

Building End-to-end Solutions with Gemini

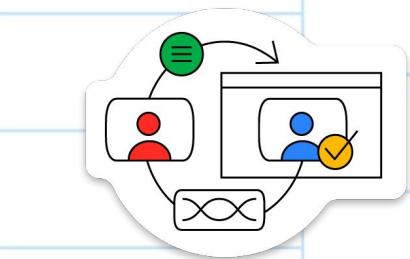
Multimodal approaches to solving
problems with AI



Karthik Muthuswamy He/Him

ML GDE

He / Him



Get started!

Visit the cookbook: bit.ly/gemini-cookbook

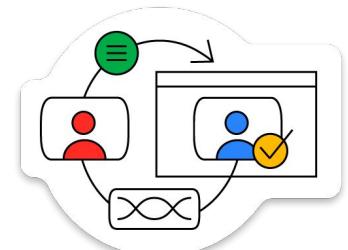
Start developing

1. Go to [Google AI Studio](#).
2. Login with your Google account.
3. [Create](#) an API key.
4. Use a [quickstart](#) for Python, or call the REST API [using curl](#).



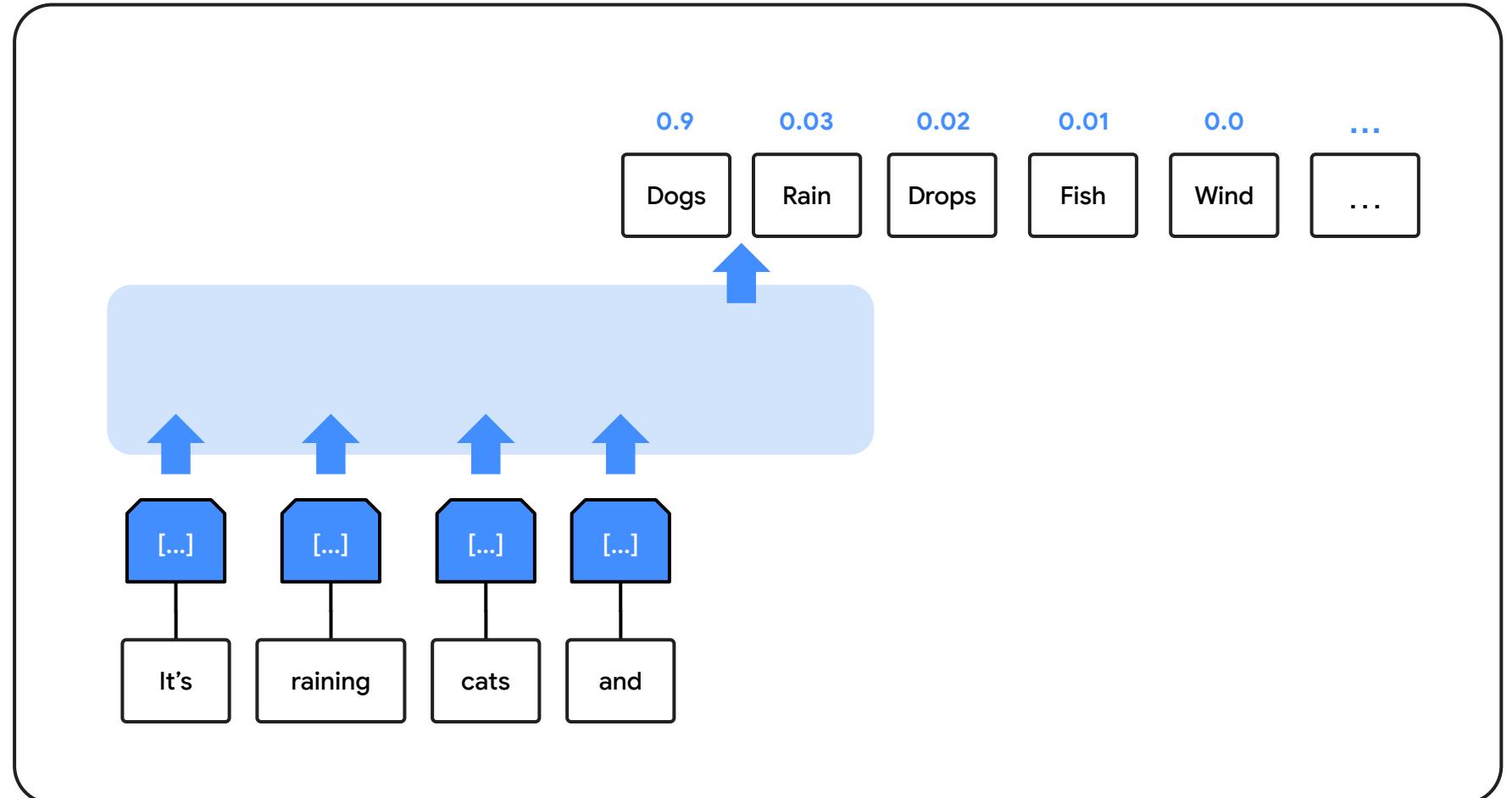
Gemini Cookbook

Read on to learn more about the Gemini API



what is generative ai?

What is an LLM?



Roses are red,

Roses are red,
Violets are blue,
Sugar is sweet,

```
for(var i = 0;
```

```
for(var i = 0; i < 10; i++) {
```

Modern LLMs are
large.

