```
!pip install -q langchain_experimental langchain_core
!pip install -q google-generativeai==0.3.1
!pip install -q google-ai-generativelanguage==0.4.0
!pip install -q langchain-google-genai
!pip install -q "langchain[docarray]"
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 163.0/163.0 kB 1.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 190.6/190.6 kB 9.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 809.1/809.1 kB 15.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.2/46.2 kB 5.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 32.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 6.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.3/215.3 kB 1.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 138.7/138.7 kB 9.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Building wheel for hnswlib (pyproject.toml) ... done
```

```
!pip show langchain langchain-core
```

```
Name: langchain
Version: 0.0.350
Summary: Building applications with LLMs through composability
Home-page: https://github.com/langchain-ai/langchain
Author:
Author-email:
License: MIT
Location: /usr/local/lib/python3.10/dist-packages
Requires: aiohttp, async-timeout, dataclasses-json, jsonpatch, langchain-community, langchain-core, langsmith, numpy, pydantic, PyYA
Required-by: langchain-experimental
---
Name: langchain-core
Version: 0.1.1
Summary: Building applications with LLMs through composability
Home-page: https://github.com/langchain-ai/langchain
Author:
Author-email:
License: MIT
Location: /usr/local/lib/python3.10/dist-packages
Requires: anyio, jsonpatch, langsmith, packaging, pydantic, PyYAML, requests, tenacity
Required-by: langchain, langchain-community, langchain-experimental, langchain-google-genai
```

```
import os
import google.generativeai as genai

from IPython.display import display
from IPython.display import Markdown

os.environ['GOOGLE_API_KEY'] = "API-KEY-FREE-YEAH"
```

```
genai.configure(api_key=os.environ['GOOGLE_API_KEY'])
```

```
models = [m for m in genai.list_models()]
```

```
models
```

```
            top_p=0.95,
            top_k=40),
      Model(name='models/embedding-gecko-001',
            base_model_id='',
            version='001',
            display_name='Embedding Gecko',
            description='Obtain a distributed representation of a text.',
            input_token_limit=1024,
            output_token_limit=1,
            supported_generation_methods=['embedText', 'countTextTokens'],
            temperature=None,
            top_p=None,
            top_k=None),
      Model(name='models/gemini-pro',
            base_model_id='',
            version='001',
            display_name='Gemini Pro',
            description='The best model for scaling across a wide range of tasks',
            input_token_limit=30720,
            output_token_limit=2048,
            supported_generation_methods=['generateContent', 'countTokens'],
            temperature=0.9,
            top_p=1.0,
            top_k=1),
      Model(name='models/gemini-pro-vision',
            base_model_id='',
            version='001',
            display_name='Gemini Pro Vision',
            description='The best image understanding model to handle a broad range of applications',
            input_token_limit=12288,
            output_token_limit=4096,
            supported_generation_methods=['generateContent', 'countTokens'],
            temperature=0.4,
            top_p=1.0,
            top_k=32),
      Model(name='models/embedding-001',
            base_model_id='',
            version='001',
            display_name='Embedding 001',
            description='Obtain a distributed representation of a text.',
            input_token_limit=2048,
            output_token_limit=1,
            supported_generation_methods=['embedContent', 'countTextTokens'],
            temperature=None,
            top_p=None,
            top_k=None),
      Model(name='models/aqa',
            base_model_id='',
            version='001',
            display_name='Model that performs Attributed Question Answering.',
            description=('Model trained to return answers to questions that are grounded in provided '
                         'sources, along with estimating answerable probability.'),
            input_token_limit=7168,
            output_token_limit=1024,
            supported_generation_methods=['generateAnswer'],
            temperature=0.2,
            top_p=1.0,
            top_k=40)]
```

```
# Generate Text

prompt = "Tell about deadlist Sea Monster any two and proof for existance"

model = genai.GenerativeModel('gemini-pro')

response = model.generate_content(prompt)
Markdown(response.text)
```

1. **Megalodon (Otodus Megalodon)**

- **Size:** Up to 59 feet (18 meters) long
- **Weight:** Up to 100 tons
- **Diet:** Whales, seals, and other marine mammals
- **Extinction:** Approximately 2.6 million years ago

Proof for Existence:

- Fossilized teeth and vertebrae have been found all over the world, suggesting that megalodon was a widespread predator.
- Whale bones from the Miocene and Pliocene epochs often show distinctive bite marks that are consistent with those of megalodon.

