Libraries:

- `torch`: Deep learning framework https://pytorch.org/docs/stable/index.html
- `transformers`: Library for working with transformers https://huggingface.co/docs/transformers/en/index
- `datasets`: Library for loading and processing datasets [Hugging Face datasets ON huggingface.co]

Code Structure:

1. Import Libraries:
   - Imports necessary libraries for transformers, data processing, and training.
2. Load Dataset:
   - Uses `load_dataset` from `datasets` to load the "Customer-Support-Responses" dataset from the Hugging Face Hub.
3. Train-Test Split:
   - Splits the loaded dataset into 80% training and 20% test sets using `train_test_split`.
4. Load Tokenizer and Model:
   - Defines the model name (e.g., "t5-small") and loads the corresponding tokenizer and model from the Hugging Face model hub using `T5Tokenizer.from_pretrained` and `T5ForConditionalGeneration.from_pretrained`. Refer to https://huggingface.co/docs/transformers/en/index for available models.
5. Preprocess Data:
   - Defines a function `preprocess_function` that takes examples from the dataset and performs the following:
     - Adds "summarize: " prefix to the query for context.
     - Tokenizes the query and response with truncation (maximum length) using the loaded tokenizer.
     - Sets the labels as target input IDs for the decoder.
   - Applies the `preprocess_function` to the entire dataset using `map` with batch processing for efficiency.
6. Data Collator:
   - Creates a `DataCollatorForSeq2Seq` instance to handle padding and batching during training.
7. Training Arguments:
   - Defines training arguments using `TrainingArguments` from transformers:
     - `output_dir`: Path to store training outputs.
     - `evaluation_strategy`: How often to evaluate the model during training (e.g., "epoch").
     - `save_strategy`: How often to save model checkpoints (e.g., "epoch").
     - `learning_rate`: Optimizer learning rate.
     - `per_device_train_batch_size`: Training batch size per device.
     - `per_device_eval_batch_size`: Evaluation batch size per device.
     - `num_train_epochs`: Number of training epochs.
     - `weight_decay`: Weight decay for regularization.
     - `save_total_limit`: Maximum number of checkpoints to save.

        - **`load_best_model_at_end`: Load the best model based on evaluation metric.**

8. **Create Trainer:**
    - **Initializes a `Trainer` instance from transformers to manage the training process. It takes the model, training arguments, datasets (train/eval), and data collator as input.**
9. **Train the Model:**
    - **Call the `train` method on the trainer to start the training process.**
10. **Evaluate the Model:**

- **Calls the `evaluate` method on the trainer to assess the model's performance on the test set. Prints the evaluation results.**