Classic Natural Language Problems

 Entity extraction

 Classification

 Summarization

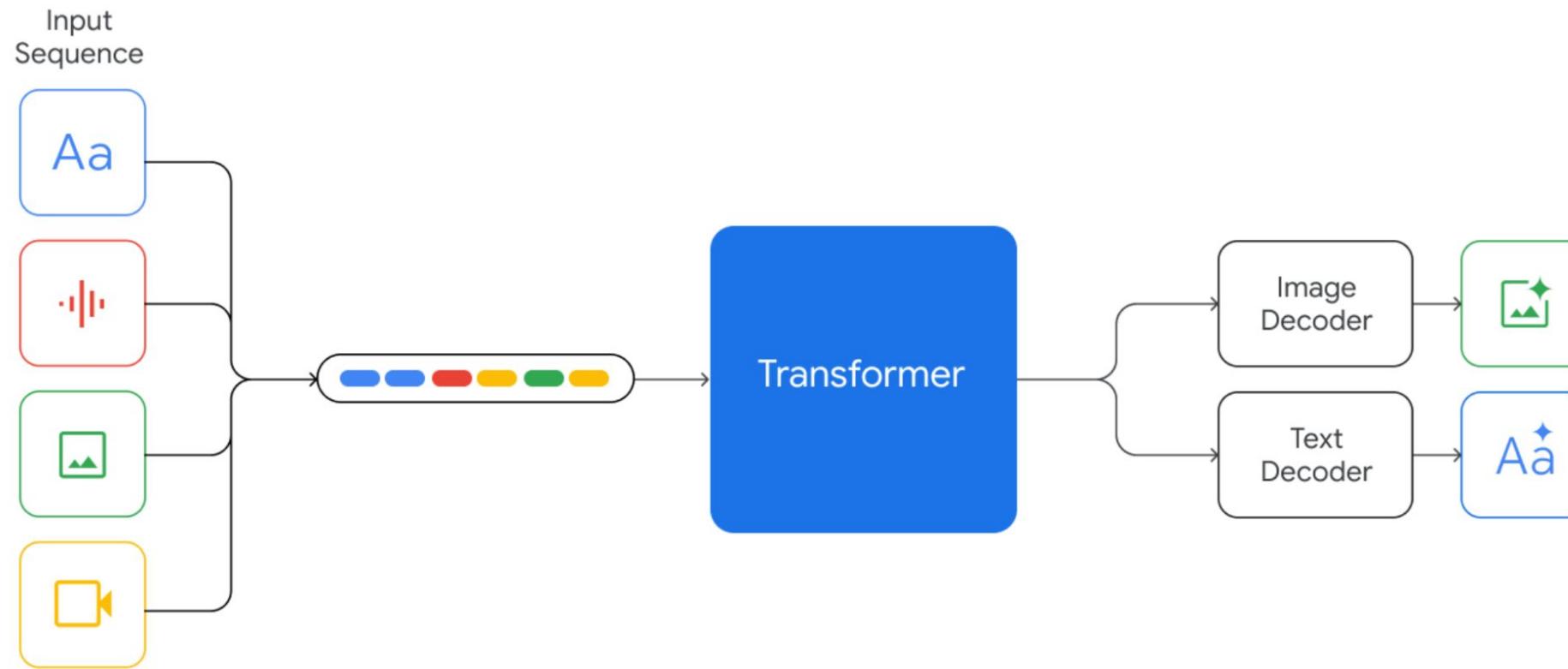
 Sentiment Analysis

 Translation

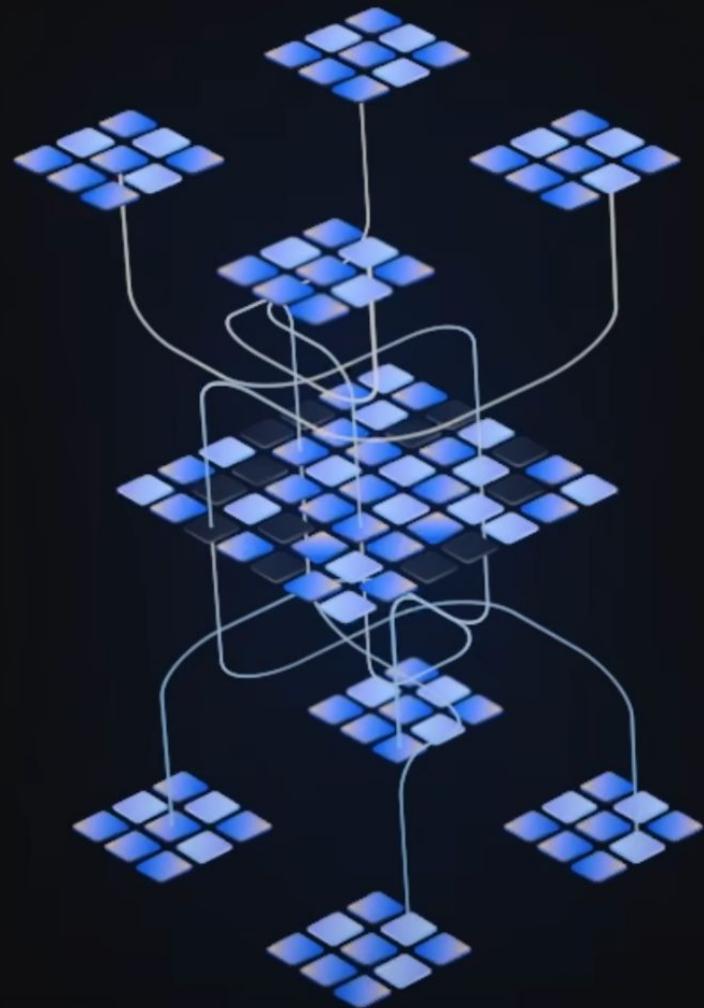
 ...

LLMs let us
prototype *fast*.

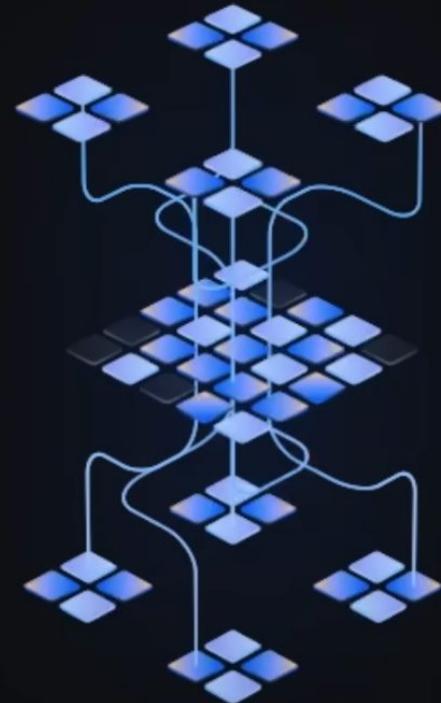
Multimodality



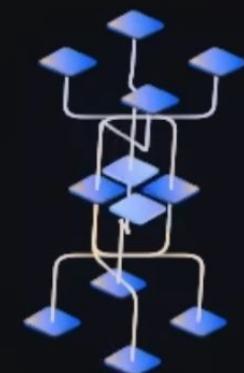
Welcome to
the Gemini era



Ultra



Pro



Nano

(Android AICore)

