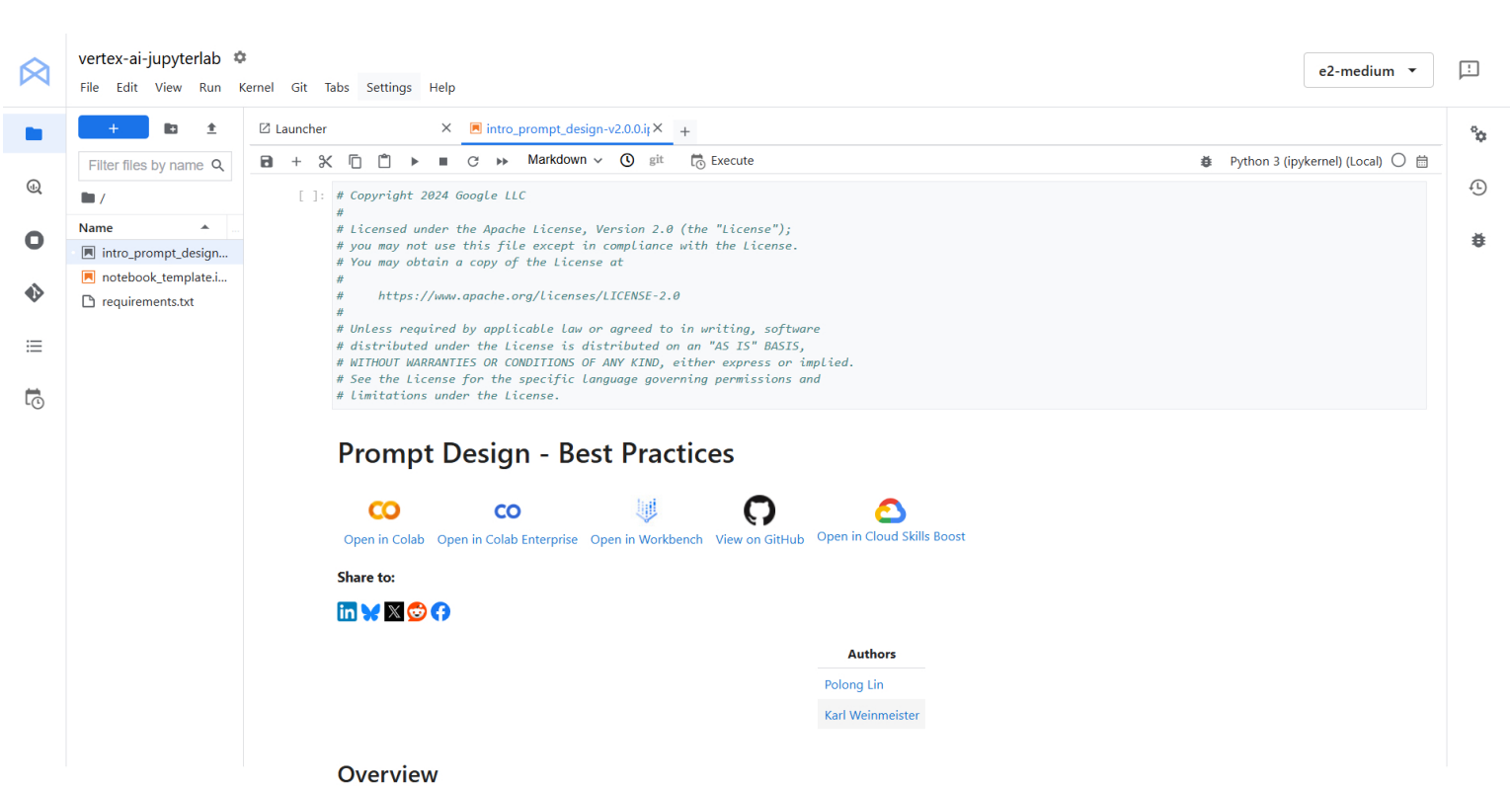
vertex-ai-jupyterlab
File Edit View Run Kernel Git Tabs Settings Help
e2-medium
Launcher
intro_prompt_design-v2.0.0.ip
Markdown
git
Execute
Python 3 (ipykernel) (Local)
Filter files by name
/
Name
intro_prompt_design...
notebook_template.i...
requirements.txt
[ ]:
Copyright 2024 Google LLC
#
Licensed under the Apache License, Version 2.0 (the "license");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
#
https://www.apache.org/licenses/LICENSE-2.0
#
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
Prompt Design - Best Practices
Open in Colab
Open in Colab Enterprise
Open in Workbench
View on GitHub
Open in Cloud Skills Boost
Share to:
Authors
Polong Lin
Karl Weinmeister
Overview
This notebook covers the essentials of prompt engineering, including some best practices.
Learn more about prompt design in the official documentation.
In this notebook, you learn best practices around prompt engineering -- how to design prompts to improve the quality of your responses.
This notebook covers the following best practices for prompt engineering:
Be concise
Be specific and well-defined
Ask one task at a time
Turn generative tasks into classification tasks
Improve response quality by including examples
Getting Started
Install Google Gen AI SDK
[1]:
%pip install --upgrade --quiet google-genai
Note: you may need to restart the kernel to use updated packages.
Restart runtime
To use the newly installed packages in this Jupyter runtime, you must restart the runtime. You can do this by running the cell below, which will restart the current kernel.
[2]:
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
[2]:
{'status': 'ok', 'restart': True}
The kernel is going to restart. Please wait until it is finished before continuing to the next step.
Authenticate your notebook environment (Colab only)
Authenticate your environment on Google Colab.
[1]:
import sys
if "google.colab" in sys.modules:
from google.colab import auth
auth.authenticate_user()
Import libraries
[2]:
from IPython.display import Markdown, display
from google import genai
from google.genai.types import GenerateContentConfig
Set Google Cloud project information and create client
To get started using Vertex AI, you must have an existing Google Cloud project and enable the Vertex AI API.
Learn more about setting up a project and a development environment.
Initialize the Gen AI SDK for Python for your project.

[3]:
```
# Define project information
PROJECT_ID = "qwiklabs-gcp-02-fec0c2c8ac8f"  # @param {type:"string"}
LOCATION = "us-central1"  # @param {type:"string"}

# Create the API client
from google import genai
client = genai.Client(vertexai=True, project=PROJECT_ID, location=LOCATION)
```

Load model

Learn more about all Gemini models on Vertex AI.

[4]:
```
MODEL_ID = "gemini-2.0-flash-001"  # @param {type: "string"}
```

Prompt engineering best practices

Prompt engineering is all about how to design your prompts so that the response is what you were indeed hoping to see.

The idea of using "unfancy" prompts is to minimize the noise in your prompt to reduce the possibility of the LLM misinterpreting the intent of the prompt. Below are a few guidelines on how to engineer "unfancy" prompts.

In this section, you'll cover the following best practices when engineering prompts:

- Be concise
- Be specific, and well-defined
- Ask one task at a time
- Improve response quality by including examples
- Turn generative tasks to classification tasks to improve safety

Be concise

🔴 Not recommended. The prompt below is unnecessarily verbose.

[5]:
```
prompt = "What do you think could be a good name for a flower shop that specializes in selling bouquets of dried flowers more than fresh flowers?"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, here are some name ideas for a dried flower shop, categorized for different vibes:

Elegant & Sophisticated:

- **The Everbloom Studio:** Sounds timeless and artistic.
- **Petrified Petals:** A bit edgy but hints at the preservation process.
- **Golden Still Life:** Evokes a sense of artistic beauty and permanence.
- **The Dried Daisy:** Simple, memorable, and elegant
- **The Preserved Posy:** Cute, alliterative, and speaks to the product.

Rustic & Natural:

- **Wild & Withered:** Has a romantic, nature-inspired feel.
- **The Dusty Bloom:** Suggests a vintage, rustic aesthetic.
- **Prairie Petals:** If you source from or emulate prairie flowers.
- **The Seed & Stem:** Simple, earthy, and evokes the origins of flowers.
- **Whispering Wheat:** Emphasizes the dried nature, especially if you use grains.

Modern & Minimalist:

- **Still Life Flowers:** Clean and direct.
- **The Dried Flower Co.:** Simple and business-like.
- **Floral Echoes:** Suggests a lingering beauty.
- **Studio Dried:** Short, punchy, and emphasizes the curated aspect.
- **Bloom Edit:** Suggests a curated selection of dried flowers.

Playful & Whimsical:

- **The Blooming Bones:** A little quirky and memorable.
- **Forever Flora:** Emphasizes the longevity of dried flowers.
- **Petal & Paper:** Suggests the dried texture and craft aspect.
- **The Thistle & Twig:** Creates a charming, storybook feel.
- **Bloom Box:** if you're selling in boxes.

Tips for Choosing a Name:

- **Check Availability:** Make sure the name isn't already in use by another flower shop in your area and that the domain name is available.
- **Consider Your Target Audience:** Who are you trying to attract? A younger, trendier audience might respond to a more modern name, while an older audience might prefer something more traditional.
- **Think About Your Branding:** The name should align with the overall aesthetic of your shop and your brand.
- **Say it Out Loud:** Make sure the name is easy to pronounce and remember.
- **Get Feedback:** Ask friends, family, or potential customers what they think of your top choices.

I hope this helps! Good luck with your flower shop!

✅ Recommended. The prompt below is to the point and concise.

[6]:
```
prompt = "Suggest a name for a flower shop that sells bouquets of dried flowers"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, here are some name suggestions for a dried flower shop, categorized by vibe:

Elegant & Classic:

- The Everlasting Bloom
- Dried & True
- Preserved Petals
- The Dried Bouquet
- Eternal Blooms
- The Conservatory Dry
- Golden Flora

- Withered Wonders

Modern & Minimalist:

- Dry Goods Floral
- The Dried Stalk
- Still Life Flowers
- Dried Design
- The Floral Archive
- Dry Bloom
- Studio Dry
- Root & Stem (even though they are dried)

Whimsical & Playful:

- The Never-Ending Garden
- Petal Pushers (Dried)
- Forever Flowers
- Dust & Bloom
- The Dried Fairy
- Sun Dried Florals

Location Specific:

- (Your City/Town) Dried Flowers
- (Local Landmark) Florals
- (Street Name) Blooms

Rustic & Earthy:

- Prairie Dried
- Field & Stem
- Harvest Blooms
- The Rustic Bloom
- Wildflower Dry
- Earthbound Florals
- Golden Grass
- Sun Bleached Florals

Tips for Choosing a Name:

- **Availability:** Check if the name is available as a website domain and social media handle.
- **Target Audience:** Consider who you are trying to reach with your shop. Does your name appeal to them?
- **Memorability:** Is it easy to remember and pronounce?
- **Branding:** Does the name fit the overall aesthetic and brand you want to create?

