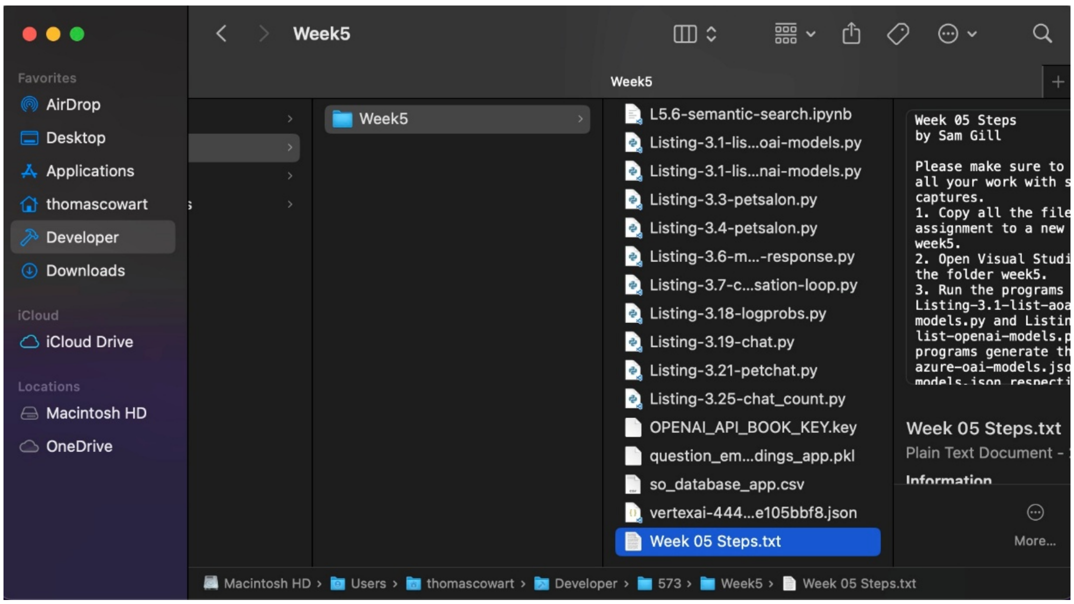
Thomas Cowart

Prof. Gill

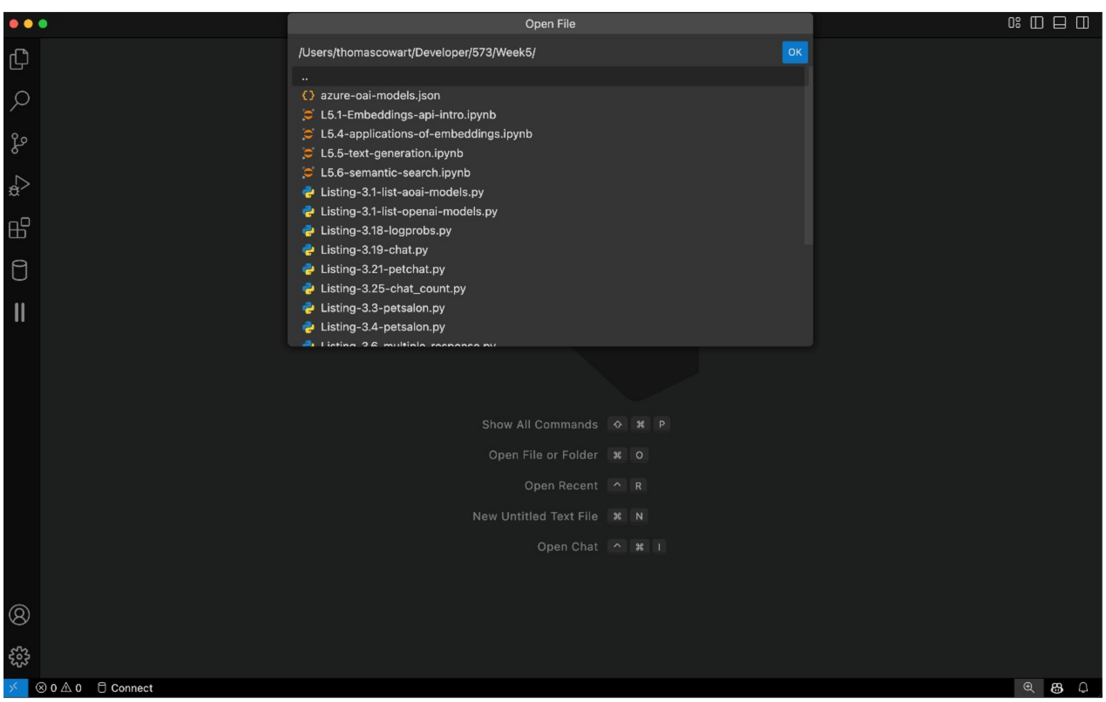
ISYS 573

Week 05 HW

1)



2)



3)

oai-models.json — Edited

SyncPage[Model](data=[Model(id='gpt-4.5-preview', created=1740623059, object='model', owned_by='system'), Model(id='omni-moderation-2024-09-26', created=1732734466, object='model', owned_by='system'), Model(id='gpt-4.5-preview-2025-02-27', created=1740623304, object='model', owned_by='system'), Model(id='gpt-4o-mini-audio-preview-2024-12-17', created=1734115920, object='model', owned_by='system'), Model(id='dall-e-3', created=1698785189, object='model', owned_by='system'), Model(id='dall-e-2', created=1698798177, object='model', owned_by='system'), Model(id='gpt-4o-audio-preview-2024-10-01', created=1727389042, object='model', owned_by='system'), Model(id='gpt-4o-audio-preview', created=1727460443, object='model', owned_by='system'), Model(id='gpt-4o-mini-realtime-preview-2024-12-17', created=1734112601, object='model', owned_by='system'), Model(id='gpt-4o-2024-11-20', created=1739331543, object='model', owned_by='system'), Model(id='gpt-4o-mini-realtime-preview', created=1734387380, object='model', owned_by='system'), Model(id='o1-mini-2024-09-12', created=1725648979, object='model', owned_by='system'), Model(id='o1-preview-2024-09-12', created=1725648865, object='model', owned_by='system'), Model(id='o1-mini', created=1725649008, object='model', owned_by='system'), Model(id='o1-preview', created=1725648897