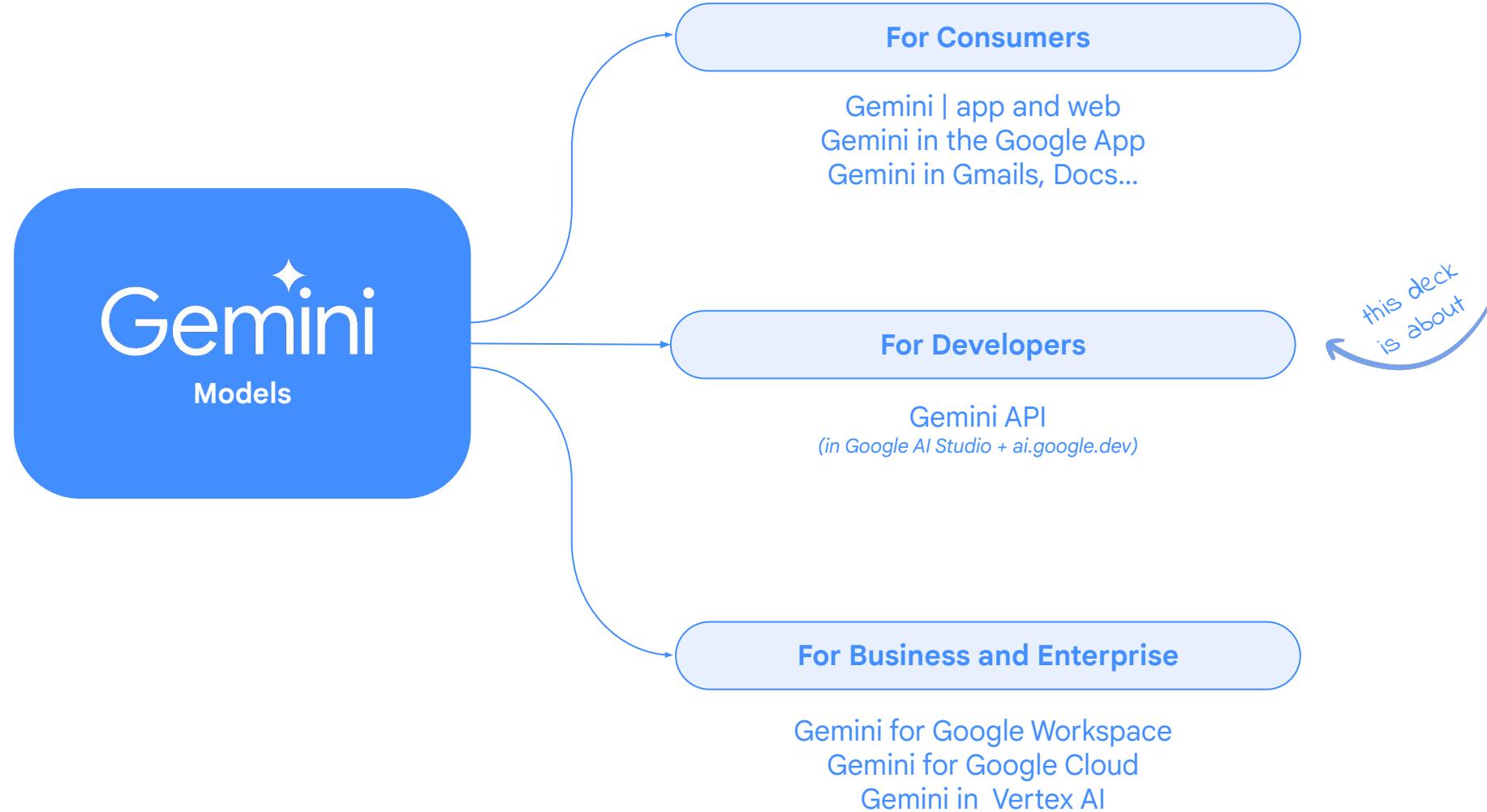


Gemma Open Models

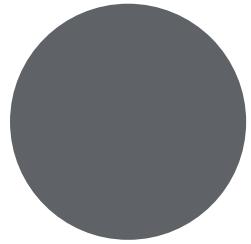
ai.google.dev/gemma

The Gemini Ecosystem

The most advanced AI from Google

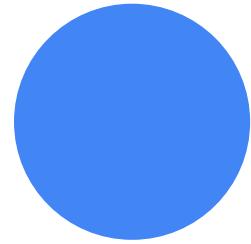


Evolution of AI Use Cases



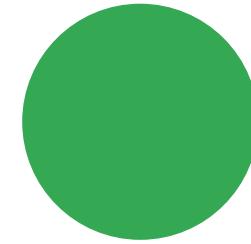
Predictive AI

Regression & Classification
Forecasting
Sentiment Analysis
Entity Extraction
Object Detection



Generative AI

Text, Image & Code Generation
Text & Code Rewriting & Formatting
Summarization
Extractive Q&A
Image & Video Descriptions



Multimodal Generative AI

Natural Image Understanding
Spatial Reasoning and Logic
Mathematical Reasoning in Visual Contexts
Video Question Answering
Automatic Speech Recognition & Translation

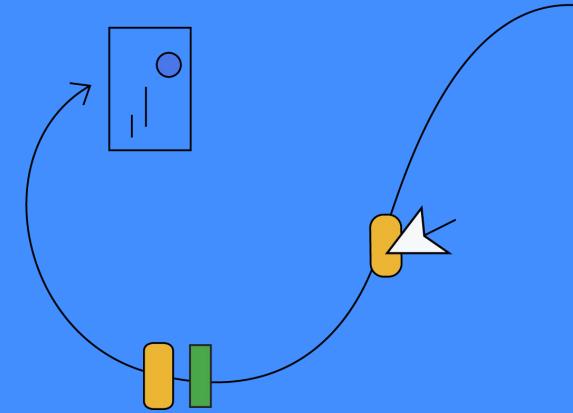
RAG

Function Calling

Extensions Grounding

Punting & Safety

Getting started with the Gemini API



aistudio.google.com

- Generate API Keys
- Create, test, and save prompts
- Customize models in minutes
- Generate starter code

Google AI Studio

Untitled prompt edit

Insert: Image Video File Folder Test input

Run settings Reset

Model
Gemini Pro Vision

Temperature
0.4

Add stop sequence
Add stop...

Safety settings
Edit safety settings

Advanced settings

Get API key Save Get code ⋮

Suggest a meal plan inspired by these 3 dishes



Getting started Documentation Prompt gallery Discord community

Build with Vertex AI on Google Cloud

Settings

macd@google.com

Run Preview 0 / 12288

AI Studio

Google AI Studio

Untitled prompt

Save Get code :

Insert: Image Video File Folder Test input

Run settings Reset

Suggest a meal plan inspired by these 3 dishes

Model: Gemini Pro Vision

Temperature: 0.4

Add stop sequence: Add stop...

Safety settings: Edit safety settings

Advanced settings

Get API key Create new My library

Getting started Documentation Prompt gallery Discord community

Build with Vertex AI on Google Cloud

Settings

macd@google.com

Run Preview 0 / 12288

aistudio.google.com

Google AI Studio

Untitled prompt

Save Get code :

Insert: Image Video File Folder Test input

Run settings Reset

Model: Gemini Pro Vision

Temperature: 0.4

Add stop sequence: Add stop...

Safety settings: Edit safety settings

Advanced settings

Suggest a meal plan inspired by these 3 dishes



Tab

Getting started Documentation Prompt gallery Discord community

Build with Vertex AI on Google Cloud

Settings

macd@google.com

Run Preview 0 / 12288

AI Studio

Google AI Studio

Untitled prompt edit

Save Get code ⋮

Insert: Image Video File Folder Test input

Run settings Reset

Suggest a meal plan inspired by these 3 dishes



Model: Gemini Pro Vision

Temperature: 0.4

Add stop sequence: Add stop...

Safety settings: Edit safety settings

Advanced settings

Getting started Documentation Prompt gallery Discord community

Build with Vertex AI on Google Cloud

Settings

macd@google.com

Run Preview 0 / 12288

aistudio.google.com

Get code X

Create your API key before using the code in your project

You can call this prompt from the Gemini API by copying the following code into your project

cURL JavaScript **Python** Android (Kotlin) Swift Open in Colab Copy

```
1 """
2 At the command line, only need to run once to install the package via pip:
3
4 $ pip install google-generativeai
5 """
6
7 import google.generativeai as genai
8
9 genai.configure(api_key="YOUR_API_KEY")
10
11 # Set up the model
12 generation_config = {
13     "temperature": 2,
14     "top_p": 0.4,
15     "top_k": 50
```

SDKs

ai.google.dev

REST API + Client libraries for Python, Node, Java, and Swift



Vertex AI

Gemini API and Vertex AI

Both give access Gemini family models.

