Developing a Chatbot using Python with ANN Framework

To Build a chatbot using a custom Artificial Neural Network (ANN) framework can be complex and typically involves natural language processing techniques.
To create a basic chatbot using a feedforward neural network with a popular deep learning framework, PyTorch.

Here's a step-by-step guide and my code:

- ☐ **Step 1:** Installation
  To Install PyTorch and other necessary libraries:

  pip install torch
  pip install nltk

- ☐ **Step 2:** Preprocessing and Data
  To Prepare a dataset of input-output pairs for training a chatbot.
  To use a small dataset:

  conversations = [
  ("Hi", "Hello! How can I assist you today?"),
  ("What's the weather like today?", "I'm sorry, I can't provide weather information."),
  ("Tell me a joke", "Sure! Why did the chicken cross the road? To get to the other side."),

```
    ("Goodbye", "Goodbye! Have a great day."),
]
```

☐ **Step 3:** Data Preprocessing

To Tokenize the text and convert it to lowercase:

```
import nltk
from nltk.tokenize import word_tokenize
  nltk.download('punkt')
  def tokenize(text):
  return word_tokenize(text.lower())
```

☐ **Step 4:** To Build a Simple Feedforward Neural Network with PyTorch

To Define and train a basic feedforward neural network:

```
import torch
import torch.nn as nn
import torch.optim as optim

# To Define the neural network model
class Chatbot(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(Chatbot, self).__init__()
        self.embedding = nn.Embedding(input_size,
hidden_size)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        embedded = self.embedding(x)
```

```
        output = self.fc(embedded)
        return output

# To Set hyperparameters
input_size = vocab_size
hidden_size = 128
output_size = vocab_size
learning_rate = 0.001
n_epochs = 50

# To Initialize and train the model
model = Chatbot(input_size, hidden_size, output_size)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
lr=learning_rate)

# To Train the model
for epoch in range(n_epochs):
    for conversation in conversations:
        X_train = [tokenize(conversation[0])]
        y_train = [tokenize(conversation[1])]
        X_train = torch.tensor([tokens_to_indices(X_train,
vocab)])
        y_train = torch.tensor([tokens_to_indices(y_train,
vocab)])

        optimizer.zero_grad()
        output = model(X_train)
        loss = criterion(output.view(-1, output_size),
y_train.view(-1))
        loss.backward()
```

```
        optimizer.step()
```

☐ **Step 5:** Chatbot Interaction

To Create a function to interact with a chatbot using the trained model:

```
def chat_with_bot(user_input):
    input_tokens = tokenize(user_input)
    input_indices =
torch.tensor([tokens_to_indices([input_tokens], vocab)])
    output = model(input_indices)
    response_indices = torch.argmax(output, dim=2)
    response_tokens =
indices_to_tokens(response_indices[0], vocab)
    return " ".join(response_tokens)
```

☐ **Step 6:** Interact with a Chatbot

To interact with a chatbot by providing user input and receiving responses:

```
while True:
    user_input = input("You: ")
    if user_input.lower() == 'bye':
        print("Bot: Goodbye!")
        break
    response = chat_with_bot(user_input)
    print("Bot:", response)
```