Assignment Outline:

Task 1: Custom Transformation Using FunctionTransformer

In this task, create a log transformer and a ratio transformer using Scikit-Learn's FunctionTransformer. Apply these transformers to a dataset and observe the output.

- **Step 1:** Import a dataset (you can use housing.csv dataset or use a built-in dataset like California housing).
- **Step 2:** Create a **log transformer** for transforming numerical features with heavy-tailed distributions.
- **Step 3:** Create a **ratio transformer** that computes the ratio of two columns from the dataset.
- **Step 4:** Apply these transformers to the dataset and check the results.

Hints:

- Use FunctionTransformer.
- Example: FunctionTransformer(np.log, inverse_func=np.exp).

Task 2: Building Custom Transformer Class

Students will create their own custom transformer by subclassing BaseEstimator and TransformerMixin from Scikit-Learn. The transformer will standardize a specific feature in the dataset by removing its mean and scaling by its standard deviation.

- **Step 1:** Create a class StandardScalerClone that implements fit, transform, and fit_transform methods.
- **Step 2:** Add input validation to the fit method using check_array from sklearn.utils.validation.
- **Step 3:** Apply this custom transformer to any numerical column of the dataset.

Requirements:

- The class should inherit from BaseEstimator and TransformerMixin.
- Implement input validation using check_array in the fit method.

Task 3: Clustering-Based Custom Transformer

Create a custom transformer that uses **K-Means clustering** to group data points and computes the similarity of each point to the cluster centers using the **RBF kernel**.

- **Step 1:** Implement a class ClusterSimilarity that:
  - Uses KMeans clustering in the fit method.
  - Computes similarities to the cluster centers using rbf_kernel in the transform method.
- **Step 2:** Apply this custom transformer to the latitude and longitude columns of the dataset.

Instructions:

- Use KMeans from sklearn.cluster and rbf_kernel from sklearn.metrics.pairwise.
- Implement methods for both fitting and transforming the data.

Task 4: Pipelines and ColumnTransformers

Combine the transformers in a **pipeline** and apply them to both numerical and categorical features of the dataset.

- **Step 1:** Create a numerical pipeline that:
  - Handles missing values using SimpleImputer (median strategy).
  - Applies standardization using the custom StandardScalerClone class.
- **Step 2:** Create a categorical pipeline that:
  - Imputes missing categorical values using SimpleImputer (most frequent strategy).
  - Encodes the categories using OneHotEncoder.
- **Step 3:** Create a ColumnTransformer that:
  - Applies the numerical pipeline to numerical columns.
  - Applies the categorical pipeline to categorical columns.
- **Step 4:** Apply the full pipeline to the dataset and output the transformed data.

You can take hints from the Hands on Machine Learning book but don’t copy paste.