Vertex AI

Enterprise grade support.
Full MLOps (Examples: Model evaluation, monitoring, registry)



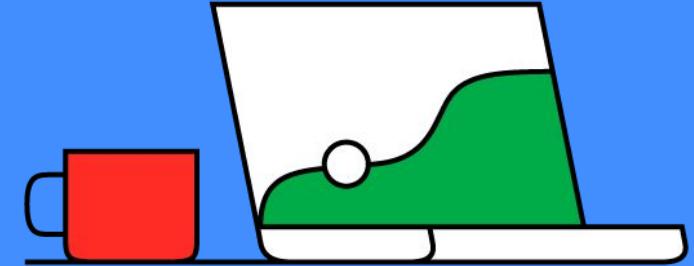
Vertex AI

Check it out when you're ready for production



[ai.google.dev/docs/migrate to cloud](https://ai.google.dev/docs/migrate_to_cloud)

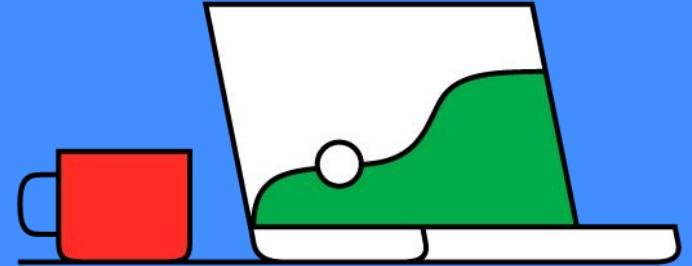
Usage patterns



We'll cover

- Basic Prompting
- Code generation
- Search, retrieval and information synthesis
- Function calling
- Image understanding
- Tuning

Basic prompting



aistudio.google.com

Write a poem about a magic backpack.

aistudio.google.com

Write a poem about a magic backpack.

In a world of wonder, behold,
A backpack with tales yet untold.
Its fabric shimmers, alive with might,
A portal to realms, day and night.

...

Prompting

Generate a short tip about JavaScript based on the provided topic. Good tips are short and contain a code example showing the tip.

3 / 500 examples

Actions ▾

INPUT topic:

: OUTPUT tip:

⋮

let and const

Prefer using 'let' and 'const' to initialise variables, instead of 'var', as 'var' has no block scope or protection against mutability.

e.g.

```
const myVar = "EXAMPLE";
```

arrow syntax

Use the arrow syntax `=>` instead of defining anonymous functions. Your code will be clearer and easier to read.

e.g.

```
const doubles = input.map(x => x + x);
```

template strings

Template strings allow you to build multi-line strings using a template and variables from the outer scope.

e.g.

```
const emailHeader = `To: ${recipient}\nFrom: ${sender}\nSubject: Welcome!`;
```

The user's input

The model's response

```
import google.generativeai as genai

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

resp = model.generate_content(
    'Write the first paragraph of a story about a magic backpack')

>>> print(resp.text)
In a bustling city, amidst the vibrant tapestry of human
existence, there existed a peculiar entity named Archie. Archie,
however, was no ordinary backpack...
```

```
chat = model.start_chat()

response = chat.send_message(
    "Hello, what should I have for dinner?")

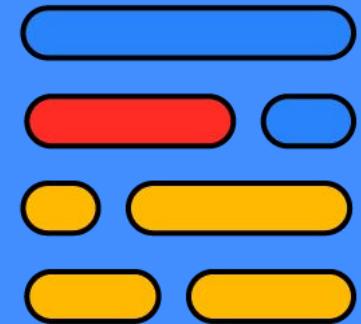
print(response.text)
    # 'Here are some suggestions...'

response = chat.send_message(
    "How do I cook the first one?")
```

- Chained prompts - make a plan, then execute it
- Context - Few shot prompts
- Generation parameters - Temperature, Safety settings, Top-P, TopK

https://ai.google.dev/docs/prompt_best_practices

Code generation



Content generation

Code Generation

Generate code based on a user prompt

Convert human language requests to machine requests

Rapid tool building

Examples

- Generate data

Insert: Image Video File Folder

Chat

C



User

Generate a table that has all the countries in the world listed alphabetically, their capitals, and what continent they belong to.

Model



Country	Capital	Continent
Afghanistan	Kabul	Asia
Albania	Tirana	Europe
Algeria	Algiers	Africa
Andorra	Andorra la Vella	Europe
Angola	Luanda	Africa
Antigua and Barbuda	Saint John's	North America
Argentina	Buenos Aires	South America
Armenia	Yerevan	Asia

Examples

- Generate data
- Generate a SQL query



Generate a SQL query that will query for capitals in countries of Asia, following this format:
Table countries, columns = [Country]
Table capitals, columns = [CountryID, CapitalID, Capital]

Model

```
SELECT c.Capital
FROM Capitals c
JOIN Countries co ON c.CountryID = co.CountryID
WHERE co.ContinentID = (
    SELECT ContinentID
    FROM Continents
    WHERE Continent = 'Asia'
);
```



Use code with caution.

SQL

Examples

- Generate data
- Generate a SQL query
- Simulate Execution

What is the output from this command and table?

Model



The output of the following command:

```
SELECT c.Capital
FROM Capitals c
JOIN Countries co ON c.CountryID = co.CountryID
WHERE co.ContinentID = (
    SELECT ContinentID
    FROM Continents
    WHERE Continent = 'Asia'
);
```

Use code with caution.

SQL

when executed against the provided table data will be:

Kabul
Yerevan
Dhaka

Examples

- Generate data
- Generate a SQL query
- Simulate Execution
- Quick coding help - Smart pair programmer.

Insert:  Image  Video  File  Folder

Chat



Generate a Docker script to create a simple linux machine that has python 3.10 installed with following libraries: pandas, tensorflow, numpy

Model



```
# Create a new Dockerfile
FROM ubuntu:20.04

# Update the package manager
RUN apt-get update -y

# Install Python 3.10
RUN apt-get install -y python3.10

# Install necessary libraries
RUN pip install pandas tensorflow numpy

# Set the default Python version to 3.10
RUN update-alternatives --install /usr/bin/python python /usr/bin/python3.10 1

# Copy the user's code to the container
WORKDIR /usr/src/app
COPY . .
```

Write a unit test for this Python function:

```
def is_diver_neutrally_buoyant(density_water, density_diver):  
    """Check if a diver is neutrally buoyant."""  
    if density_water == density_diver:  
        return True
```

```
import unittest

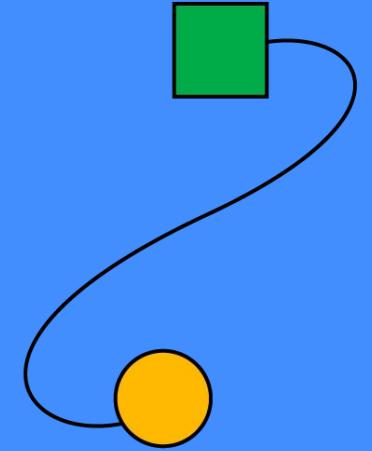
class IsDiverNeutrallyBuoyantTest(unittest.TestCase):

    def test_diver_is_neutrally_buoyant(self):
        self.assertTrue(is_diver_neutrally_buoyant(1000, 1000))

    def test_diver_is_not_neutrally_buoyant(self):
        self.assertFalse(is_diver_neutrally_buoyant(1000, 1010))

    def test_invalid_input_types(self):
        with self.assertRaises(TypeError):
            is_diver_neutrally_buoyant("1000", 1000)
        with self.assertRaises(TypeError):
            is_diver_neutrally_buoyant(1000, "1000")
```

Search and Information Synthesis



BYO Data

- Models have knowledge cut-offs
- LLMs are not fact engines
- No exposure to private data

Use the prompt's context window

- Instructions + Context + Question all in the prompt
- Easy to implement
 - No extra code, just ask.

