ExamplesGemini

March 3, 2026

1 Using Google Gemini models through the Gemini API

[ ]:
```
#!pip install -q -U google-genai
```

[1]:
```
import os
from google import genai
from dotenv import load_dotenv, find_dotenv

# Read the local .env file, containing the Gemini secret API key.
_ = load_dotenv(find_dotenv())

client = genai.Client(api_key = os.environ["GEMINI_API_KEY"])
```

1.1 Conversational API Example

[2]:
```
query = "Justin sits next to Razvan. One of them is happy and one of them is␣
 ↪grumpy. " \
                "The person sitting next to Justin is grumpy. Who is happy?"

response = client.models.generate_content(
    model = "gemini-2.5-flash",
    contents = query
)
print(response.text)
```

```
Let's break it down:

1.  Justin sits next to Razvan.
2.  One of them is happy, and one is grumpy.
3.  The person sitting next to Justin is grumpy.

Since Razvan is the person sitting next to Justin, **Razvan is grumpy**.

Therefore, **Justin is happy**.
```

1.2 Simple two-turn conversation

[3]:
```
chat = client.chats.create(model = "gemini-2.5-flash")

response = chat.send_message("Who received the Nobel prize in medicine for␣
 ↪cancer immunotherapy?")
# print(response.text)

response = chat.send_message("Where did they do their research?")
# print(response.text)

for message in chat.get_history():
    print(f'role - {message.role}:', end = '\n')
    print('\t')
    print(message.parts[0].text, end = '\n')
    print('\n')
```

```
role - user:

Who received the Nobel prize in medicine for cancer immunotherapy?


role - model:

The Nobel Prize in Physiology or Medicine for cancer immunotherapy was awarded
in **2018** to:

*   **James P. Allison**
*   **Tasuku Honjo**

They were recognized for their discovery of cancer therapy by inhibition of
negative immune regulation, which revolutionized cancer treatment.


role - user:

Where did they do their research?


role - model:

Both **James P. Allison** and **Tasuku Honjo** conducted their Nobel-winning
research at distinct, highly respected institutions.

*   **James P. Allison** did his pivotal work on CTLA-4 (Cytotoxic T-Lymphocyte-
Associated protein 4) and its role in T-cell inhibition, leading to the
development of the first immune checkpoint blockade therapy, primarily at the
**University of California, Berkeley**, specifically at the Cancer Research
```

Laboratory. He was also an investigator with the **Howard Hughes Medical Institute (HHMI)** during this period.
    *   Later, he continued his work at Memorial Sloan Kettering Cancer Center in New York and is currently at MD Anderson Cancer Center in Houston, Texas.

*   **Tasuku Honjo** discovered PD-1 (Programmed cell Death protein 1) and elucidated its function as an inhibitory receptor on T-cells, which opened the door for PD-1 blockade therapies. He conducted this groundbreaking research primarily at **Kyoto University** in Japan, where he has been a distinguished professor for many years.

1.3 Sequence Completion Example

[4]:
```
question = "Provide the next number in the sequence 1, 2, 3, 5, 5, 8, 7, 11, 9,␣
 ↪..."

response = client.models.generate_content(
    model = "gemini-2.5-flash",
    contents = question
)
print(response.text)
```

This sequence is actually two interleaved sequences:

1. **The numbers in the odd positions:** 1, 3, 5, 7, 9, … (This sequence adds 2 each time)
2. **The numbers in the even positions:** 2, 5, 8, 11, … (This sequence adds 3 each time)

The last number given (9) is in an odd position (9th term). The next number will be in an even position (10th term).

Following the pattern for the even positions: 2, 5, 8, 11, **14**

The next number in the sequence is **14**.

1.4 Image Understanding example: captioning

[5]:
```
from google.genai import types

with open('roxby.jpg', 'rb') as f:
    image_bytes = f.read()

    response = client.models.generate_content(
        model = 'gemini-2.5-flash',
        contents = [
```

```
            types.Part.from_bytes(
            data = image_bytes,
            mime_type = 'image/jpeg'),
          'Caption this image.'
        ])

print(response.text)
```

A young child, wearing a Hello Kitty shirt and a blue smiley face sticker on their hand, is diligently coloring an underwater scene on a "Boardwalk Billy's" restaurant placemat with a blue crayon. Other crayons, a word search, and brunch menus are spread across the wooden table, providing entertainment during the meal.

1.5 Image Understanding example: conversation

[6]:
```
from google.genai import types

with open('pump.jpg', 'rb') as f:
    image_bytes = f.read()

    response1 = client.models.generate_content(
        model = 'gemini-2.5-pro', # 'gemini-3.1-pro-preview'
        contents = [
            types.Part.from_bytes(data = image_bytes, mime_type = 'image/jpeg'),
            'What is in this image?'
        ])

print(response1.text)
```

This is a close-up photograph of a gas pump display and the surrounding unit.

On the digital display:
*   The top line, next to a dollar sign ($), shows the total cost as **20.00**.
*   The bottom line, next to the word "Gallons," shows the amount of fuel pumped as **10.403**.
*   A reflection of the person taking the picture is visible on the screen. It appears to be a man with glasses and a beard, holding a phone.