Google's Gemini with Langchain

2. **Livyatan Melvillei**

```
print("Basic LLM Chain")
```

```
Basic LLM Chain
```

- **Extinction:** Approximately 12 million years ago

```
from langchain_core.messages import HumanMessage
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatGoogleGenerativeAI(
    model="gemini-pro",
    temperature=0.7
)

result = llm.invoke("What is extact location of Titanic ship?")

Markdown(result.content)
```

The exact location of the Titanic shipwreck is:

- Latitude: 41° 43' 32" N
- Longitude: 49° 56' 49" W

This position is approximately 1,240 nautical miles (2,300 kilometers) east-southeast of Newfoundland, Canada, and about 370 miles (600 kilometers) south of the Grand Banks of Newfoundland.

The wreck of the Titanic lies on the seabed at a depth of approximately 12,415 feet (3,784 meters). It is split into two main sections, the bow and the stern, which are separated by a distance of about 1,970 feet (600 meters).

The exact location of the Titanic was first determined in 1985 by a joint French-American expedition led by Dr. Robert Ballard and Jean-Louis Michel. The expedition used a combination of sonar technology and a remotely operated vehicle (ROV) to locate and survey the wreck.

Since then, several other expeditions have visited the Titanic, including the 2010 expedition led by

```
for chunk in llm.stream("Write a famous titanic movie song."):
  print(chunk.content)
  print("---"*100)
```

```
(Verse 1)
In shadows of the night, a ship sets sail
--------------------------------------------------------------------------------------

With hearts entwined, a love that won't fail
A maiden voyage, a destiny unknown
Titanic, a legend to be shown

(Chorus
--------------------------------------------------------------------------------------
)
Oh, Titanic, sailing through the sea
A symphony of love, a tragic decree
With Rose and Jack, their passion takes flight
In this grand ship, they find their guiding light

(Verse 2)
```

Basic Multi Chain

```
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from langchain.prompts import ChatPromptTemplate
from langchain.schema.output_parser import StrOutputParser
```

```
(Bridge)
```

```
model = ChatGoogleGenerativeAI(
    model="gemini-pro",
    temperature=0.7
)
```

```
(Chorus)
```

```
prompt = ChatPromptTemplate.from_template(
    "Tell me a short facts about {topic} return in proper points"
)

output_parsers = StrOutputParser()
```

```
chain = prompt | model | output_parsers
```

```
Titanic's tale, forever engraved in our veins
```

```
chain.invoke({"topic":"Large Language Model"})
```

```
'- **Data-driven:** LLMs are trained on massive datasets of text and code, allowing them to learn patterns and relationships in lan
guage.\n- **Generative:** LLMs can generate text that is both coherent and grammatically correct, making them useful for a variety
of tasks such as writing articles, creating dialogue, and translating languages.\n- **Contextual:** LLMs take into account the cont
ext of the input they receive, allowing them to generate text that is relevant and responsive.\n- **Transfer learning:** LLMs can b
e fine-tuned on specific tasks or domains, allowing them to adapt to new scenarios quickly and efficiently.\n- **Scalability:** LLM
s can be scaled up to process larger amounts of data, resulting in improved performance and accuracy.\n- **Unsupervised learning:**
LLMs are typically trained using unsupervised learning methods, meaning they do not require labeled data.\n- **Attention mechanism
```

More Complicated Chain - Mini RAG

```
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from langchain.vectorstores import DocArrayInMemorySearch
```

```
model = ChatGoogleGenerativeAI(
    model="gemini-pro",
    temperature=0.7
)
```

```
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
```

```
vectorstore = DocArrayInMemorySearch.from_texts(
    ["Gemini Pro is a Large Language Model was made by GoogleDeepMind",
     "Gemini can be either a star sign or a name of a series of language models",
     "A Language model is trained by predicting the next token",
     "LLMs can easily do a variety of NLP tasks as well as text generation"],

    embedding=embeddings
)
```

```
retriever = vectorstore.as_retriever()
```

```
retriever.get_relevant_documents("what is Gemini?")
```

```
[Document(page_content='Gemini can be either a star sign or a name of a series of language models'),
 Document(page_content='Gemini Pro is a Large Language Model was made by GoogleDeepMind'),
 Document(page_content='A Language model is trained by predicting the next token'),
 Document(page_content='LLMs can easily do a variety of NLP tasks as well as text generation')]
```

```
retriever.get_relevant_documents("what is gemini pro?")
```

```
[Document(page_content='Gemini Pro is a Large Language Model was made by GoogleDeepMind'),
 Document(page_content='Gemini can be either a star sign or a name of a series of language models'),
 Document(page_content='A Language model is trained by predicting the next token'),
 Document(page_content='LLMs can easily do a variety of NLP tasks as well as text generation')]
```

```
template = """Answer the question a a full sentence, based only on the following context:
{context}

Return you answer in three back ticks

Question: {question}
"""

prompt = ChatPromptTemplate.from_template(template)
```

```
from langchain.schema.runnable import RunnableMap
```

```
retriever.get_relevant_documents("Who made Gemini pro?")
```

```
[Document(page_content='Gemini Pro is a Large Language Model was made by GoogleDeepMind'),
 Document(page_content='Gemini can be either a star sign or a name of a series of language models'),
 Document(page_content='A Language model is trained by predicting the next token'),
 Document(page_content='LLMs can easily do a variety of NLP tasks as well as text generation')]
```

```
chain = RunnableMap({
    "context":lambda x: retriever.get_relevant_documents(x['question']),
    "question":lambda x: x['question']
}) | prompt | model | output_parsers
```

```
chain.invoke({'question':'Who created Gemini pro?'})
```

```
'`GoogleDeepMind created Gemini pro.`'
```

