

04



Implementing the PaLM API

In this module, you learn to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

Program Python applications the use PaLM to generate content

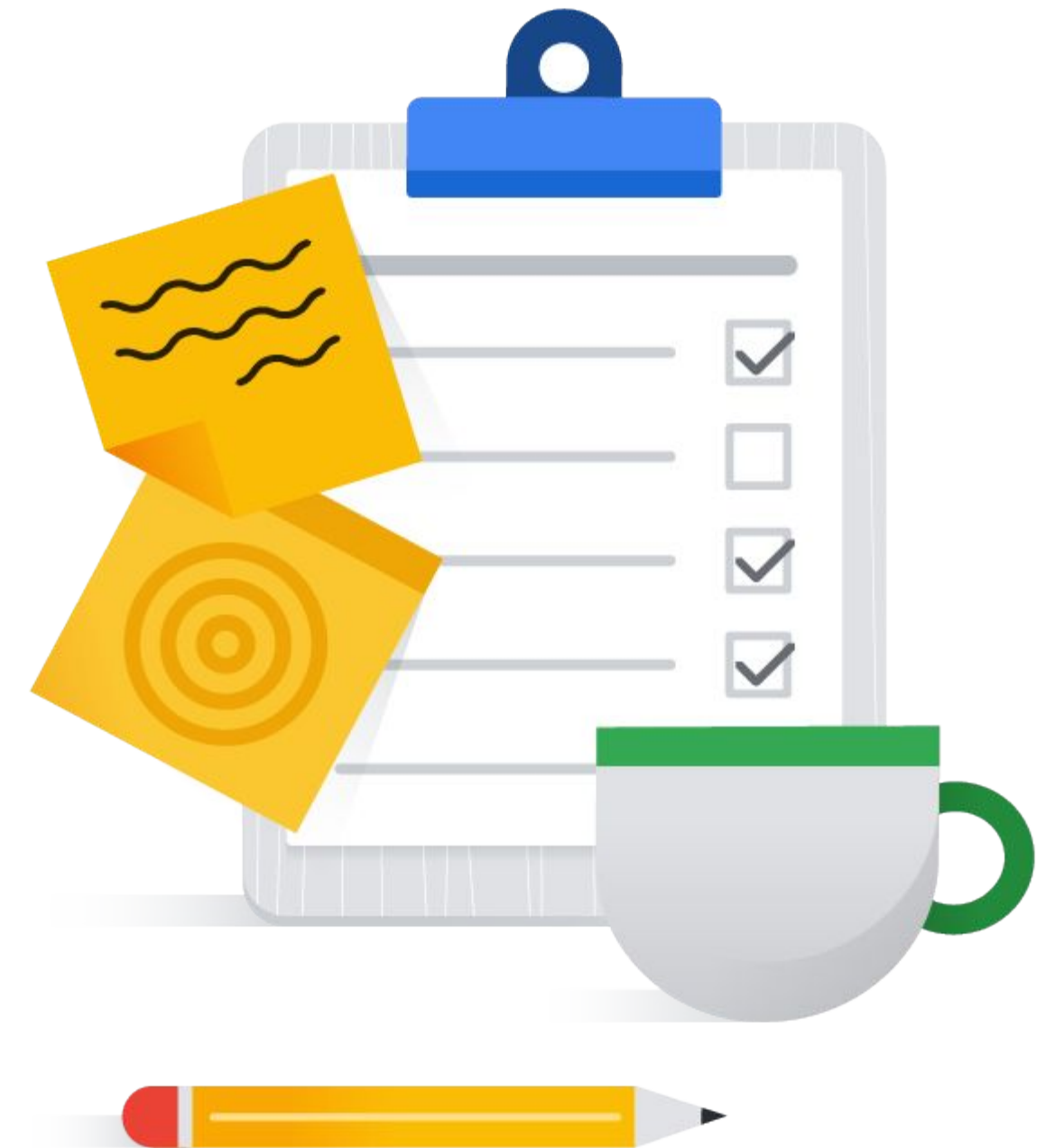
03

Integrate PaLM and GenAI into your applications



Topics

- | | |
|----|--|
| 01 | Introduction to the PaLM API |
| 02 | Generative AI Powered Applications with Python |
| 03 | Using the PaLM API in Applications |



PaLM is a Large Language Model (LLM)