Bonus Tip: Say the names out loud! See how they feel and sound. Good luck!

Be specific, and well-defined

Suppose that you want to brainstorm creative ways to describe Earth.

🔴 The prompt below might be a bit too generic (which is certainly OK if you'd like to ask a generic question!)

[7]:
```
prompt = "Tell me about Earth"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, let's dive into the fascinating subject of Earth! Here's a breakdown of key aspects:

Overview:

- **Our Home:** Earth is the third planet from the Sun and the only known celestial body to harbor life.
- **Part of the Solar System:** It resides in the Solar System, which is located in the Orion Arm of the Milky Way galaxy.
- **Age:** Estimated to be around 4.54 billion years old.
- **Shape:** An oblate spheroid (slightly flattened at the poles and bulging at the equator).
- **Symbol:** ⊕

Key Physical Characteristics:

- **Size:**
  - Equatorial Radius: 6,378.1 kilometers (3,963.2 miles)
  - Polar Radius: 6,356.8 kilometers (3,949.0 miles)
  - Equatorial Circumference: 40,075 kilometers (24,901 miles)
- **Mass:** 5.972 × 10^24 kg
- **Density:** 5.514 g/cm^3 (the densest planet in the Solar System)
- **Rotation:** Rotates on its axis once every 23 hours, 56 minutes, and 4 seconds (approximately 24 hours, defining our day). The tilt of Earth's axis (about 23.5 degrees) is responsible for the seasons.
- **Revolution:** Orbits the Sun once every 365.25 days (approximately one year). The extra .25 days are accounted for by leap years.
- **Atmosphere:**
  - Primarily composed of nitrogen (78%) and oxygen (21%), with trace amounts of argon, carbon dioxide, and other gases.
  - The atmosphere protects us from harmful solar radiation and regulates the planet's temperature.
- **Surface:**
  - 71% covered by water (oceans, seas, lakes, rivers, ice).
  - 29% land, consisting of continents and islands.
- **Internal Structure:**
  - **Crust:** The outermost layer, relatively thin and rocky. It's divided into tectonic plates that move and interact, causing earthquakes, volcanoes, and mountain formation.
  - **Mantle:** A thick, mostly solid layer beneath the crust, composed of silicate rocks rich in iron and magnesium.
  - **Outer Core:** A liquid layer composed mainly of iron and nickel. The movement of this liquid generates Earth's magnetic field.
  - **Inner Core:** A solid sphere composed primarily of iron and nickel, under immense pressure.
- **Magnetic Field:** Generated by the movement of molten iron in the outer core. It protects Earth from harmful solar wind.
- **Temperature:** Varies widely depending on location and season. The average global surface temperature is around 15°C (59°F).
- **Water:** Exists in all three states (solid, liquid, gas) and is essential for life as we know it.

Life and Biosphere:

- **Biodiversity:** Earth is home to an incredible diversity of life, from microscopic organisms to giant whales and towering trees

- **Biodiversity:** Earth is home to an incredible diversity of life, from microscopic organisms to giant whales and towering trees.
- **Ecosystems:** Diverse ecosystems exist, each with its unique set of organisms and interactions (e.g., rainforests, deserts, coral reefs).
- **Humans:** Humans are the dominant species on Earth, significantly impacting the planet's environment.
- **Biosphere:** The part of Earth where life exists, including the atmosphere, lithosphere (land), and hydrosphere (water).

Moon (Natural Satellite):

- Earth has one natural satellite, the Moon.
- It's believed to have formed from debris resulting from a giant impact between Earth and a Mars-sized object early in the Solar System's history.
- The Moon influences Earth's tides and stabilizes the planet's axial tilt.

Plate Tectonics:

- Earth's lithosphere is divided into several large and small tectonic plates.
- These plates are constantly moving, driven by convection currents in the mantle.
- Plate tectonics are responsible for many geological phenomena, including:
  - Earthquakes
  - Volcanoes
  - Mountain building
  - Continental drift

Earth's Place in the Solar System:

- It's the third planet from the Sun, situated within the "habitable zone" (also known as the "Goldilocks zone"), where temperatures are suitable for liquid water to exist on the surface.
- It is positioned between Venus and Mars.
- Earth is the largest of the four inner, rocky planets (Mercury, Venus, Earth, and Mars).

Unique Aspects:

- **Liquid Water:** The abundance of liquid water on the surface is a unique characteristic among the planets in our Solar System and is essential for life.
- **Plate Tectonics:** Earth is the only planet in our Solar System known to have active plate tectonics.
- **Oxygen-Rich Atmosphere:** The high concentration of oxygen in Earth's atmosphere is a result of photosynthesis by plants and algae.
- **Life:** The existence of life is, as far as we know, unique to Earth.

Human Impact:

- Human activities have a significant impact on Earth's environment.
- Key concerns include:
  - Climate change (due to greenhouse gas emissions)
  - Deforestation
  - Pollution (air, water, and land)
  - Loss of biodiversity
- Sustainable practices are crucial for preserving Earth's environment for future generations.

Ongoing Research and Exploration:

- Scientists continue to study Earth to understand its past, present, and future.
- Space missions and remote sensing technologies provide valuable data about the planet's atmosphere, oceans, land, and interior.
- Research efforts focus on:
  - Climate change
  - Natural disasters
  - Resource management
  - The search for life beyond Earth

In Summary:

Earth is a dynamic and complex planet, characterized by its unique combination of features that support life. Understanding Earth's processes and our impact on them is crucial for ensuring its long-term sustainability.

✅ Recommended. The prompt below is specific and well-defined.

[8]:
```
prompt = "Generate a list of ways that makes Earth unique compared to other planets"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, here's a list of things that make Earth unique (or at least, highly unusual) compared to other planets we know of, focusing on what sets it apart:

Key Features Supporting Life & Conditions

- **Liquid Water on the Surface:** This is arguably the most significant. Earth is the only known planet in our solar system with stable bodies of liquid water on its surface. Water is essential for life as we know it, acting as a solvent, a transport medium, and participating in crucial biochemical reactions.
- **Oxygen-Rich Atmosphere:** Earth's atmosphere is about 21% oxygen, a byproduct of photosynthesis by plants and algae. Most other planets have atmospheres dominated by carbon dioxide, nitrogen, or hydrogen, with very little free oxygen. This oxygen is crucial for complex, energy-intensive life forms like animals.
- **Plate Tectonics:** Earth's crust is divided into moving plates. This process recycles materials between the Earth's interior and surface, helps regulate temperature and the carbon cycle, and is linked to the formation of continents and mountain ranges. Plate tectonics is not definitively confirmed on other planets in our solar system.
- **A Strong Magnetic Field:** Generated by the Earth's iron core, the magnetic field deflects harmful solar wind and cosmic radiation, protecting the atmosphere and surface life.
- **Ozone Layer:** A layer in the stratosphere that absorbs most of the Sun's harmful ultraviolet (UV) radiation, making the surface habitable.
- **Relatively Stable Axial Tilt:** Earth's axial tilt (obliquity) is relatively stable due to the presence of the Moon. This stable tilt results in predictable and moderate seasons, which are conducive to life. Planets with highly variable axial tilts can experience extreme climate swings.
- **Habitable Zone Location:** Earth orbits the Sun at a distance that allows for liquid water to exist on its surface – not too hot, not too cold. This region around a star is called the habitable zone or Goldilocks zone.
- **Complex Ecosystems & Biodiversity:** Earth is the only planet we know of with a vast and diverse range of life, from microscopic bacteria to giant whales, interacting in complex ecosystems.
- **Presence of a Large Moon:** Our Moon is unusually large compared to Earth. It stabilizes our axial tilt, influences tides, and may have played a role in the early development of life.

Other Notable Differences:

- **Surface Composition:** While other planets have rocky surfaces, Earth's composition is unique in its specific blend of minerals, water, and other materials.
- **Albedo:** Earth has a specific albedo (reflectivity) that influences how much solar energy it absorbs. The presence of clouds, ice, and vegetation contributes to this albedo.
- **Age and Geological History:** Earth has a long and complex geological history, shaped by various processes over billions of years. This history has influenced the planet's current state and its suitability for life.
- **Active Hydrological Cycle:** Earth has a dynamic water cycle, involving evaporation, condensation, precipitation, and runoff. This cycle redistributes heat, shapes landscapes, and is vital for life.

Important Considerations:

- **Our Limited Knowledge:** We have only extensively studied a very small number of planets, primarily those in our own solar system and a small fraction of exoplanets. It's possible that planets with similar characteristics to Earth exist elsewhere in the universe, but we haven't found them yet.
- **Definition of "Unique":** Some of these features might exist to some degree on other planets, but the combination of all of them, in the right balance, is what makes Earth truly special.
- **Changing Understanding:** Our understanding of planetary science is constantly evolving, and new discoveries may challenge or refine this list in the future.

In summary, Earth's unique combination of liquid water, an oxygen-rich atmosphere, plate tectonics, a strong magnetic field, a stable climate, and a location in the habitable zone makes it a very special place in the universe – at least, as far as we know right now.

Ask one task at a time

🔴 Not recommended. The prompt below has two parts to the question that could be asked separately.

[9]:
```
prompt = "What's the best method of boiling water and why is the sky blue?"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Let's tackle these two very different questions:

Best Method of Boiling Water

There isn't a single "best" method, as it depends on your priorities and resources. Here's a breakdown:

- **Fastest:**
  - **Electric Kettle:** Generally the fastest for small to medium quantities of water (up to a liter or two). They are designed specifically for boiling water, and efficiency is a priority.
- **Most Energy Efficient (for small amounts):**
  - **Electric Kettle:** Again, the electric kettle often wins for small amounts because all the energy is directly heating the water, with minimal loss.
- **Most Energy Efficient (for larger amounts):**
  - **Induction Cooktop:** Induction is highly efficient because it directly heats the pot itself, not the surrounding air. This minimizes energy loss.
- **Most Convenient (for some situations):**
  - **Microwave:** Convenient for very small amounts (a cup or two) and for quick heating. However, it's generally not as efficient as other methods and can be uneven. It can also be dangerous if you superheat the water (see below).
- **Most Versatile (for general cooking):**
  - **Stovetop (Gas or Electric):** Good for boiling water as part of cooking, when you already have the stovetop on. Gas is often faster to heat up, but electric can be more evenly heated, especially flat glass cooktops.
- **Off-Grid:**
  - **Campfire/Wood Stove:** Obviously, necessary when electricity or gas isn't available.

Important Considerations:

- **Quantity:** The best method for a single cup of tea is different from the best method for a large pot of pasta water.
- **Safety:** Always use caution when handling hot water.
- **Superheating:** Microwaving water can sometimes lead to *superheating*. The water becomes hotter than its boiling point without actually boiling. When you disturb it (e.g., add a spoon), it can erupt violently. Add a wooden spoon or chopstick when microwaving to allow for even heating.

Why is the Sky Blue?

The sky is blue due to a phenomenon called **Rayleigh scattering.** Here's a simplified explanation:

1. **Sunlight and the Atmosphere:** Sunlight is made up of all the colors of the rainbow. When sunlight enters the Earth's atmosphere, it collides with tiny air molecules (mostly nitrogen and oxygen).
2. **Scattering of Light:** This collision causes the light to scatter in different directions.
3. **Blue Light is Scattered More:** Blue and violet light have shorter wavelengths compared to other colors. Shorter wavelengths are scattered *much* more effectively by these tiny air molecules than longer wavelengths like red and orange. Rayleigh scattering is inversely proportional to the fourth power of the wavelength. This means if the wavelength is halved, the scattering is 16 times stronger (2^4 = 16).
4. **Why not Violet?** Although violet light is scattered even *more* than blue, there are a few reasons why we see a blue sky:
   - The sun emits slightly less violet light than blue light.
   - Our eyes are more sensitive to blue light than violet light.
   - The upper atmosphere absorbs some of the violet light.
5. **What About Sunsets?** At sunrise and sunset, the sunlight has to travel through a much greater distance of the atmosphere to reach our eyes. As it travels, most of the blue light is scattered away. This leaves the longer wavelengths like orange and red to dominate, creating beautiful sunset colors.

In short: Blue light is scattered more than other colors by the air molecules in the atmosphere, making the sky appear blue from our perspective.

✅ Recommended. The prompts below asks one task a time.

[10]:
```
prompt = "What's the best method of boiling water?"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

There's no single "best" method for boiling water, as the ideal approach depends on factors like:

- **Speed:** How quickly do you need the water to boil?
- **Energy Efficiency:** How much electricity or fuel do you want to use?
- **Convenience:** How easy is the method to use and clean?
- **Quantity:** How much water do you need to boil?
- **Accessibility:** What equipment do you have available?

Here's a breakdown of common methods and their pros and cons:

1. Electric Kettle:

- **Pros:**
  - **Fast:** Generally the fastest way to boil small to medium amounts of water.
  - **Efficient:** Uses only the amount of energy needed. Many have auto shut-off.
  - **Convenient:** Easy to use, often with one-button operation.
  - **Safe:** Typically has automatic shut-off to prevent boiling dry.
- **Cons:**
  - Limited capacity.
  - Requires an electrical outlet.

2. Stovetop Kettle:

- **Pros:**

- **Pros:**
  - **Versatile:** Can be used on gas, electric, or induction stoves.
  - **Large capacity:** Can boil larger amounts of water compared to electric kettles.
  - **Durable:** Typically lasts a long time.
- **Cons:**
  - **Slower** than electric kettles.
  - **Less efficient** than electric kettles, as heat is lost to the surrounding air.
  - Requires monitoring to prevent boiling dry.

3. Stovetop Pot (without a lid):

- **Pros:**
  - **Most basic:** Requires only a pot and a heat source.
  - **Large capacity:** Can boil very large amounts of water.
- **Cons:**
  - **Slowest** method.
  - **Least efficient:** Significant heat loss.
  - **Evaporation:** More water evaporates, which can be an issue if you need a specific amount.

4. Stovetop Pot (with a lid):

- **Pros:**
  - **Faster and more efficient** than boiling in a pot without a lid.
  - **Reduces evaporation.**
- **Cons:**
  - Still slower and less efficient than electric kettles.
  - Requires monitoring.

5. Microwave:

- **Pros:**
  - **Fast:** Can be quick for small amounts of water.
  - **Convenient:** Easy to use.
- **Cons:**
  - **Uneven heating:** Can result in superheated water, which can violently erupt when disturbed. **Always use a microwave-safe container and never boil water in a sealed container.** Use a wooden or plastic utensil inside the container to disturb the water before removing it.
  - **Less efficient** than electric kettles.
  - Not ideal for large amounts of water.

Summary Table:

Method	Speed	Efficiency	Convenience	Capacity	Notes
Electric Kettle	Fastest	Highest	Highest	Small-Med	Ideal for quick, efficient boiling of smaller amounts.
Stovetop Kettle	Medium	Medium	Medium	Med-Large	Good balance of speed and capacity.
Pot (with lid)	Medium-Slow	Medium	Low	Large	Suitable for larger amounts, but slower and less efficient.
Pot (no lid)	Slowest	Lowest	Low	Large	Least efficient, should only be used if no other option is available.
Microwave	Fast	Low	High	Small	Use with caution due to potential for superheating. Only for small amounts of water.

Recommendations:

- **For everyday use, quick cups of tea/coffee:** An **electric kettle** is generally the best choice due to its speed, efficiency, and convenience.
- **For larger quantities of water or when an electrical outlet isn't available:** A **stovetop kettle** is a good option.
- **If you only need to boil a very small amount and are in a hurry:** A **microwave** *can* be used, but with caution. Stir the water afterwards to prevent eruptions.
- **In a pinch, or if you need a *very* large quantity:** A **stovetop pot with a lid** will work.

[11]:
```
prompt = "Why is the sky blue?"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

The sky is blue because of a phenomenon called **Rayleigh scattering**. Here's a breakdown of the process:

- **Sunlight enters the Earth's atmosphere:** Sunlight is composed of all the colors of the rainbow.
- **Light collides with air molecules:** The atmosphere is made up of gases like nitrogen and oxygen. When sunlight enters the atmosphere, it collides with these tiny air molecules.
- **Scattering of light:** This collision causes the light to scatter in different directions. This scattering is more effective for shorter wavelengths of light. Blue and violet light have shorter wavelengths compared to other colors like red and orange.
- **Blue light scatters the most:** Because blue and violet light have shorter wavelengths, they are scattered much more strongly than other colors. Violet is scattered even more than blue, but our eyes are more sensitive to blue light and the sun emits less violet light to begin with.
- **Why not violet?** While violet light is scattered even *more* than blue light, several factors contribute to why we perceive the sky as blue rather than violet:
  - **The sun emits less violet light:** The sun's spectrum contains less violet light compared to blue light.
  - **Our eyes are less sensitive to violet:** The human eye is less sensitive to violet light than it is to blue light.
  - **Atmospheric absorption:** Some violet light is absorbed by the upper atmosphere.
- **Reaching our eyes:** The scattered blue light reaches our eyes from all directions, making the sky appear blue.

In summary: Sunlight interacts with air molecules in the atmosphere, causing the shorter wavelengths of blue light to scatter more effectively. This scattered blue light is what we see when we look at the sky.

Watch out for hallucinations

Although LLMs have been trained on a large amount of data, they can generate text containing statements not grounded in truth or reality; these responses from the LLM are often referred to as "hallucinations" due to their limited memorization capabilities. Note that simply prompting the LLM to provide a citation isn't a fix to this problem, as there are instances of LLMs providing false or inaccurate citations. Dealing with hallucinations is a fundamental challenge of LLMs and an ongoing research area, so it is important to be cognizant that LLMs may seem to give you confident, correct-sounding statements that are in fact incorrect.

Note that if you intend to use LLMs for the creative use cases, hallucinating could actually be quite useful.

Try the prompt like the one below repeatedly. We set the temperature to `1.0` so that it takes more risks in its choices. It's possible that it may provide an inaccurate, but confident answer.

[12]:
```
generation_config = GenerateContentConfig(temperature=1.0)
```

```
prompt = "What day is it today?"

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

I am a large language model, so I don't know what day it is today.

Since LLMs do not have access to real-time information without further integrations, you may have noticed it hallucinates what day it is today in some of the outputs.

Using system instructions to guardrail the model from irrelevant responses

How can we attempt to reduce the chances of irrelevant responses and hallucinations?

One way is to provide the LLM with system instructions.

Let's see how system instructions works and how you can use them to reduce hallucinations or irrelevant questions for a travel chatbot.

Suppose we ask a simple question about one of Italy's most famous tourist spots.

[13]:
```
generation_config = GenerateContentConfig(temperature=1.0)

chat = client.chats.create(
    model=MODEL_ID,
    config=GenerateContentConfig(
        system_instruction=[
            "Hello! You are an AI chatbot for a travel web site.",
            "Your mission is to provide helpful queries for travelers.",
            "Remember that before you answer a question, you must check to see if it complies with your mission.",
            "If not, you can say, Sorry I can't answer that question.",
        ]
    ),
)

prompt = "What is the best place for sightseeing in Milan, Italy?"

response = chat.send_message(prompt)
display(Markdown(response.text))
```

The Duomo di Milano is the most famous place for sightseeing in Milan, Italy.

Now let us pretend to be a user asks the chatbot a question that is unrelated to travel.

[14]:
```
prompt = "What is the best place for sightseeing in Milan, Italy?"

response = chat.send_message(prompt)
display(Markdown(response.text))
```

Sure, I can help you with that! The Duomo di Milano, Santa Maria delle Grazie, Galleria Vittorio Emanuele II, Teatro alla Scala, Pinacoteca di Brera, Castello Sforzesco, Navigli District, and the San Siro Stadium are all great places for sightseeing in Milan, Italy.

You can see that this way, a guardrail in the prompt prevented the chatbot from veering off course.

Turn generative tasks into classification tasks to reduce output variability

Generative tasks lead to higher output variability

The prompt below results in an open-ended response, useful for brainstorming, but response is highly variable.

[15]:
```
prompt = "I'm a high school student. Recommend me a programming activity to improve my skills."

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, here are a few programming activities, tailored for a high school student looking to improve their skills, categorized by difficulty and focusing on different areas:

Beginner-Friendly (Focus on fundamentals and problem-solving):

- **1. Text-Based Adventure Game:**
  - **Concept:** Create a simple text-based adventure game where the user makes choices and the program responds accordingly.
  - **Skills Reinforced:**
    - `if/else` statements
    - `input/output`
    - Variables
    - Functions (to organize game logic)
    - Basic string manipulation
  - **Languages:** Python (easiest to start), Java (slightly more complex, good for learning object-oriented concepts)
  - **Example:** "You are in a dark forest. Do you go left (L) or right (R)?" Based on the user's input ('L' or 'R'), the program progresses the story.
  - **Progression:** Add inventory management, combat, puzzles, and multiple endings.
- **2. Simple Calculator:**
  - **Concept:** Build a calculator that can perform basic arithmetic operations (+, -, *, /).
  - **Skills Reinforced:**
    - `input/output`
    - Variables (numbers, operators)
    - `if/else` or `switch` statements (to determine which operation to perform)
    - Error handling (division by zero)
  - **Languages:** Python, Java, C++ (if you want to learn about data types in more detail)
  - **Progression:** Add more advanced functions (square root, exponents, trigonometric functions), handle more complex expressions (using order of operations).
- **3. Number Guessing Game:**
  - **Concept:** The computer generates a random number, and the user has to guess it. The program provides feedback ("Too high," "Too low").
  - **Skills Reinforced:**
    - Random number generation
    - `input/output`
    - Loops (`while` loop to keep the game going)
    - `if/else` statements (for feedback)
  - **Languages:** Python, Java, C++

Intermediate (Building on fundamentals, introducing more complex concepts):

- **4. To-Do List Application:**
  - **Concept:** Create a program that allows the user to add, remove, and view tasks in a to-do list.

- **Skills Reinforced:**
  - Data structures (lists, arrays)
  - `input/output`
  - Loops
  - Functions (to organize the application logic)
  - File I/O (optional, to save the to-do list to a file)
- **Languages:** Python, Java, C++
- **Progression:** Add features like task priorities, due dates, categories, and the ability to mark tasks as complete. You could also explore using a GUI library (like Tkinter in Python or Swing in Java) to make it a visual application.

- **5. Simple Web Scraper:**
  - **Concept:** Write a program that retrieves data from a website. For example, scrape headlines from a news website.
  - **Skills Reinforced:**
    - HTTP requests (getting data from the web)
    - HTML parsing (extracting specific information from HTML code)
    - Regular expressions (optional, for more complex pattern matching)
    - Error handling
  - **Languages:** Python (using libraries like `requests` and `Beautiful Soup`), Java
  - **Important Note:** Be ethical and respectful of websites. Check the website's `robots.txt` file to see if web scraping is allowed, and don't overload the server with too many requests.
- **6. Basic Game (e.g., Hangman, Tic-Tac-Toe):**
  - **Concept:** Implement a simple game with more complex logic than the number guessing game.
  - **Skills Reinforced:**
    - Arrays/Lists
    - Game logic implementation
    - `input/output`
    - Function decomposition
    - (If GUI) Basic event handling
  - **Languages:** Python (with Pygame or Tkinter), Java (with Swing), C++ (with SDL or SFML)
  - **Progression:** Add AI opponents (e.g., a Tic-Tac-Toe AI), improve the user interface, and add scoring.

Advanced (Challenging projects that introduce more advanced topics):

- **7. Chatbot:**
  - **Concept:** Create a simple chatbot that can respond to user input.
  - **Skills Reinforced:**
    - Natural Language Processing (NLP) concepts (tokenization, stemming, intent recognition - even at a basic level)
    - Data structures (dictionaries for storing question-answer pairs)
    - String manipulation
    - API integration (optional, to connect to more advanced NLP services)
  - **Languages:** Python (using libraries like NLTK or spaCy), Java
  - **Progression:** Train the chatbot on a larger dataset of conversations, add more sophisticated NLP techniques, and integrate it with a messaging platform.
- **8. Data Analysis Project:**
  - **Concept:** Analyze a dataset (e.g., from Kaggle or a government website) to find trends and insights.
  - **Skills Reinforced:**
    - Data analysis libraries (e.g., Pandas in Python)
    - Data visualization (e.g., Matplotlib or Seaborn in Python)
    - Statistical concepts (mean, median, standard deviation)
    - Data cleaning and preprocessing
  - **Languages:** Python (the most common choice for data analysis)
  - **Progression:** Explore more complex statistical models, build machine learning models, and create interactive dashboards to visualize the data.
- **9. Contribute to an Open-Source Project:**
  - **Concept:** Find a small, well-documented open-source project that you're interested in and contribute a bug fix or a new feature.
  - **Skills Reinforced:**
    - Version control (Git)
    - Collaboration
    - Reading and understanding existing code
    - Following coding standards
  - **Languages:** Depends on the project you choose.
  - **How to find projects:** Look on GitHub for projects labeled with "good first issue" or "help wanted".

General Tips for Success:

- **Start Small:** Don't try to build the next Facebook right away. Break down your projects into smaller, manageable tasks.
- **Plan Before You Code:** Before you start writing code, take some time to think about the overall structure of your program and how the different parts will interact. Use pseudocode or flowcharts to help you visualize your program.
- **Learn to Debug:** Debugging is a crucial skill. Learn how to use debugging tools (like print statements, debuggers) to find and fix errors in your code.
- **Use Online Resources:** Don't be afraid to Google your questions! There are tons of online resources available to help you learn programming, including tutorials, documentation, and forums. Stack Overflow is your friend.
- **Practice Regularly:** The more you practice, the better you'll become at programming. Try to code every day, even if it's just for a few minutes.
- **Ask for Help:** Don't be afraid to ask for help from teachers, classmates, or online communities. Explain your problem clearly and provide the code you've tried.
- **Stay Motivated:** Programming can be challenging, but it's also very rewarding. Choose projects that you're interested in, and celebrate your successes along the way.

To help me give you even better recommendations, tell me:

- **What languages are you already familiar with?**
- **What kind of projects appeal to you (games, web development, data analysis, etc.)?**

Good luck, and have fun!

Classification tasks reduces output variability

The prompt below results in a choice and may be useful if you want the output to be easier to control.

[16]:
```
prompt = """I'm a high school student. Which of these activities do you suggest and why:
a) learn Python
b) learn JavaScript
c) learn Fortran
"""