PAL Chain

```
from langchain_experimental.pal_chain import PALChain
from langchain.chains.llm import LLMChain
```

```
model = ChatGoogleGenerativeAI(
    model="gemini-pro",
    temperature=0
)
```

```
pal_chain = PALChain.from_math_prompt(model,verbose=True)
```

```
question="There are 18 pencils in a box. If you give away 7 pencils, how many pencils are left?"
```

```
pal_chain.invoke(question)
```

```
> Entering new PALChain chain...
WARNING:langchain_community.utilities.python:Python REPL can execute arbitrary code. Use with caution.
```

```
question = "If you wake up at 7:00 a.m. and it takes you 1 hour and 30 minutes to get ready \
 and walk to school, at what time will you get to school?"

pal_chain.invoke(question)
```

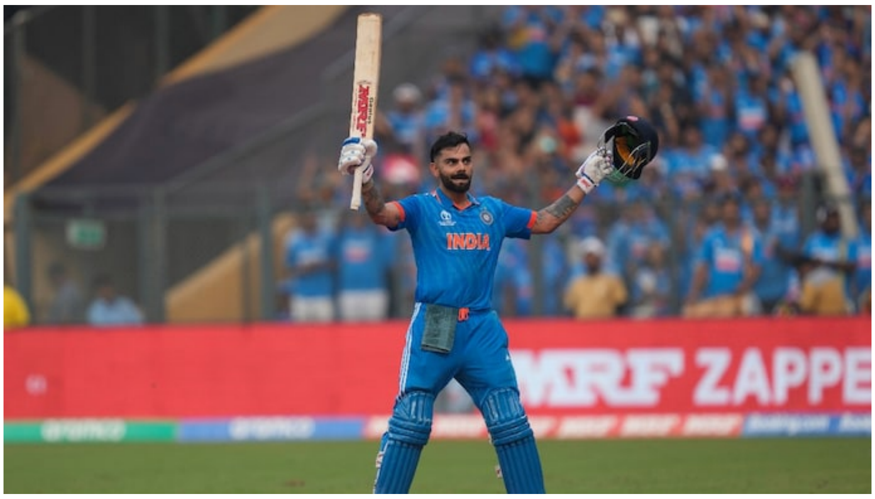
```
> Entering new PALChain chain...
def solution():
    """If you wake up at 7:00 a.m. and it takes you 1 hour and 30 minutes to get ready  and walk to school, at what time will you ge
    wake_up_time = 7
    hours_to_get_ready = 1
    minutes_to_get_ready = 30
    total_time_to_get_ready = hours_to_get_ready + minutes_to_get_ready / 60
    arrival_time = wake_up_time + total_time_to_get_ready
    result = arrival_time
    return result

> Finished chain.
{'question': 'If you wake up at 7:00 a.m. and it takes you 1 hour and 30 minutes to get ready  and walk to school, at what time
```

Multi Modal

```
import requests
from IPython.display import Image
```

```
image_url = "https://akm-img-a-in.tosshub.com/indiatoday/images/story/202311/virat-kohli-had-the-belief-of-going-past-sachin-tendulkar-r
```

```
content = requests.get(image_url).content
Image(content,width=650)
```



```
from langchain_core.messages import HumanMessage
from langchain_google_genai import ChatGoogleGenerativeAI
```

```
llm = ChatGoogleGenerativeAI(
    model="gemini-pro-vision"
)
```

```
message = HumanMessage(
    content=[
        {
            "type":"text",
            "text":"What's in this Image and Tell about that Person",
        },
        {
            "type":"image_url",
            "image_url":image_url
        },
    ]
)
```

```
llm.invoke([message])
```

```
AIMessage(content=' This is a picture of Virat Kohli, an Indian cricketer who is the current captain of the Indian national team.
He is considered one of the best batsmen in the world and is known for his aggressive batting style. He has scored over 20,000 runs
in international cricket, including 70 centuries. He is also the fastest batsman to reach 10,000 ODI runs.')
```

```
Start coding or generate with AI.
```