- LLMs are very sophisticated autocomplete applications
 - They learn patterns from large amounts of text
 - Use those patterns to generate text
- When generating text they calculate the next most likely tokens (words)
 - They aren't smart; it's math and statistics
- PaLM can generate text with two basic services
 - Text service for single request interactions
 - Chat service is for interactive, multi-turn interactions

To use the PaLM API, your application will need to be authenticated

- Ways to authenticate your application
 - Obtain an authorization token
 - Run the application using a service account

An authorization token identifies the caller of an API

- Created using the Google Cloud CLI
 - The gcloud CLI must be initialized with either a user or service account
- Set the Authorization header variable with the token generated using gcloud

```
API_ENDPOINT="us-central1-aiplatform.googleapis.com"
PROJECT_ID="vertext-ai-dar"
MODEL_ID="chat-bison"
curl -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/${LOCATION_ID}/publishers/google/models/${MODEL_ID}:predict" \
<<code omitted>>
```

If running an application in Google Cloud, assign a service account to the runtime

- Create a service account using **IAM**
 - Assign the **Vertex AI Service Agent** role
 - Use the service account to identify the runtime
- If using Cloud Run, App Engine, or Cloud Functions, the runtime will use the Compute Engine Default Service Account by default
 - This will work as it uses the Editor role
 - Violates principle of least privilege
- You can also download Service Account keys to authenticate programs that use the language client libraries

The screenshot displays the Google Cloud IAM console interface. It features two main panels. The first panel, titled '1 Service account details', contains fields for 'Service account name' (set to 'vertex-ai-sa'), 'Display name for this service account', 'Service account ID *' (set to 'vertex-ai-sa'), and 'Email address: vertex-ai-sa@vertex-ai-dar.iam.gserviceaccount.com'. The second panel, titled '2 Grant this service account access to project (optional)', provides instructions on granting permissions and includes a 'Role' dropdown menu currently set to 'Vertex AI Service Agent', an 'IAM condition (optional)' section with a '+ ADD IAM CONDITION' link, and a trash icon for removal.

1 Service account details

Service account name: vertex-ai-sa

Display name for this service account

Service account ID *: vertex-ai-sa

Email address: vertex-ai-sa@vertex-ai-dar.iam.gserviceaccount.com

2 Grant this service account access to project (optional)

Grant this service account access to vertex-ai-dar so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role: Vertex AI Service Agent