, object='model', owned_by='system'), Model(id='gpt-4o-mini-audio-preview', created=1734387424, object='model', owned_by='system'), Model(id='whisper-1', created=1677532384, object='model', owned_by='openai-internal'), Model(id='gpt-4o-2024-05-13', created=1715368132, object='model', owned_by='system'), Model(id='gpt-4o-realtime-preview-2024-10-01', created=1727131766, object='model', owned_by='system'), Model(id='babbage-002', created=1692634615, object='model', owned_by='system'), Model(id='chatgpt-4o-latest', created=1723515131, object='model', owned_by='system'), Model(id='gpt-4-turbo-preview', created=1706037777, object='model', owned_by='system'), Model(id='tts-1-hd-1106', created=1699053533, object='model', owned_by='system'), Model(id='gpt-4o-audio-preview-2024-12-17', created=1734034239, object='model', owned_by='system'), Model(id='gpt-4', created=1687882411, object='model', owned_by='openai'), Model(id='gpt-4-turbo', created=1712361441, object='model', owned_by='system'), Model(id='tts-1-hd', created=1699046015, object='model', owned_by='system'), Model(id='gpt-4o-mini-2024-07-18', created=1721172717, object='model', owned_by='system'), Model(id='gpt-4o-mini', created=1721172741, object='model', owned_by='system'), Model(id='text-embedding-3-large', created=1705953180, object='model', owned_by='system'), Model(id='tts-1', created=1681940951, object='model', owned_by='openai-internal'), Model(id='tts-1-1106', created=1699053241, object='model', owned_by='system'),