On the gas pump unit:
*   The pump is white and has the number **5** in green on the right side.
*   Numerous stickers and advertisements are visible.
*   Above the display is a large blue and yellow ad for a partnership between **earnify & Amazon Prime** offering savings of "10¢ off EVERY GALLON."
*   To the left of the display is a green **2025 Approval Seal** from the North Carolina Department of Agriculture & Consumer Services.
*   To the right, a white sticker warns, "**DO NOT USE PHONE WHILE REFUELING**."
*   Below this are notices about refueling services for motorists with

disabilities and a yellow **WARNING** sign with pictograms prohibiting smoking, leaving keys in the vehicle, and using a phone.
*   Other stickers promote the "earnify" rewards program and partnerships with **bp** and **Amoco**.
*   Logos for accepted payment methods like **Discover, VISA, Mastercard, and American Express** are visible at the bottom.

[7]:
```
response2 = client.models.generate_content(
        model = 'gemini-2.5-pro',
        contents = [
        genai.types.Content(
            role = "user",
            parts = [
                    types.Part.from_bytes(data = image_bytes, mime_type =␣
 ↪'image/jpeg'),
                    types.Part(text = "What is in this image?"),
                    # genai.types.Part(inline_data=image_data)
                ]),
        genai.types.Content(
            role = "model",
            parts = [types.Part(text = response1.text)]),
        genai.types.Content(
            role = "user",
            parts = [types.Part(text = "What is the price per gallon at this␣
 ↪gas station?")])
    ]
)
print(response2.text)
```

Based on the numbers on the display, we can calculate the price per gallon.

*   **Total Cost:** $20.00
*   **Gallons Pumped:** 10.403

By dividing the total cost by the number of gallons ($20.00 / 10.403), the price per gallon is approximately **$1.923**.

[8]:
```
with open('museum.png', 'rb') as f:
    image_bytes = f.read()

    response = client.models.generate_content(
        model = 'gemini-2.5-pro',
        contents = [
            types.Part.from_bytes(data = image_bytes, mime_type = 'image/png'),
            'Provide a short, humorous caption for this image.'
        ])

print(response.text)
```

```
The family reunion got a little emotional.
```

1.6 Image Understanding + Reasoning example

For large files or to be able to use the same image file repeatedly, we can upload images using the File upload API.

[9]:
```
math_file = client.files.upload(file = "math.png")

response3 = client.models.generate_content(
    model = "gemini-2.5-flash",
    contents = [math_file, "Extract the text from this image."],
)

print(response3.text)
```

```
A stock went down 60% to $17.80. What percent does it need to rise to get back
to its original price?
```

[10]:
```
from google.genai import types

response4 = client.models.generate_content(
        model = 'gemini-2.5-pro',
        contents = response3.text,
#         config = types.GenerateContentConfig(
#             temperature = 0,
#             candidate_count = 1, # default
#             max_output_tokens = 500)
)

print(response4.text)
```

```
Excellent question! This is a classic problem that shows why the percentage
change is different depending on your starting point.

The stock needs to rise by **150%** to get back to its original price.

Here's the step-by-step breakdown:

### Step 1: Find the Original Price

*   If the stock went down by 60%, it means the current price ($17.80) is only
**40%** of its original value (100% - 60% = 40%).
*   Let 'P' be the original price. We can set up the equation:
    `0.40 * P = $17.80`
*   To find the original price, divide $17.80 by 0.40:
    `P = $17.80 / 0.40`
    `P = $44.50`
```

```
So, the original price of the stock was **$44.50**.

### Step 2: Calculate the Increase Needed

*   To get from the new price back to the original price, we need to find the
difference in dollars:
    `$44.50 (Original Price) - $17.80 (New Price) = $26.70`

The stock needs to increase by **$26.70**.

### Step 3: Calculate the Percentage Rise

*   To find the percentage increase, you compare the dollar increase to the
price you are starting from ($17.80).
*   The formula is: `(Amount of Increase / Starting Price) * 100`
    `($26.70 / $17.80) * 100 = 1.5 * 100 = 150%`

### Why Isn't It Just 60%?

The 60% decrease was calculated from a higher original price ($44.50). The
percentage increase required to get back to that original price is calculated
from the new, much lower price ($17.80). Because the starting base is smaller,
the percentage increase must be larger to cover the same dollar amount.
```

1.7 Code execution tool

[11]:
```python
from google import genai
from google.genai.types import (
#    HttpOptions,
    Tool,
    ToolCodeExecution,
    GenerateContentConfig,
)

#client = genai.Client(http_options = HttpOptions(api_version="v1"))
model_id = "gemini-2.5-flash"

code_execution_tool = Tool(code_execution = ToolCodeExecution())
response = client.models.generate_content(
    model = model_id,
    contents = "Calculate 20th fibonacci number. Then find the nearest␣
 ↪palindrome to it.",
    config = GenerateContentConfig(
        tools = [code_execution_tool],
        temperature = 0,
    ),
)
```

```
print("# Code:")
print(response.executable_code)
print("# Outcome:")
print(response.code_execution_result)
```

```
# Code:
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a

fib_20 = fibonacci(20)
print(f'{fib_20=}')
# Outcome:
fib_20=6765
```

[ ]: