**Reflective Journal**

**Overview**

Working with these Python scripts has given me a hands-on understanding of how to use the OpenAI API, handle simple string algorithms, and automate code generation using AI. Each file focused on a unique task, allowing me to explore chatbot interactions, prompt-based code generation, and basic algorithm testing. This was a solid practice of both AI integration and foundational Python logic.

**What I Learned**

Through Chatbot\_Conversation.py and ChatGPT\_OpenAI.py, I learned how to structure conversations with an AI model using the OpenAI API. These files demonstrated the difference between single-turn (ChatGPT\_OpenAI.py) and multi-turn conversations (Chatbot\_Conversation.py), giving me insight into how context can be maintained in chat-based applications.

The generate\_code.py file introduced me to the concept of using AI not just to answer questions, but to create executable code. Seeing how a well-crafted prompt can generate a function to check if a string is a palindrome was pretty interesting. It also showed me how AI can assist in automating repetitive coding tasks or serve as a learning assistant.

Lastly, pali.py brought it back to basics. This simple palindrome checker reinforced the importance of string manipulation in Python. It reminded me that while AI is powerful, knowing the core logic behind basic problems is still necessary.

**Challenges**

One minor challenge I ran into was understanding the different ways to call the OpenAI API. Switching between methods like openai.ChatCompletion.create() and the OpenAI client instance was a bit confusing at first.

**Key Takeaways**

* AI can be integrated into Python scripts easily with the right API and structure.
* Multi-turn conversations require careful handling of the conversation list to maintain context.
* Prompt engineering is vital—how you ask the model to do something directly affects the output.
* Simple algorithms like palindrome checking are great practice for clean, efficient code.

**Future Goals**

Moving forward, I want to expand on this by creating more interactive AI-based projects. For example, building a full chatbot with memory, or an AI-powered code tutor that can explain the code it generates.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.