azure-oai-models.json — Edited

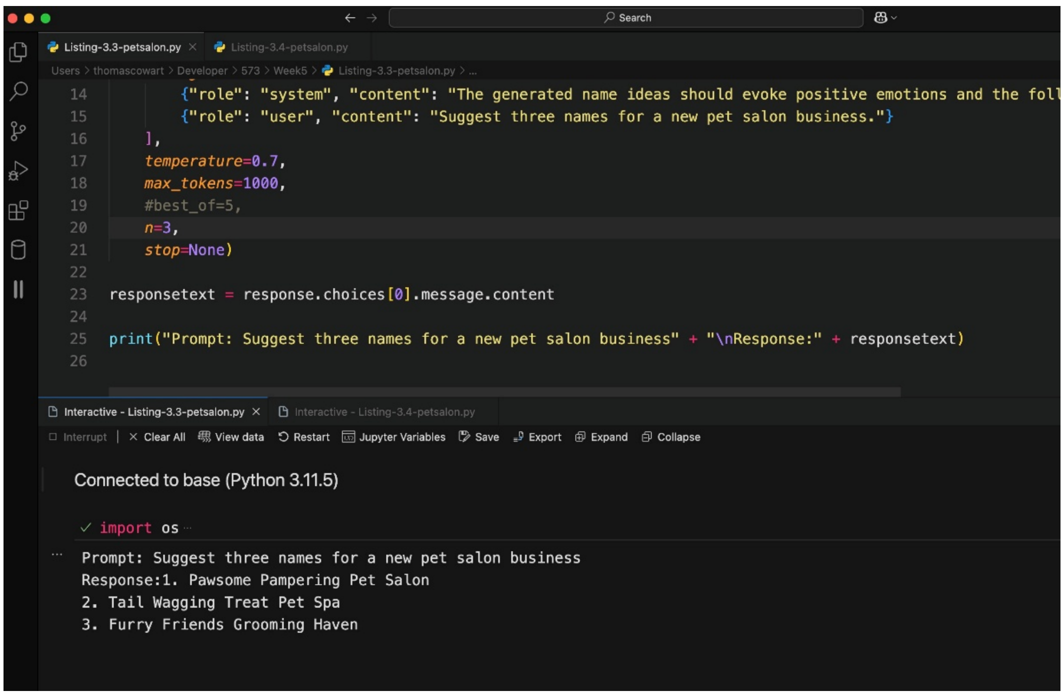
[{"id": "dall-e-3-3.0", "created": null, "object": "model", "owned_by": null}, {"id": "dall-e-2-2.0", "created": null, "object": "model", "owned_by": null}, {"id": "code-cushman-001", "created": null, "object": "model", "owned_by": null}, {"id": "code-cushman-fine-tune-002", "created": null, "object": "model", "owned_by": null}, {"id": "whisper-001", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-0301", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-0613", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-1106", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-0125", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-instruct-0914", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-35-turbo-16k-0613", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-0125-Preview", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-1106-Preview", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-0314", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-0613", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-32k-0314", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-32k-0613", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-vision-preview", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-turbo-2024-04-09", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4-turbo-jp", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-2024-05-13", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-2024-08-06", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-mini-2024-07-18", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-2024-11-20", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-audio-mai", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-realtime-preview", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-mini-realtime-preview-2024-12-17", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-realtime-preview-2024-12-17", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-canvas-2024-09-25", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-audio-preview-2024-10-01", "created": null, "object": "model", "owned_by": null}, {"id": "gpt-4o-audio-preview-2024-12-17", "created": null, "object": "model", "owned_by": null}, {"id":

The two JSON files list available models from different sources: Azure OpenAI (azure-oai-models.json) and OpenAI's own API (oai-models.json).