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Okay, as a high school student looking to learn a programming language, I highly recommend **a) Learn Python**. Here's why:

Python is the Best Choice for Most High School Students:

- **Beginner-Friendly:** Python's syntax is designed to be readable and easy to understand. It uses clear English-like keywords, which makes it much easier to pick up the fundamentals of programming without getting bogged down in complex syntax.
- **Versatile and Widely Used:** Python is used in a HUGE range of fields:
  - **Data Science:** Analyzing data, creating visualizations, machine learning, and AI are all areas where Python dominates. This is a very hot and growing field.
  - **Web Development (Backend):** Building the server-side logic for websites and web applications.
  - **Game Development:** Prototyping games, creating game tools, and even building simple games.
  - **Scripting and Automation:** Automating tasks on your computer, writing scripts to manage files, etc.
  - **Education:** Many universities use Python as the introductory language for Computer Science programs.
- **Large and Supportive Community:** There's a massive online community of Python developers. This means if you run into problems, you can easily find help on forums, Stack Overflow, Reddit, and other online resources. There are also tons of tutorials, courses, and documentation available.
- **Good for Academic Projects:** Python's versatility makes it great for school projects. You can use it for science simulations, data analysis for history projects, creating interactive presentations, and more.

Why not JavaScript or Fortran?

- **JavaScript (b):**
  - **Good for Web Development (Frontend):** JavaScript is essential for making websites interactive and dynamic (what you see and interact with in your browser).
  - **Can be more challenging for beginners:** While JS is very powerful, its concepts like the DOM (Document Object Model) and asynchronous programming can be a bit harder to grasp initially compared to Python. It's often better to learn the fundamentals of programming first with a more straightforward language.
  - **Best learned after a solid foundation:** JavaScript is a great language to learn *after* you have a good understanding of programming concepts with a language like Python.
- **Fortran (c):**
  - **Specialized for Scientific and Engineering Computing:** Fortran is an older language that's still used in very specific niches, primarily high-performance scientific and engineering calculations.
  - **Not as widely applicable:** Unless you know you want to go into a very specialized field where Fortran is common (like climate modeling or computational fluid dynamics), it's not the most practical choice for general programming skills.
  - **Smaller Community:** The community is smaller than Python or JavaScript.

In summary:

- **Learn Python first.** It will give you a solid foundation in programming principles and open doors to many different areas.
- Once you're comfortable with Python, **then consider learning JavaScript** if you're interested in web development.
- **Only learn Fortran** if you have a specific need for it in a scientific or engineering context.

Good luck with your programming journey!

Improve response quality by including examples

Another way to improve response quality is to add examples in your prompt. The LLM learns in-context from the examples on how to respond. Typically, one to five examples (shots) are enough to improve the quality of responses. Including too many examples can cause the model to over-fit the data and reduce the quality of responses.

Similar to classical model training, the quality and distribution of the examples is very important. Pick examples that are representative of the scenarios that you need the model to learn, and keep the distribution of the examples (e.g. number of examples per class in the case of classification) aligned with your actual distribution.

Zero-shot prompt

Below is an example of zero-shot prompting, where you don't provide any examples to the LLM within the prompt itself.

[17]:
```
prompt = """Decide whether a Tweet's sentiment is positive, neutral, or negative.

Tweet: I loved the new YouTube video you made!
Sentiment:
"""

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Sentiment: Positive

One-shot prompt

Below is an example of one-shot prompting, where you provide one example to the LLM within the prompt to give some guidance on what type of response you want.

[18]:
```
prompt = """Decide whether a Tweet's sentiment is positive, neutral, or negative.

Tweet: I loved the new YouTube video you made!
Sentiment: positive

Tweet: That was awful. Super boring 😠
Sentiment:
"""

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Sentiment: negative

Few-shot prompt

Below is an example of few-shot prompting, where you provide a few examples to the LLM within the prompt to give some guidance on what type of response you want.

[19]:
```
prompt = """Decide whether a Tweet's sentiment is positive, neutral, or negative.

Tweet: I loved the new YouTube video you made!
Sentiment: positive

Tweet: That was awful. Super boring 😠
Sentiment: negative

Tweet: Something surprised me about this video - it was actually original. It was not the same old recycled stuff that I always see. Watch it - you will not
Sentiment:
"""

response = client.models.generate_content(model=MODEL_ID, contents=prompt)
display(Markdown(response.text))
```

Sentiment: positive

Choosing between zero-shot, one-shot, few-shot prompting methods

Which prompt technique to use will solely depends on your goal. The zero-shot prompts are more open-ended and can give you creative answers, while one-shot and few-shot prompts teach the model how to behave so you can get more predictable answers that are consistent with the examples provided.

[ ]: