json import dumps
       from pprint import pprint
       from pymongo import MongoClient
       from datetime import datetime, timedelta, date
      from datetime import datetime
from dateutil.parser import parse
       from time import sleep
       from <mark>kafka</mark> import KafkaProducer
      FILENAME = "hotspot_AQUA_streaming.csv"
       TOPIC = "assignment"
       PORT_NUMBER = 9092
       PRODUCER_FIELD = "producer"
       STREAM_DATA_LIM = 5000
      DEFAULT_LATEST_CLIMATE_DATE = datetime(2021, 12, 31)
      client = MongoClient()
      db = client.fit3182_assignment_db
       climate = db climate
      climate_streaming = db.climate_streaming
                          (lst): return random choice(lst)
          def has lead or end space(string):
              return string.strip() != string
                                    (text, start = 0):
              stripped = text.strip()
              res = ""
              i = start
               while i < len(stripped) and (stripped[i] not in ['_', '.'] ):</pre>
                   res += stripped[i]
                  i += 1
               return res, i
          def first_word(text):
               first, _ = Utils.word_before_separator(text)
               return first
          def producer_info(filename, limit=-1, lower=False):
               first, first_end = Utils_word_before_separator(filename)
               second, _ = Utils.word_before_separator(filename, first_end + 1)
               if lower:
                   first, second = first.lower(), second.lower()
               <u>if</u> limit != -1:
                  return f'{first[:limit]}_{second[:limit]}
               return f'{first}_{second}
                          (json_list):
              max_date = json_list[0]["date"]
               for dict_obj in json_list:
                   if dict_obj["date"] > max_date:
                      max_date = dict_obj["date"]
              return max_date
          def new_date(latest_climate_date, unit_time_to_add, day_factor):
               days_offset = unit_time_to_add * day_factor
               new_date = latest_climate_date + timedelta(days=days_offset)
              return new_date
           def curr_date_time(filename): pass
          def date_time_str(date, need_time=False):
              if need_time:
                   date_format = f'%d/%m/%yT%H:%M:%S'
              elif not need_time:
                   date_format = f'' d/\%m/\%y'
               date_str = date.strftime(date_format)
               return date_str
                                 tr(new_date_time, data, has_time=False):
               if not has_time:
                   data["date"] = Utils.date_time_str(new_date_time, need_time=False)
               if has time:
                   data["date"] = Utils.date_time_str(new_date_time, need_time=False)
                   data["datetime"] = Utils.date_time_str(new_date_time, need_time=True)
              return data
In []: def get_json_list(filename):
           first_word = Utils.first_word(filename)
           df = pd.read_csv(filename, encoding = 'ISO-8859-1')
           records = df.to_json(orient="records", date_format="iso")
          json_list = json_loads(records)
           for i in range(len(json_list)):
              doc = json_list[i]
               doc[f'{first_word}_id'] = i + 1
           producer_inf = Utils producer_info(filename)
           for dict_obj in json_list:
               dict_obj[PRODUCER_FIELD] = producer inf
           for dict_row in json_list:
               keys = dict_row keys()
               keys = list(keys)
               for key in keys:
                   If Utils has_lead_or_end_space(key):
                       stripped_key = key.strip()
                       dict_row[stripped_key] = dict_row[key].strip()
                       del dict_row[key]
          return json_list
     DAY_FACTOR = 0.2
       def publish_message(producer_instance, topic_name, key, data):
               key_bytes = bytes(key, encoding='utf-8')
               producer_instance.send(topic_name, key=key_bytes, value=data)
              producer_instance flush()
              print('Message published successfully. Data: ' + str(data))
              print(str(ex))
       def connect_kafka_producer(port_number):
          _producer = Mone
               _producer = KafkaProducer(bootstrap_servers=[f'localhost:{port_number}'],
                                         value_serializer=lambda x:dumps(x).encode('ascii'),
                                         api_version=(0, 10))
              print(str(ex))
              return _producer
           streaming_json_list = get_json_list(FILENAME)
           producer_inf = Utils.producer_info(FILENAME)
           producer = connect_kafka_producer(PORT_NUMBER)
           climate_historic_data = list(climate.find({}))
           climate_streaming_data = list(climate_streaming.find({}))
          if len(climate_streaming_data) == 0 and len(climate_historic_data) >= 1:
               latest = Utils.latest_date(climate_historic_data)
          elif len(climate_streaming_data) > 0:
               latest = Utils.latest_date(climate_streaming_data)
          elif len(climate_streaming_data) == 0:
               latest = DEFAULT_LATEST_CLIMATE_DATE
               raise Exception("error when getting latest")
          for ctr in range(len(streaming_json_list)):
               chosen_data = Utils.random_elem(streaming_json_list) # randomly chooses a data w
               new_date = Utils.new_date(latest, ctr + 1, DAY_FACTOR)
               appended_data = Utils.append_date_time_str(new_date, chosen_data, has_time=True)
               print(f"ctr: {ctr}")
               publish_message(producer, TOPIC, producer_inf, appended_data)
               sleep(DAY_FACTOR*10) # choose the interval to send data
```