```
!pip install -q datasets
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 542.0/542.0 kB 7.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 7.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 7.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 8.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 388.9/388.9 kB 12.7 MB/s eta 0:00:00
```

```
!pip install -q transformers
```

```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from transformers import BartTokenizer, BartForConditionalGeneration
from datasets import load_dataset
import torch
from torch.utils.data import DataLoader
from torch.nn.parallel import DistributedDataParallel as DDP

# Load dataset
dataset = load_dataset("code_x_glue_ct_code_to_text","python")
dataset_subset = dataset["train"].select(range(1000))

# Load tokenizer and model
tokenizer = BartTokenizer.from_pretrained("facebook/bart-base")
model = BartForConditionalGeneration.from_pretrained("facebook/bart-base")

# Define optimizer and loss function
optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
criterion = torch.nn.CrossEntropyLoss()

# Training loop
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
model.train()

# Define custom dataset class
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self, dataset):
        self.dataset = dataset

    def __getitem__(self, index):
        example = self.dataset[index]
        code = example["code"]
        summary = example["docstring"]

        # Tokenize inputs and targets
        inputs = tokenizer(f"code: {code}", padding="max_length", truncation=True, max_length=512, return_tensors="pt")
        input_ids = inputs.input_ids.squeeze(0)
        attention_mask = inputs.attention_mask.squeeze(0)
        target = tokenizer(summary, padding="max_length", truncation=True, max_length=128, return_tensors="pt")
        target_ids = target.input_ids.squeeze(0)

        return input_ids, attention_mask, target_ids

    def __len__(self):
        return len(self.dataset)

# Create DataLoader
custom_dataset = CustomDataset(dataset_subset)
train_dataloader = DataLoader(custom_dataset, batch_size=8, shuffle=True, num_workers=4)

# Training loop
for step, (input_ids, attention_mask, target_ids) in enumerate(train_dataloader):
    input_ids, attention_mask, target_ids = input_ids.to(device), attention_mask.to(device), target_ids.to(device)

    # Forward pass
    optimizer.zero_grad()
    outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=target_ids)
    logits = outputs.logits

    # Calculate loss
    loss = criterion(logits.view(-1, logits.shape[-1]), target_ids.view(-1))

    # Backward pass and optimization step
    loss.backward()
```

```
        optimizer.step()

        # Print loss every 100 iterations
        if (step + 1) % 100 == 0:
            print(f"Iteration {step + 1}, Loss: {loss.item()}")

save_directory = "/content/drive/MyDrive/finetuned_model"

# Save the finetuned model
model.save_pretrained(save_directory)
print("Training completed!")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: U
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings ta
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access p
  warnings.warn(
Downloading readme: 100% 26.7k/26.7k [00:00<00:00, 1.79MB/s]
Downloading data: 100% 144M/144M [00:00<00:00, 199MB/s]
Downloading data: 100% 147M/147M [00:03<00:00, 55.5MB/s]
Downloading data: 100% 16.7M/16.7M [00:01<00:00, 9.12MB/s]
Downloading data: 100% 18.0M/18.0M [00:01<00:00, 15.6MB/s]
Generating train split: 100% 251820/251820 [00:05<00:00, 25927.81 examples/s]
Generating validation split: 100% 13914/13914 [00:00<00:00, 34282.26 examples/s]
Generating test split: 100% 14918/14918 [00:00<00:00, 48100.07 examples/s]
vocab.json: 100% 899k/899k [00:00<00:00, 12.4MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 6.31MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 6.88MB/s]
config.json: 100% 1.72k/1.72k [00:00<00:00, 134kB/s]
model.safetensors: 100% 558M/558M [00:02<00:00, 199MB/s]
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558: U
  warnings.warn(_create_warning_msg(
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork
  self.pid = os.fork()
Iteration 100, Loss: 0.25369447469711304
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork
  self.pid = os.fork()
Some non-default generation parameters are set in the model config. These sho
Non-default generation parameters: {'early_stopping': True, 'num_beams': 4, '
Training completed!
```