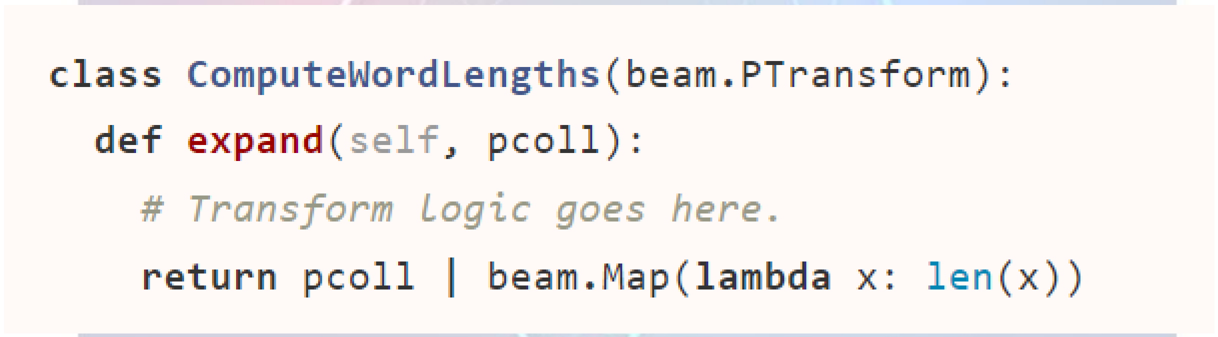
Tutorial 3g - Composite Transformation in Apache Beam:

Composite Transform:

- Transforms can have a nested structure, where a complex transform performs multiple simpler transforms (such as more than one ParDo, Combine, GroupByKey, or even other composite transforms). These transforms are called composite transforms.
- **Nesting multiple transforms inside a single composite transform** can make your code more modular and easier to understand.

Creating a composite transform:

- To create your own composite transform, create a subclass of the PTransform class and override the expand method to specify the actual processing logic.
- The transforms can include core transforms, composite transforms, or the transforms included in the Beam SDK libraries.
- The following code sample shows how to declare a PTransform that accepts a PCollection of Strings for input, and outputs a PCollection of Integers:



```python
class ComputeWordLengths(beam.PTransform):
  def expand(self, pcoll):
    # Transform logic goes here.
    return pcoll | beam.Map(lambda x: len(x))
```

- The expand method is where you add the processing logic for the PTransform. Your override of expand must accept the appropriate type of input PCollection as a parameter, and specify the output PCollection as the return value.
- You can include as many transforms as you want. These transforms can include core transforms, composite transforms, or the transforms included in the Beam SDK libraries.
- Your composite transform's parameters and return value must match the initial input type and final return type for the entire transform, even if the transform's intermediate data changes type multiple times.

Resources:

- https://beam.apache.org/documentation/programming-guide/#composite-transforms
- https://beam.apache.org/releases/pydoc/2.36.0/apache_beam.transforms.html