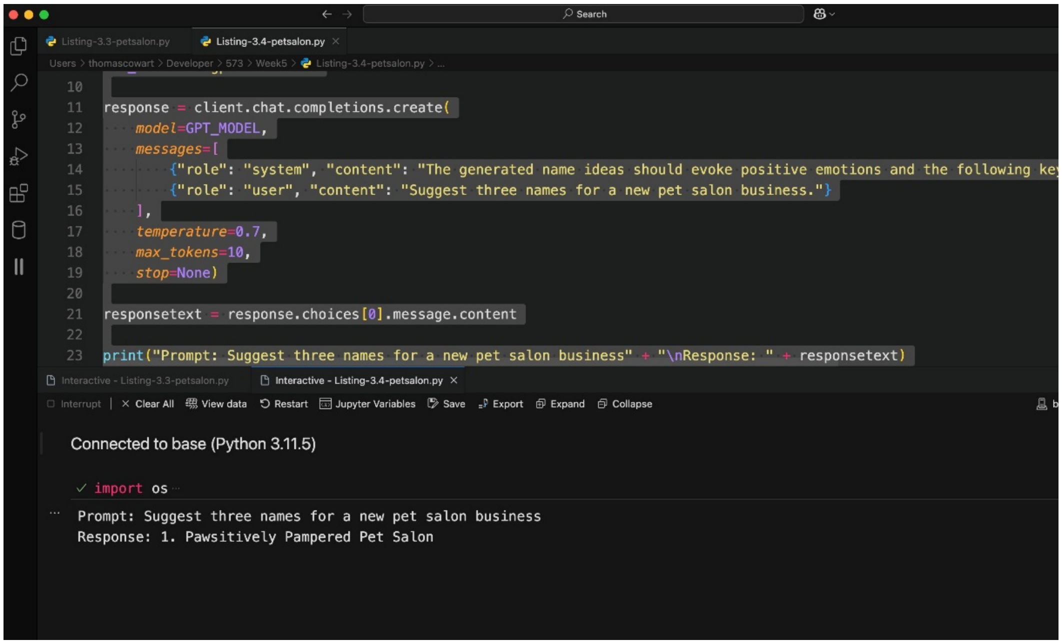
Azure's list includes older models like davinci, curie, and babbage, alongside gpt-4o and dall-e-3, but lacks explicit ownership details. OpenAI's list is more up-to-date, featuring gpt-4.5-preview, real-time variations, and models labeled with specific dates. It also includes ownership tags (system, openai, openai-internal) and models for moderation (omni-moderation) and text-to-speech (tts-1-hd).

Azure seems to have a more stable, enterprise-focused set of models, while OpenAI's API moves faster, testing and releasing updates more frequently.

4)

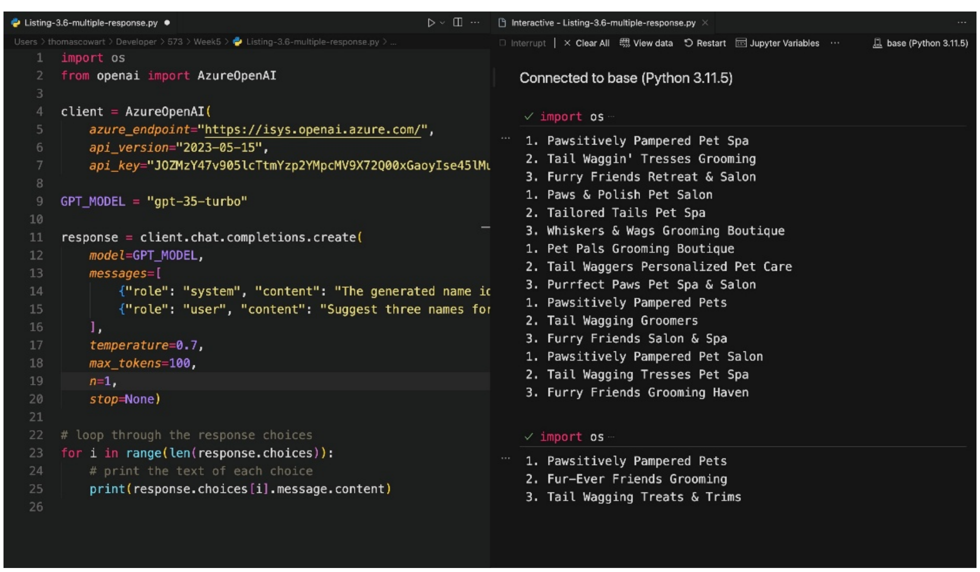


As we can see in the screenshot above, the max number of tokens is set to 1000, which gives ample room for the model to successfully complete its assigned task.