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

```
document = pathlib.Path('document.txt').read_text()
```

```
result = model.generate_content(f"""
```

```
Explain how deep-sea life survives.
```

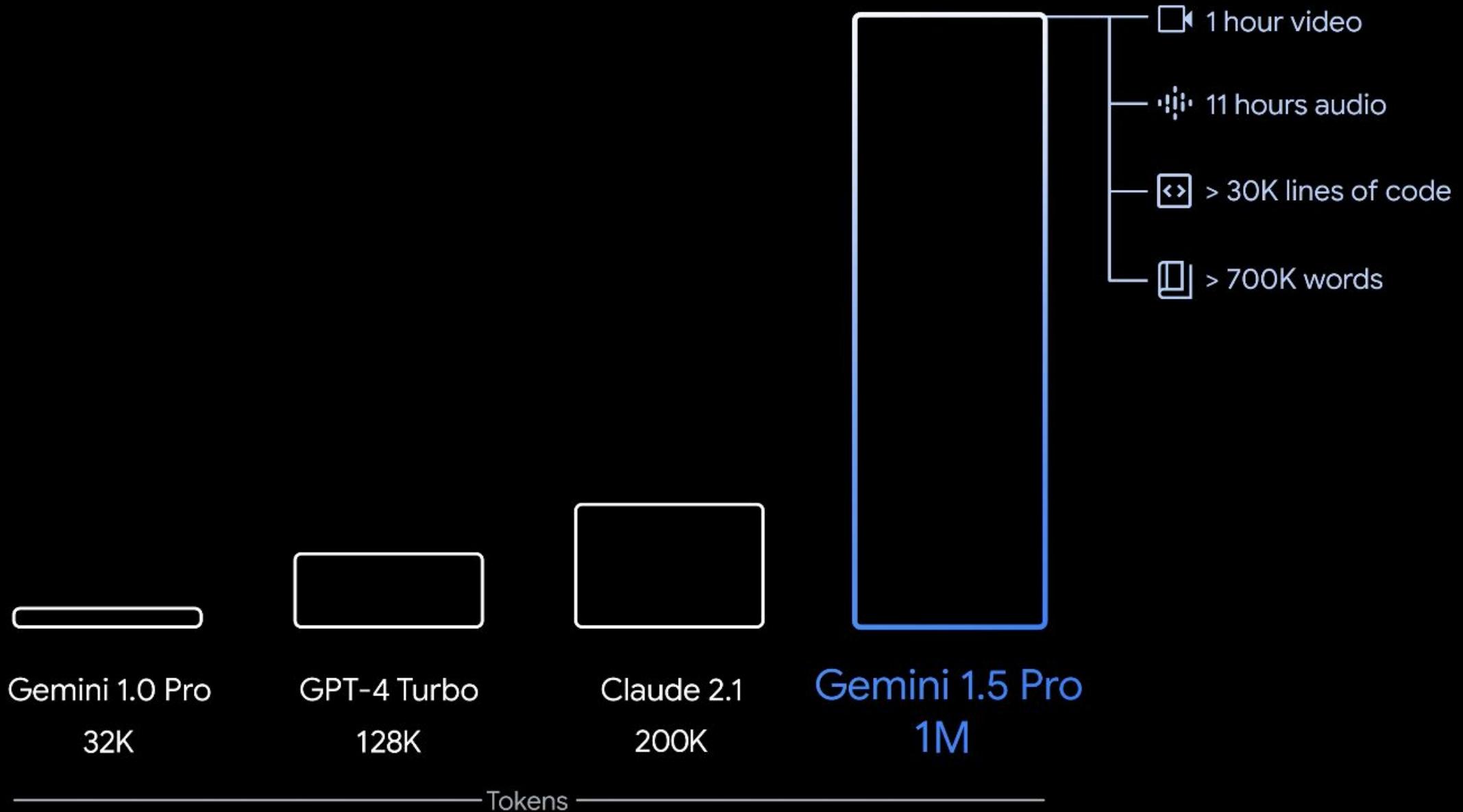
Please answer based on the following document:

```
{document}""")
```

Use the prompt's context window

- Limited by the model's context length
 - gemini-1.0-pro: 30K tokens.

