Using Agentic AI

Definition
An **LLM** is a neural network trained on a massive amount of text data, which predicts the next most likely word in a sequence. When scaled up has the emergent property of being able to complete tasks such as answering questions.

Using `google.genai`

Listing Models

First install `google-generativeai`

```
pip install google-generativeai
```

And list available models.

```
import google.generativeai as genai
import os

# do NOT include API key in source code
# use python-dotenv and add .env to .gitignore
genai.configure(api_key=os.environ["API_KEY"])

for m in genai.list_models():
  if 'generateContent' in m.supported_generation_methods:
    print(m.name)
```

Single Prompting

```
model = genai.GenerativeModel("gemini-2.5-flash-lite") # recommended

prompt = "Who invented Python?"
response = model.generate_content(prompt)

print(response.text)
```

Conversation History

Every call to `generate_content` is **stateless**, the model have no memory of past interactions. The simple implementation of memory would be to prepending the chat log to each prompt.

Or use the default implementation which summarises chat logs to reduce the lengh of each prompt. (saves tokens!)

```
chat = model.start_chat()

response = chat.send_message("Who invented Python?")
print(response.text)

response = chat.send_message("Tell me more about him!")
print(response.text)
```

Rules for Prompt Engineering

1. Simple prompt gets a simple answer, detailed prompt gets a detailed answer.
2. Provide **examples** of exaple what you want.
3. Setting rules using a **system prompt**, which is prepended to every prompt.

```
model = genai.GenerativeModel(
  "gemini-2.5-flash-lite".
  system_instruction = "You are a pirate, ARRRR!!!"
)
```

Tooling

You can provide functions which the model can call, you must write **documentation** explaining how the function should be used.

```
def add(a: int, b: int) -> int:
  """
  adds two integers together and returns the sum
  """
  return a + b

model = genai.GenerativeModel(
  "gemini-2.5-flash-lite".
  tools = [add]
)

chat = model.start_chat(enable_automatic_function_calling = True)
```

Embeddings

Definition
An **embedding** is a vector which represents a text's location in a high-dimensional meaning space. Text of similar meaning will have vectors close to each other.

```
result = genai.embed_content(
  model = "models/embedding-001".
  content = "What is the meaning of life?"
)

print(result['embedding']) # is a vector
```