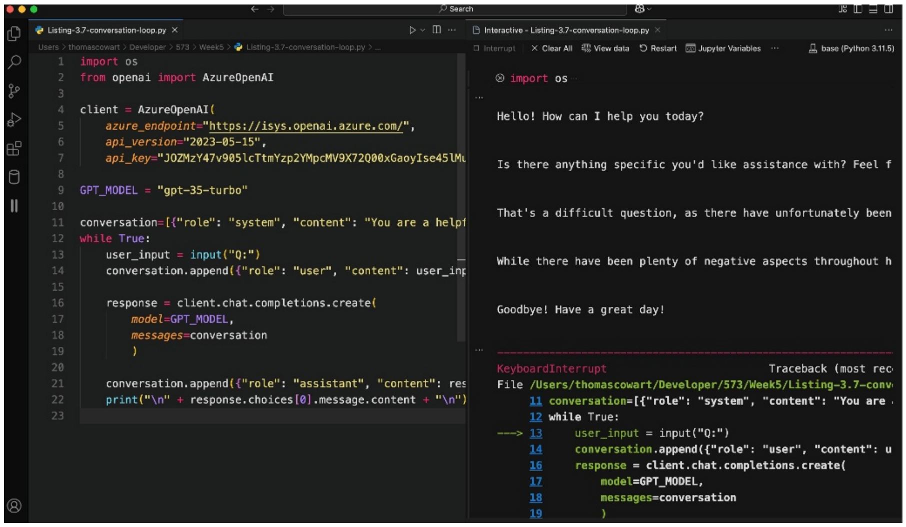
As we can see in the screenshot above, the max number of tokens is set to only 10, which does not given the model enough room to successfully complete its assigned task. Realistically for this task, I would imagine that we would want to set the max number of tokens to a value in-between 10 and 1000 for efficiency sake, while still successfully producing the desired output.

5)



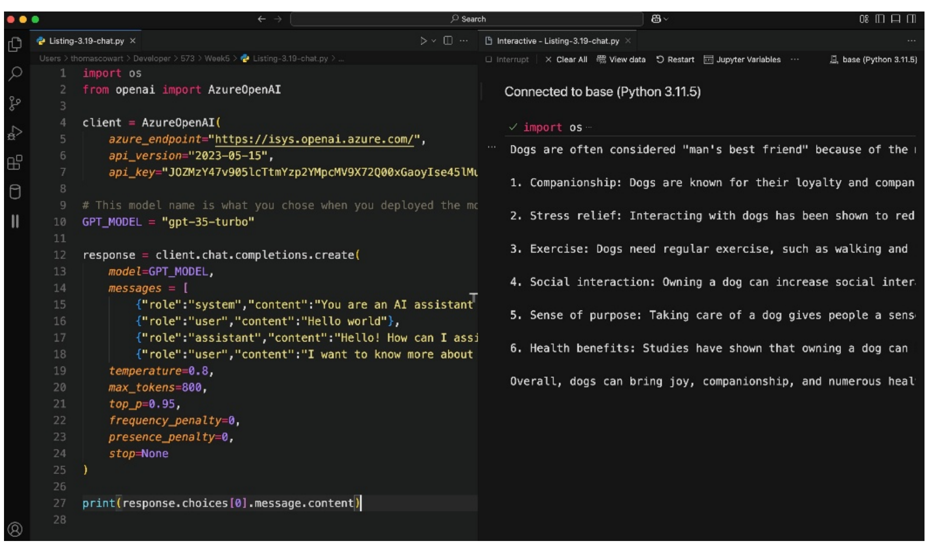
As we can see in the screenshot above, the variable “n” represents the number of times that the prompt is ran. By changing the variable n, we will receive that number or repeated prompts until the maximum number of tokens has been reached.