up to
10M
in research



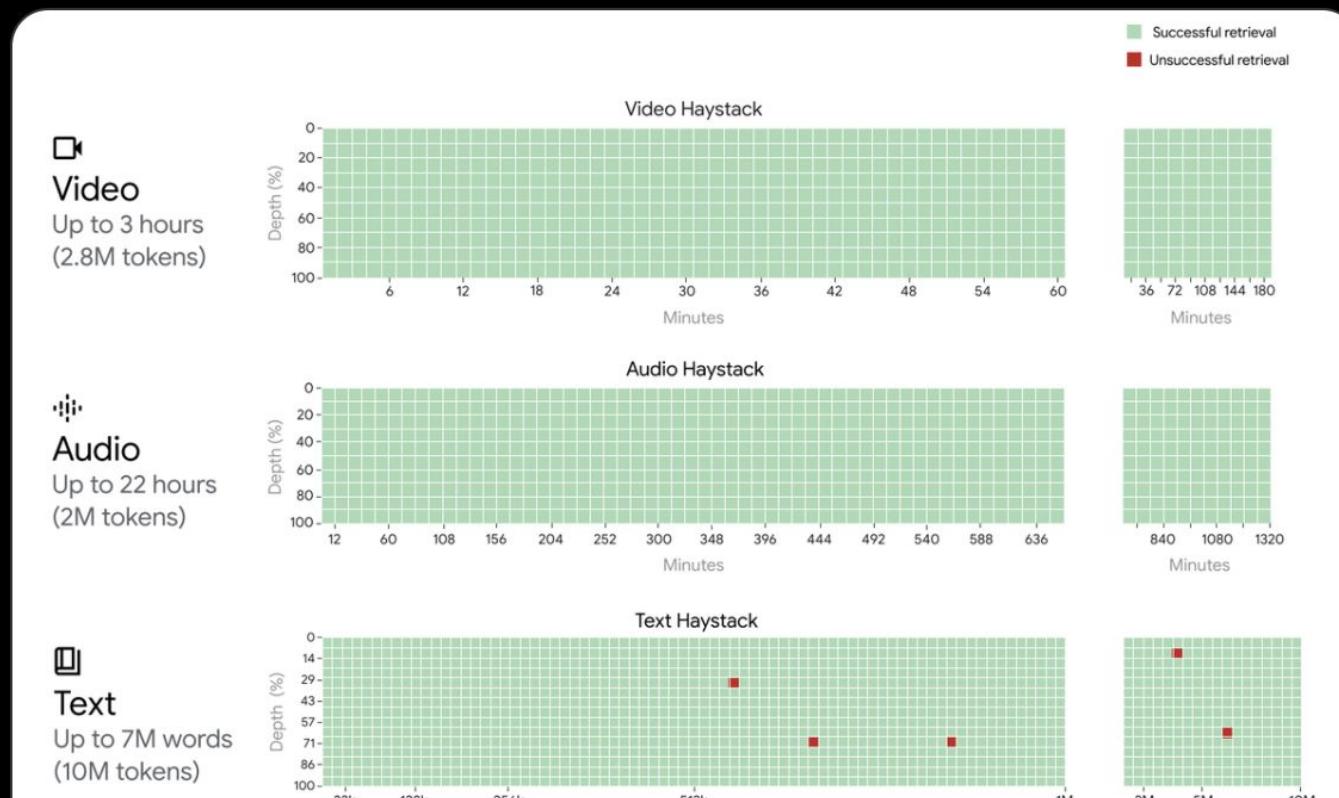


Jeff Dean (@ )
@JeffDean

...

Needle in a Haystack Tests Out to 10M Tokens

First, let's take a quick glance at a needle-in-a-haystack test across many different modalities to exercise Gemini 1.5 Pro's ability to retrieve information from its very long context. In these tests, green is good, and red is not good, and these are almost entirely green (>99.7% recall), even out to 10M tokens. Great! A bit more on needle-in-a-haystack tests later in the thread.

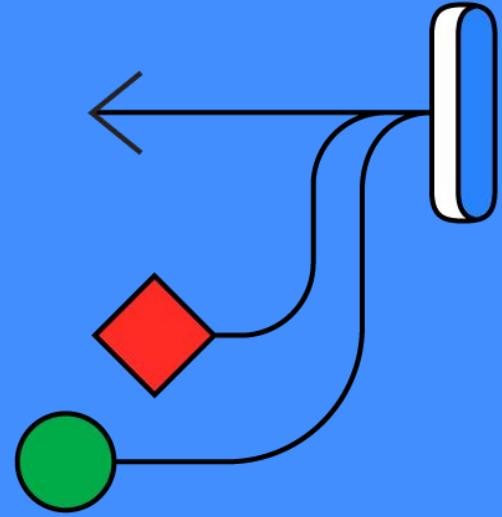


Use the prompt's context window

- **gemini-1.5-pro: 1M tokens.**
 - Remember: Everything is tokens.
- Using context may be more flexible than Retrieval Augmented Generation (RAG).
- More input context means higher latency
- Join the waitlist:

<https://aistudio.google.com/app/waitlist/97445851>

Function Calling





What percentage of orders are returned?



What percentage of orders are returned?



9.95% of orders are returned. This information comes from the `thelook_ecommerce.orders` table in BigQuery. The query used to calculate the return rate was:

```
SELECT ROUND(  
  COUNT(DISTINCT IF(returned_at IS NOT NULL, order_id, NULL)) /  
  COUNT(DISTINCT order_id)) * 100, 2)  
AS return_rate  
FROM thelook_ecommerce.orders
```

Function Calling

- Describe external functions to the model.
- The model may ask **you** to call the function to help it respond to your queries.
- Endless possibilities for integrating external tools.

Function calling - Basics

- How?
- The `google.generativeai` SDK will **inspect the function's type hints to determine the schema**.
- Allowed types are limited:

```
AllowedTypes = (  
    int | float | str | bool | list | dict )
```

```
def multiply(a: float, b: float):  
    """Returns a * b."""  
    return a*b  
  
model = genai.GenerativeModel(  
    model_name='gemini-1.0-pro' ,  
    tools=[multiply])
```

Function calling - Basics

- Because function calling requires alternating turns, it's easiest to use through chat.
- Enable "automatic function calling" when you start a chat, and the **ChatSession** will call the function(s) for you.
 - You don't have to use automatic function calling, it just makes simple cases easier.

```
chat = model.start_chat(  
    enable_automatic_function_calling=True)  
  
response = chat.send_message(  
    'I have 57 cats, each owns 44 mittens, '  
    'how many mittens is that in total?')
```

```
chat = model.start_chat(  
    enable_automatic_function_calling=True)  
  
response = chat.send_message(  
    'I have 57 cats, each owns 44 mittens, '  
    'how many mittens is that in total?')  
  
print(response.text)  
# The number of mittens in total is 2508.
```

Function calling - Basics

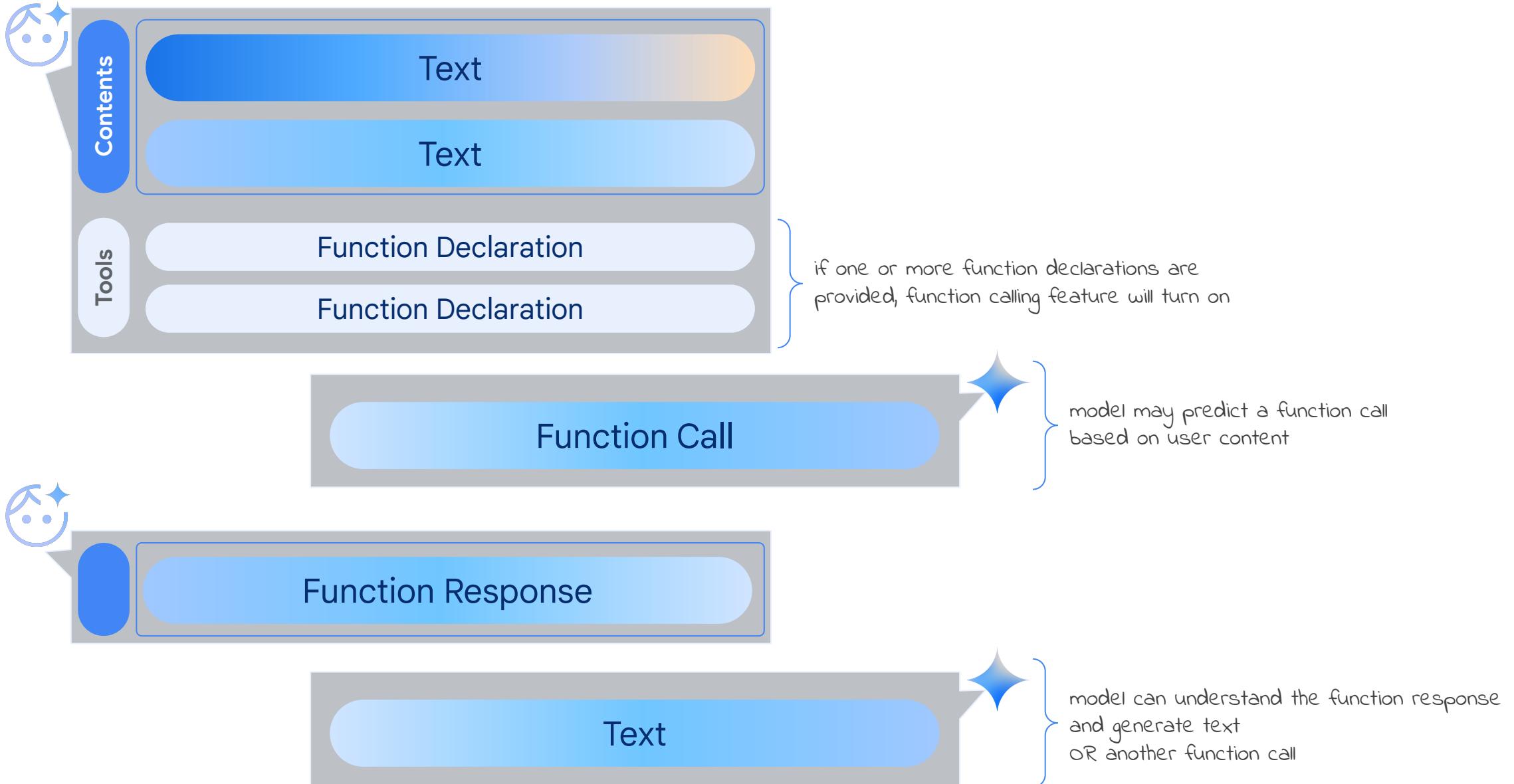
- What happened? Use the chat history to find out.
- The chat history collects all the function calls and responses that took place.

```
for content in chat.history:  
    part = content.parts[0]  
    print(content.role, "->", type(part).to_dict(part))
```

```
for content in chat.history:
    part = content.parts[0]
    print(content.role, "->", type(part).to_dict(part))

# user -> {'text': 'I have 57 cats, each owns 44 mittens, '
#           'how many mittens is that in total?'}
# model -> {'function_call': {'name': 'multiply',
#                             'args': {'a': 57.0, 'b': 44.0}}}
# user -> {'function_response': {'name': 'multiply',
#                                'response': {'result': 2508.0}}}
# model -> {'text': ' The number of mittens in total is 2508.'}
```

