# Fixing OpenAI Class Naming Conflict

## Issue Explanation:

The issue occurs because the class being defined is named `OpenAI`, which conflicts with the imported `OpenAI` class from the `openai` module. This creates ambiguity, causing Python to be unsure which `OpenAI` class is being referenced.

## How to Fix:

1. Rename the custom class from `OpenAI` to `CustomOpenAI` to prevent conflicts.

2. Ensure the class correctly extends `OpenAI` while maintaining its custom functionality.

3. Use `CustomOpenAI` instead of `OpenAI` when initializing the client.

## Corrected Python Code:

from openai import OpenAI  
  
# Fixing the class name conflict by renaming it to CustomOpenAI  
class CustomOpenAI(OpenAI):  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.base\_url = "http://learn-mux-cluster-proxy.learn-experience-platform.svc.cluster.local/openai-proxy/v1"  
 self.max\_retries = 0  
  
# Create an instance of the corrected class  
client = CustomOpenAI(api\_key="<OPENAI\_API\_TOKEN>")  
  
# Example request to ensure it works correctly  
response = client.chat.completions.create(  
 model="gpt-4o-mini",  
 messages=[{"role": "user", "content": "Who developed ChatGPT?"}]  
)  
  
print(response.choices[0].message.content)

## Explanation of the Fix:

1. Renaming the class to `CustomOpenAI` prevents Python from confusing it with the imported `OpenAI` class.  
2. The corrected class properly extends `OpenAI` without overriding the existing import.  
3. This approach allows the API request to function correctly without naming conflicts.