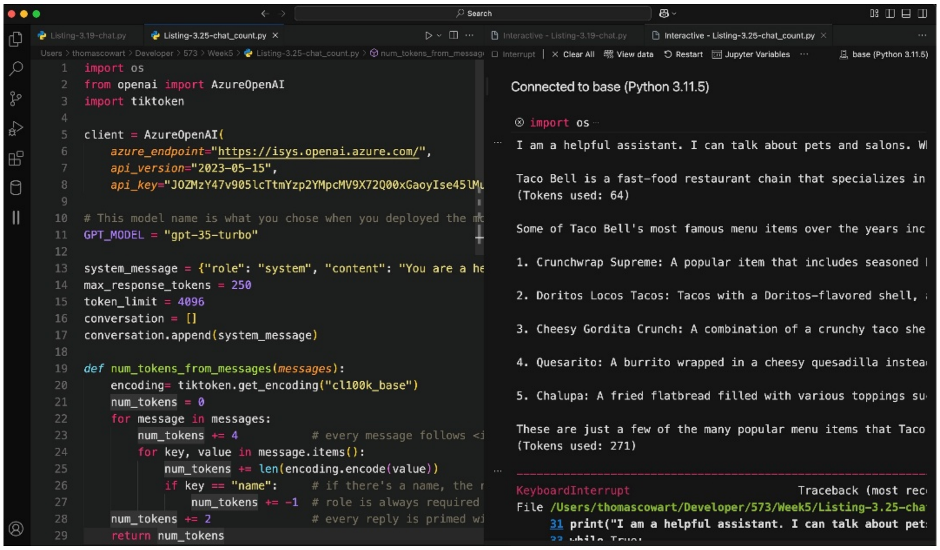
6)



I had a few back-and-forth conversations with the model. I had to interrupt the script to end the session. After several attempts to try to get the model to hallucinate/innovate, my attempts came to no avail.

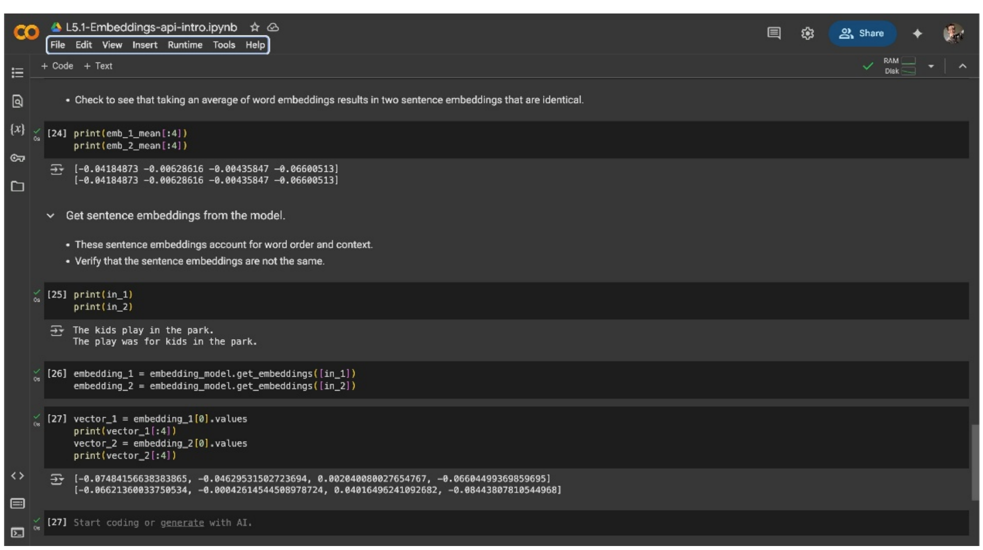
7)





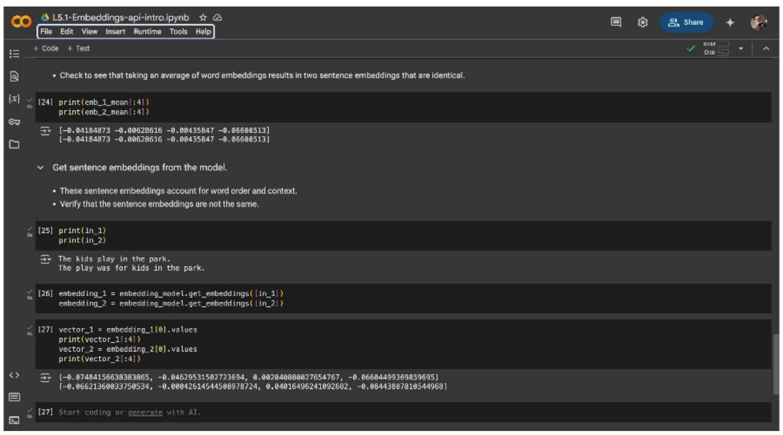
System defines how the AI behaves, user is me, assistant is the AI replying. Max tokens just limits how long the response can be. In 3.19.py, it's set to 800. In 3.25, it's 250, and old messages get deleted to stay under 4096 total tokens. If the response hits the max tokens limit, it'll stop there, and the finish reason will be "max_tokens." You can check token usage with response.usage.total_tokens.

8)



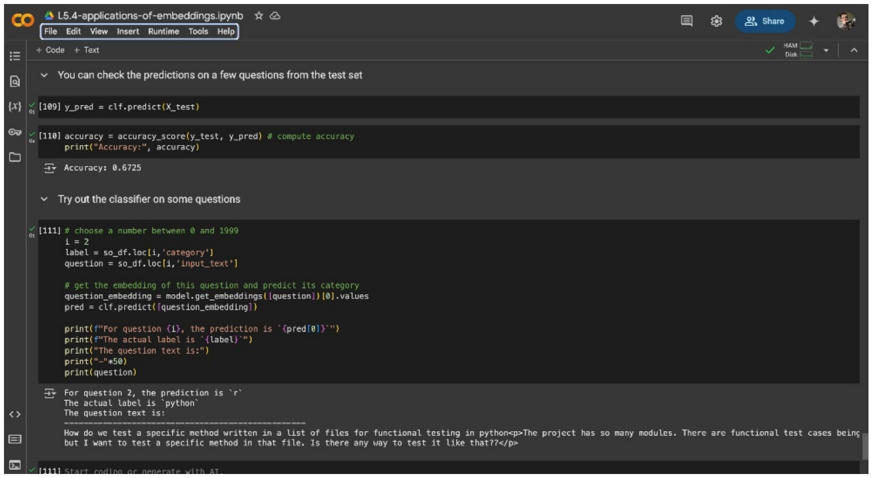
Embeddings turn words or data into numbers so a computer can understand relationships between them. Instead of just treating words as separate things, embeddings map them into a multi-dimensional space where similar things are closer together. For example, in an embedding model, “king” and “queen” would be near each other, while “king” and “banana” would be far apart. It’s like plotting words on a graph, but with way more dimensions. This is useful for search, recommendations, or AI understanding context because it helps find patterns and similarities beyond just matching exact words.

9)



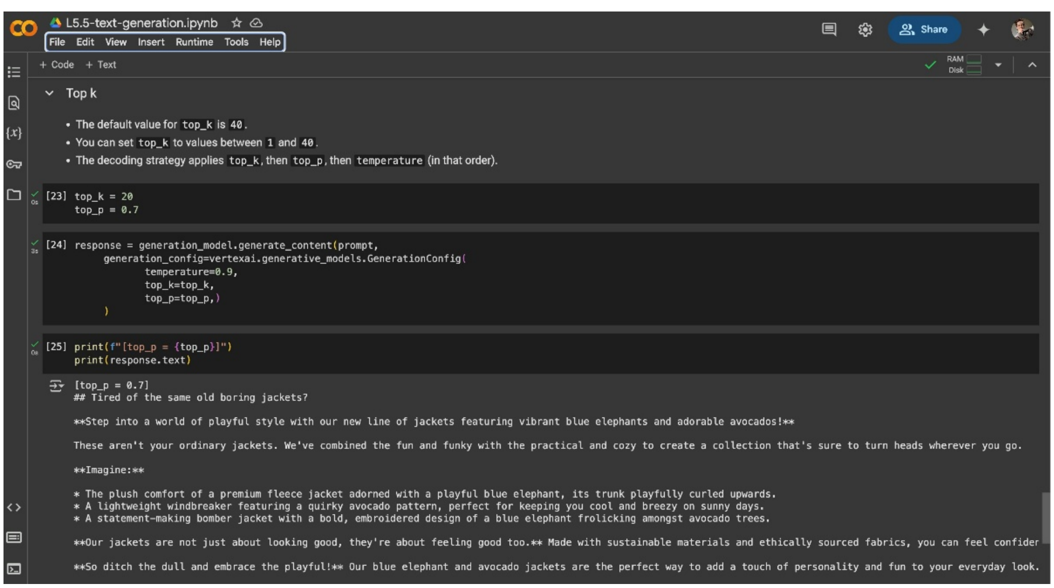
To use embeddings for classification, you first transform your data (like text) into embeddings, which are numerical representations of the data. Then, you can feed those embeddings into a classification model (like a neural network or a decision tree) to predict categories, since the embeddings capture the important relationships between words or data points. To avoid recalculating embeddings every time, you can store them once, and then reuse the precomputed embeddings whenever you need to classify new data. This saves time, since you don’t need to run the embedding model from scratch each time. Essentially, you preprocess your data, save the embeddings, and then only run the classifier on those pre-saved embeddings.

10)



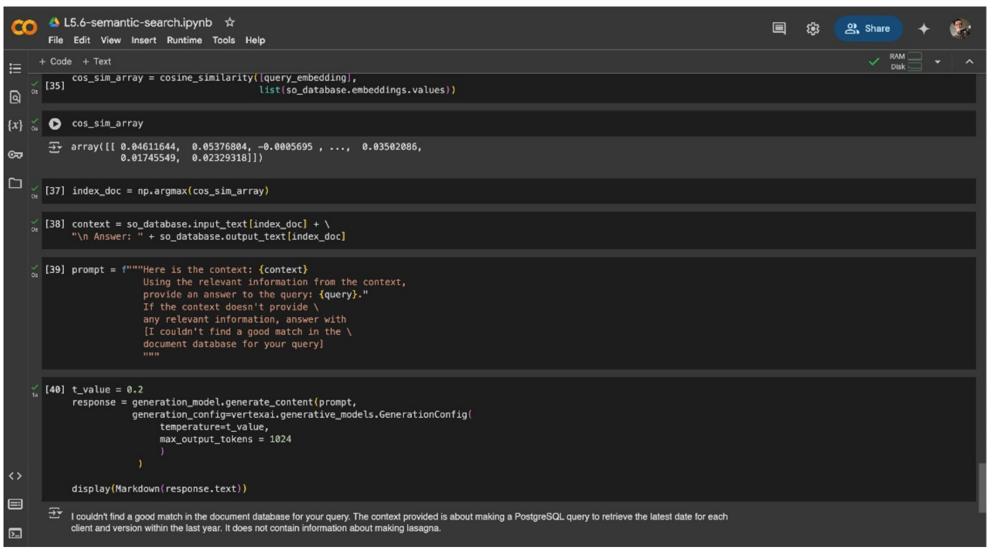
To use embeddings for classification, you first convert your data (like text) into embedding numerical vectors that capture meaning. Then, you feed these vectors into a classification model (like a neural network, SVM, or even KNN) to assign labels based on the patterns in the embedding space. Shortcut: Instead of recalculating embeddings every time, precompute and store them in a database. When new data comes in, look up the closest existing embedding instead of reprocessing everything from scratch. This saves time and computational power.

11)



The script demonstrated several features of Vertex AI prompting. It used text-bison@001 for general text generation, showing how to select a model based on the task. Authentication was handled by loading service account credentials with OAuth scopes to grant access to Vertex AI. The script set PROJECT_ID and REGION (us-central1), which are required for sending API requests. It structured a prompt to generate text from the model, demonstrating how to craft inputs for effective responses. Additionally, it integrated with Google Colab by mounting Google Drive to access credentials, making it easier to run in a cloud-based environment.

12)



Semantic search goes beyond keyword matching by understanding intent and context. It starts with embedding generation, where both queries and documents are converted into vector representations using models like BERT. These embeddings are stored in a vector database, enabling fast similarity searches. When a query is made, it's also embedded and compared to stored vectors using nearest neighbor search to find the closest matches. The results are then ranked and retrieved based on relevance, often incorporating additional factors like metadata or user behavior. This makes search more intuitive, recognizing meaning rather than just exact words.