Function Calling interaction



Function calling - More Examples

- Wikipedia research aid
 - Integrates a search tool.
 - Uses the Gemini API inside the function call to summarize pages.

```
def wikipedia_search(queries:list[str]) -> list[str]:  
    ...
```

```
model = genai.GenerativeModel('gemini-pro', tools=[wikipedia_search])

chat = model.start_chat(enable_automatic_function_calling=True)

query = "Explain how deep-sea life survives."

res = chat.send_message(instructions.format(query=query))
```

```
model = genai.GenerativeModel('gemini-pro', tools=[wikipedia_search])
```

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

```
query = "Explain how deep-sea life survives."
```

```
res = chat.send_message(instructions.format(query=query))
```

```
# Searching for "How do deep-sea creatures survive the extreme pressure?"
```

```
# Related search terms: ['Deep sea', 'Deep-sea community', 'Deep-sea fish']
```

```
# Fetching page: "Deep sea"
```

```
# Information Source: https://en.wikipedia.org/wiki/Deep\_sea
```

```
# Fetching page: "Deep-sea community"
```

```
# Information Source: https://en.wikipedia.org/wiki/Deep-sea\_community
```

```
# Fetching page: "Deep-sea fish"
```

```
# Information Source: https://en.wikipedia.org/wiki/Deep-sea\_fish
```

```
# Searching for "How do deep-sea creatures survive the cold temperatures?"
```

```
# Related search terms: ['Deep-sea community', 'Deep sea', 'Deep-water coral']
```

```
# Fetching page: "Deep-water coral"
```

```
# Information Source: https://en.wikipedia.org/wiki/Deep-water\_coral
```

Function calling - More Examples

- SQL Talk
 - Use function calling to talk to a database.
 - Live example: <https://goo.gle/gemini-fn-call-sql>

```
sql_query_func = FunctionDeclaration(  
    name="sql_query",  
    description="Get information from data in BigQuery using SQL queries",  
    parameters={  
        "type": "object",  
        "properties": {  
            "query": {  
                "type": "string",  
                "description": "SQL query on a single line ...  
    }  
}
```



What percentage of orders are returned?



9.95% of orders are returned. This information comes from the `thelook_ecommerce.orders` table in BigQuery. The query used to calculate the return rate was:

```
SELECT ROUND((  
COUNT(DISTINCT IF(returned_at IS NOT NULL, order_id, NULL)) /  
COUNT(DISTINCT order_id)) * 100, 2)  
AS return_rate  
FROM thelook_ecommerce.orders
```

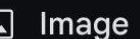
Image understanding



Gemini: Guess the movie



Insert:



Image

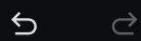


Test input

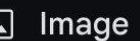


Guess the name of this movie

Gemini: Guess the movie



Insert:



Image



{} {}

Test input



Guess the name of this movie

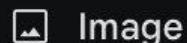
Charlie and the Chocolate Factory

Gemini: Guess the movie

 Save 



Insert:



Image

 Test input



Guess the name of the movie. Explain your reasoning.

Gemini: Guess the movie

 Save 



Insert:  Image  Test input



Guess the name of the movie. Explain your reasoning.

The movie is Home Alone.

The first emoji is a house, which could represent the house that Kevin is left home alone in. The second emoji is a scared face, which could represent Kevin's reaction when he realizes he is home alone. The third emoji is a Christmas tree, which could represent the Christmas season in which the movie is set.

Gemini: Guess the book ↗

Insert:  Image  Video  File  Folder  Test input



Guess the name of this book?

Gemini: Guess the book ↗

Insert:  Image  Video  File  Folder  Test input



Guess the name of this book?

The Secret Garden

Image understanding

- Images are just tokens in the input
- Can be used for instructions, context or query subject

```
import google.generativeai as genai

# Use the Gemini vision model.
PRO_VISION = 'models/gemini-pro-vision'
model = genai.GenerativeModel(PRO_VISION)
```

```
!wget -O instrument.jpg -q https://google/instrument-img
```

```
import PIL.Image
img = PIL.Image.open('instrument.jpg')

# Preview the image
(thumb := img.copy()).thumbnail((200, 200))
thumb
```



```
response = model.generate_content(  
    ['What instrument is this?',  
     img,  
     'What kinds of music would use it?'])  
  
print(response.text)
```

This is a pipe organ. It is a musical instrument that produces sound by driving pressurized air through pipes. Organs are often used in churches, concert halls, and other large venues. They can be used to play a wide variety of music, from classical to contemporary.

Image understanding

- Structured data extraction
- Image conditioning
- *RAG*

Image conditioning

- Generate text or structured data from images

Character Sheet Creator 

Insert: Image Video File Folder {{ }} Test input

Write a character statistic sheet based on this character:



Name: Stickmancer
Race: Human
Class: Wizard
Level: 1
Strength: 10
Dexterity: 14
Stamina: 12
Intelligence: 18
Wisdom: 16
Charisma: 10
Equipment: Staff, Dagger, Robe

Structured Data

Dog Walking Application Form

Please provide some information about your dog:

Name: Fido

Breed: Cavoodle

Colour: Black & tan

Age: 3 yrs

Temperament: Friendly Shy Aggressive

And some information about yourself:

Name: Mark

Phone: 0491 570 006

When would you like your dog walked:

Frequency: Once-off Regularly

Days:

Monday:

Morning Afternoon

Tuesday:

Morning Afternoon

Wednesday:

Morning Afternoon

Thursday:

Morning Afternoon

Friday:

Morning Afternoon

Structured Data

Img2Json 

Insert: Image Video File Folder Test input

Please extract each of the fields in this form and provide a JSON representation:

Dog Walking Application Form

Please provide some information about your dog:

Name: Fido
Breed: Cavoodle
Colour: Black & Tan
Age: 3 yrs
Temperament: Friendly Shy Aggressive

And some information about yourself:

Name: Mark
Phone: 0491 570 006

When would you like your dog walked:

Frequency: Once-off Regular
Days:

Monday: Morning Afternoon
Tuesday: Morning Afternoon
Wednesday: Morning Afternoon
Thursday: Morning Afternoon
Friday: Morning Afternoon

```
{  
  "dog": {  
    "name": "Fido",  
    "breed": "Cavoodle",  
    "colour": "Black & Tan",  
    "age": 3,  
    "temperament": "Friendly"  
  },  
  "owner": {  
    "name": "Mark",  
    "phone": "0491 570 006"  
  },  
  "schedule": {  
    "frequency": "Regular",  
    "days": [  
      {  
        "day": "Wednesday",  
        "time": "Morning"  
      }  
    ]  
  }  
}
```

Structured Data

Img2Json Few Shot 

Save

Insert:  Image

Please convert the form into JSON.

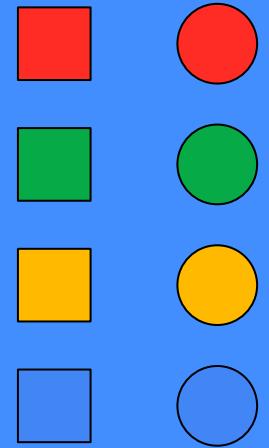
1 / 500 examples Actions ▾

INPUT form:	OUTPUT json:
<input type="checkbox"/> 	<pre>```json { "dog": { "name": "Fido", "breed": "Cavoodle", "colour": "Black & Tan", "age": 3, "temperament": "Friendly" }, "owner": { "name": "Mark", "phone": "0491 570 006" }, "schedule": { "frequency": "Regular" } }</pre>

Test your prompt ⓘ

INPUT form:	OUTPUT json:
1 	Run to get output

Tuning



Tuning

Tuning

- In AI Studio

[Get API key](#)[Create new](#)[New tuned model](#)[My library](#)

Loading...

[Getting started](#)[Documentation](#)[Prompt gallery](#)[Discord community](#)[Build with Vertex AI on Google Cloud](#)

Select data for tuning

You can tune a model from an existing structured prompt or create one by importing from Google Sheets or a CSV. Tuning only works with text at this time. We recommend using 100-500 examples. [Tuning Guide.](#)

Structured prompts

[Create a Structured prompt](#)

or

[Import](#)

Provide details

Tuned model name

Description

Choose base model

Model

[models/gemini-1.0-pro-...](#) ▾

[Create a Structured prompt](#)

or

[Import](#)

Import examples

[Tuning](#)

Assign column labels



Use first row as headers



Prefix column names

Header	inputs	outputs
Row 1	1	2
Row 2	3	4

Assign to

New input column

New output column

Avoid importing sensitive or personal information.
Only text is imported. You can add images directly
through the UI.

[Close](#)[Import 15 examples](#)

Choose base model

Model

models/gemini-1.0-pro-...

Advanced settings

Tuning epochs ⓘ

5

Learning rate multiplier ⓘ

1

Batch size ⓘ

4



Tune

Tuned model results

Tuning details

Increment

Model ID: tunedModels/tuning-9x6pbjld1zo

Base model: Gemini 1.0 Pro 001 (Tuning)

Total training time: Less than 1 min

Tuned examples: 20 examples

Epochs: 5

Batch size: 4

Learning rate: 0.001

Loss / Epochs i



Use your tuned model

 Use in freeform prompt

 Use in structured prompt

Tuning

- Using the API:

https://ai.google.dev/tutorials/tuning_quickstart_python



```
1 training_data=[  
2     {  
3         'text_input': '1',  
4         'output': '2',  
5     }, {  
6         'text_input': '3',  
7         'output': '4',  
8     }, ...  
9 ]
```

```
[ ] 1 import random  
2  
3 name = f'Increment-{random.randint(0,10000)}'  
4 operation = genai.create_tuned_model(  
5     # You can use a tuned model here too. Set `source_model="tunedModels/..."`  
6     source_model=base_model.name,  
7     training_data=training_data,  
8     id = name,  
9     epoch_count = 100,  
10    batch_size=4,  
11    learning_rate=0.001,  
12 )
```

You can use the `genai.generate_text` method and specify the name of your model to test your model performance.

```
[ ] 1 model = genai.GenerativeModel(model_name=f'tunedModels/{name}')
```

```
[ ] 1 result = model.generate_content('55')
2 result.text
```

```
→ '56'
```

```
[ ] 1 result = model.generate_content('123455')
2 result.text
```

```
→ '123456'
```

```
[ ] 1 result = model.generate_content('four')
2 result.text
```

```
→ 'five'
```

```
[ ] 1 result = model.generate_content('quatre') # French 4
2 result.text # French 5 is "cinq"
```

```
→ 'cinq'
```

```
▶ 1 result = model.generate_content('III') # Roman numeral 3
2 result.text # Roman numeral 4 is IV
```

```
→ 'IV'
```

```
[ ] 1 result = model.generate_content('七') # Japanese 7
2 result.text # Japanese 8 is /\!
```

```
→ '八'
```

Tuning

- Permissions:
 - There are permissions configuration in the API, but it's not integrated into the SDK yet.

Responsible AI



Run safety settings

Adjust how likely you are to see responses that could be harmful. Content is blocked based on its probability of being harmful.

Block some

Block medium or high probability of being unsafe

Toxic content

Sexual content

Violent content

Dangerous content

Medical content

Restore default settings

Block some

Block medium or high probability of being unsafe

Toxic content

Sexual content

Violent content

Dangerous content

Medical content

Restore default settings

Learning more

ai.google.dev

<https://ai.google.dev/docs/discord>

Learning more

Resources and what's next

ai.google.dev/

Our best resources and models for AI developers

github.com/GoogleCloudPlatform/generative-ai/tree/main/gemini/

Check out the new Gemini folder in our Gen AI Github repo

cloud.google.com/vertex-ai/docs/generative-ai/tutorials

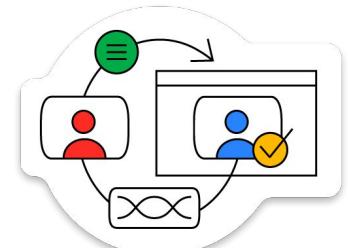
Use new Gemini tutorials to go from idea to app

goo.gle/cloud_community

Join, learn, and engage with the community



Gemini Cookbook



Have fun building with Gen AI



Karthik Muthuswamy He/Him

ML GDE

@krtk

