 **Extract data from API**: Use the requests library to make an API call.

 **Process data in Dataflow**: Use Apache Beam to process the data.

 **Write data to GCS**: Store the API data into GCS as a file.

1)Error :'PBegin' object has no attribute 'windowing'

You can’t use ParDo directly at the start because it needs a **PCollection** to operate on. Here’s a simple breakdown:

1. **Starting Point**:
   * ParDo is a transformation that processes elements in a PCollection. If you don't have a PCollection to start with, you can't use ParDo.
2. **What beam.Create([None]) Does**:
   * Using beam.Create([None]) creates an initial PCollection, even if it’s just a placeholder. This allows the pipeline to start running.
3. **Next Steps**:
   * After the initial PCollection is created, you can then use ParDo to process that data, like fetching from an API.

2) **json.dumps(data)**: Changes the dictionary into a JSON string for saving or sending.

Without json.dumps(data), you might face errors, compatibility issues, and your data might not be usable in other systems.

**3)Why divide by 1000.0?**

We use 1000.0 instead of 1000 to ensure the division result is a floating-point number, which provides more precise conversion and prevents potential issues with integer division in Python.

For example:

* Milliseconds: 1727170632290
* Converted to seconds: 1727170632290 / 1000.0 = 1727170632.290 seconds

4) import re

# Assuming 'place\_str' contains the string value from which we want to extract a substring

place\_str = "17 km NW of Boonville of xyz"

# Use regex to extract substring after the first occurrence of 'of'

match = re.search(r'of\s(.+)', place\_str)

if match:

# Get the part after 'of'

area = match.group(1)

print("Extracted Area:", area)

else:

# If no match found, set area to None

area = None

print("No 'of' found in place")

### Explanation of the Regular Expression

#### **Step-by-Step Breakdown:**

In Python, the prefix **r** before a string denotes a **raw string**. A raw string treats backslashes (\) as literal characters rather than as escape characters. This is important when writing regular expressions (regex), where backslashes are commonly used.

1. **r'of\s(.+)'**: This is the regular expression pattern used to search for a substring starting after the word **"of"**.
   * **of**: Matches the exact word **"of"**.
   * **\s**: Matches any whitespace character (e.g., space). This ensures that we match **"of "** (of followed by a space).
   * **(.+)**: The parentheses **()** denote a capture group, meaning we want to extract whatever matches this part. **.+** matches **one or more characters** (anything after "of").
2. **re.search()**: This function looks for a match of the pattern within the place\_str. If a match is found, it returns a match object; otherwise, it returns None.
3. **match.group(1)**: If the match object exists, **group(1)** returns the first capture group, which is everything after the first occurrence of **"of "**. The capture group is defined by the parentheses () in the regular expression.
4. **else: area = None**: If no match is found, the area is set to None.

5)

insert\_date = datetime.now().timestamp()  
insert\_date\_values = datetime.utcfromtimestamp(insert\_date).strftime('%Y-%m-%d %H:%M:%S')

Step 1: Get Current Timestamp

insert\_date = datetime.now().timestamp()

**1)datetime.now()**:

* This retrieves the current local date and time.

from datetime import datetime

**current\_datetime = datetime.now() print(current\_datetime) # Output: 2024-10-25 10:15:30.123456 (example output)**

2) **.timestamp()**:

* This converts the datetime object to a Unix timestamp.

**insert\_date = current\_datetime.timestamp() print(insert\_date) # Output: 1729842930.123456 (example output)**

Step2: Convert Timestamp to UTC Datetime

**insert\_date\_values = datetime.utcfromtimestamp(insert\_date).strftime('%Y-%m-%d %H:%M:%S')**

This takes the Unix timestamp and converts it back to a datetime object, treating the timestamp as UTC.

**utc\_datetime = datetime.utcfromtimestamp(insert\_date)**

**print(utc\_datetime) # Output: 2024-10-25 10:15:30.123456 (example output)**

**insert\_date\_values = utc\_datetime.strftime('%Y-%m-%d %H:%M:%S') print(insert\_date\_values) # Output: '2024-10-25 10:15:30'**