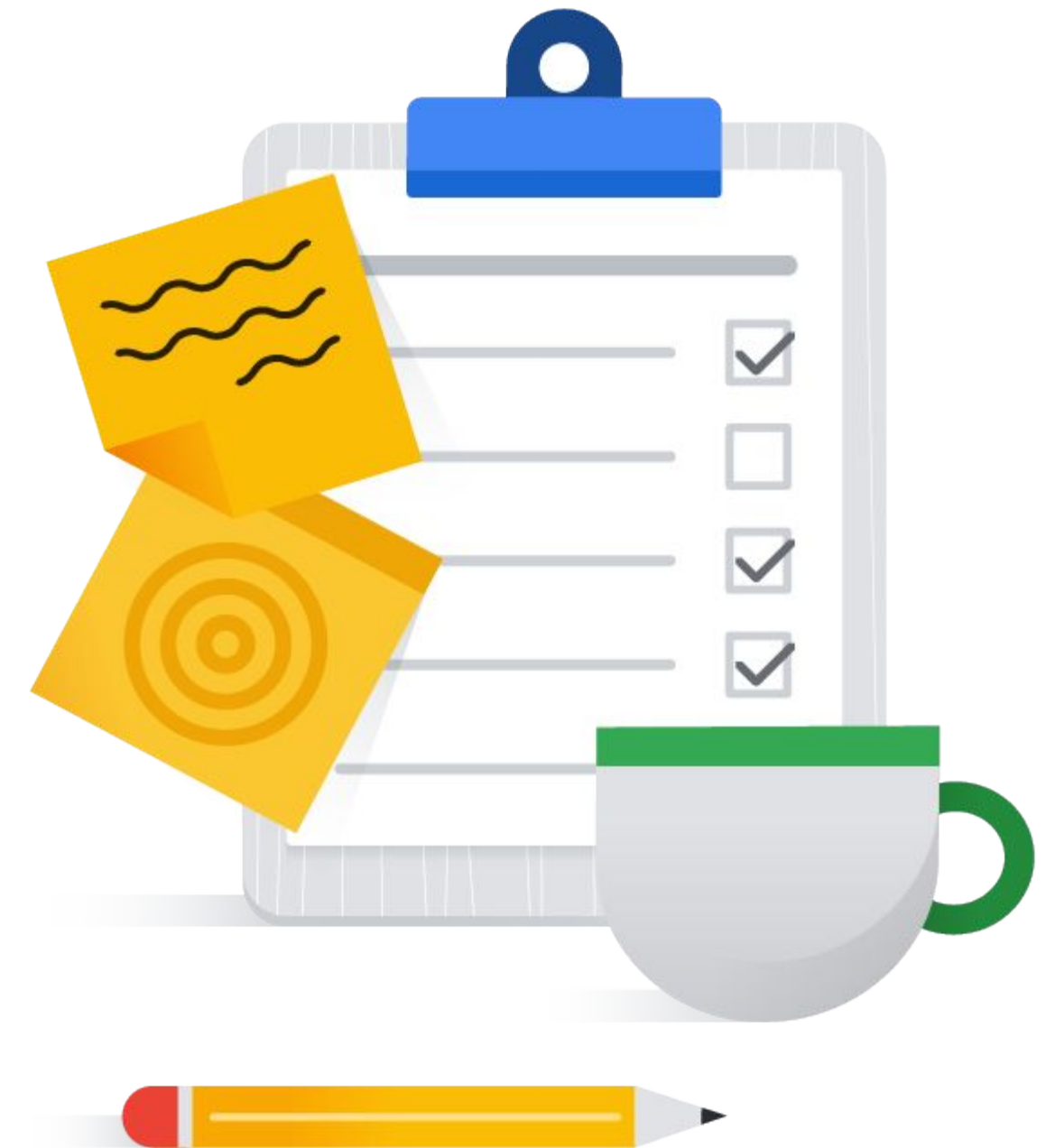
IAM condition (optional) ?

+ ADD IAM CONDITION

Gives Vertex AI the permissions it needs to function.

Topics

- | | |
|----|--|
| 01 | Introduction to the PaLM API |
| 02 | Generative AI Powered Applications with Python |
| 03 | Using the PaLM API in Applications |



You can use Vertex AI Studio to generate code for a Python, Node.js or Java app or cURL

- Click the **Get Code** button and select Python, Node.js, Java or CURL

Use this script to request a model response in your application.

1. Install the Vertex AI SDK: Open a terminal window and enter the command below. You can also [install it in a virtualenv](#).

```
!pip install --upgrade google-cloud-aiplatform
```

2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="roi-genai-joe", location="us-central1")
parameters = {
    "candidate_count": 1,
    "max_output_tokens": 1024,
    "temperature": 0.9,
    "top_p": 1
}
model = TextGenerationModel.from_pretrained("text-bison")
response = model.predict(
    """write a short story about a kitten""",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Use this script to request a model response in your application.

1. Install the Vertex AI SDK.

```
npm install https://github.com/googleapis/nodejs-vertexai
gcloud auth application-default login
```

2. Create an index.js file and add the following code:

```
const {VertexAI} = require('@google-cloud/vertexai');

// Initialize Vertex with your Cloud project and location
const vertex_ai = new VertexAI({project: 'roi-genai-joe', location: 'us-central1'});
const model = 'gemini-pro-vision';

// Instantiate the models
const generativeModel = vertex_ai.preview.getGenerativeModel({
  model: model,
  generation_config: {
    "max_output_tokens": 2048,
    "temperature": 0.4,
    "top_p": 1,
    "top_k": 32
  },
});

async function generateContent() {
  const req = {
    contents: [{role: 'user', parts: [{text: 'write a short story about a kitten'}]}],
  };

  const streamingResp = await generativeModel.generateContentStream(req);

  for await (const item of streamingResp.stream) {
    process.stdout.write('stream chunk: ' + item);
  }

  process.stdout.write('aggregated response: ' + (await streamingResp.response));
};

generateContent();
```

Vertex AI requirements for Python

- Use pip to install Google Cloud AI Platform
 - Or add to your requirements.txt file

```
pip install google-cloud-aiplatform >= 1.25.0
```

- Import Vertex AI

```
import vertexai  
from vertexai.language_models import TextGenerationModel
```

Using Vertex AI from Python

```
vertexai.init(project="vertext-ai-dar", location="us-central1")

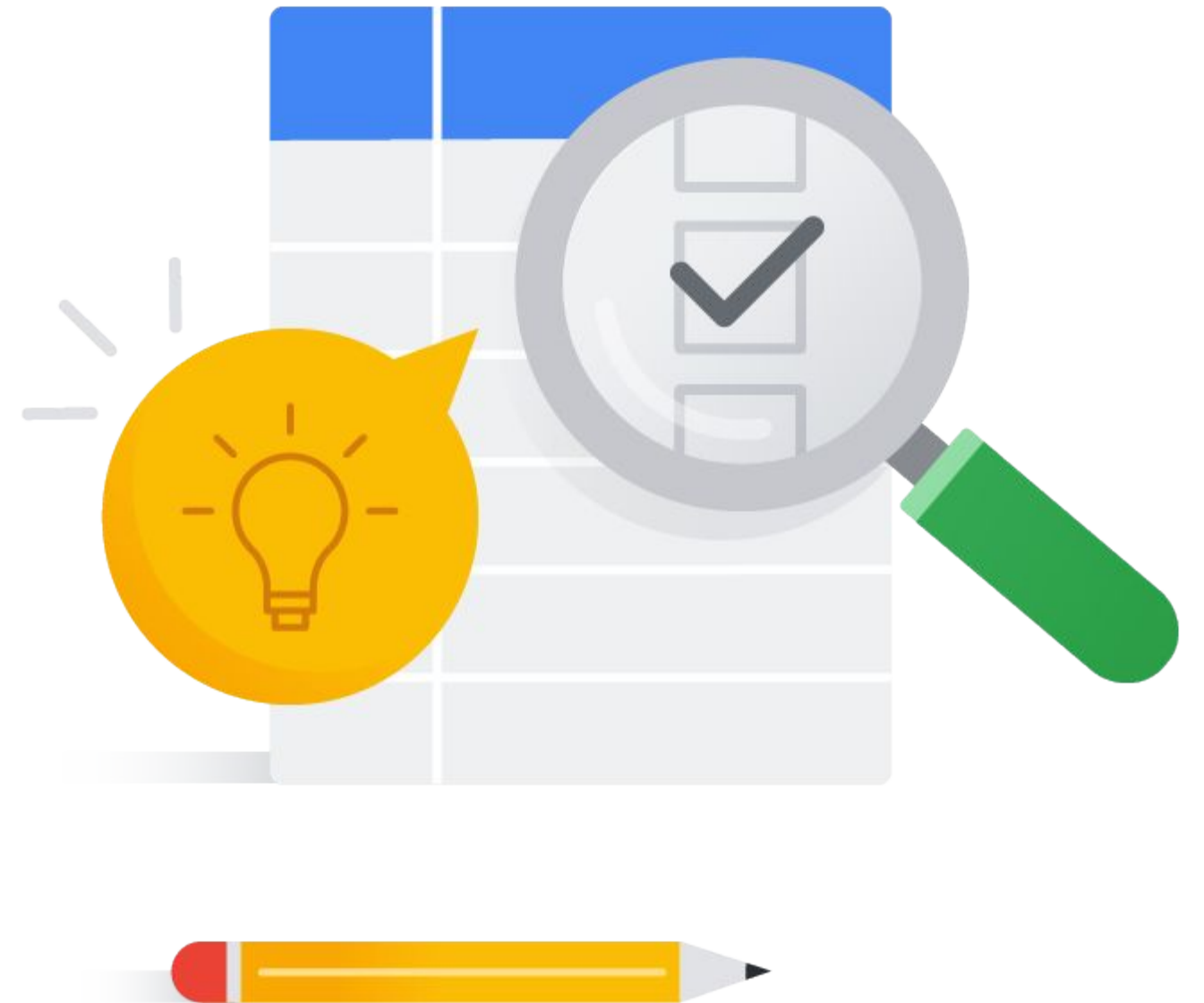
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    """Write for me a limerick about dogs and Python programming.""" ,
    **parameters
)
print(f"Response from Model: {response.text}")
```

Lab

🕒 30 min ⚙️

Getting Started with the PaLM API



Topics

- | | |
|----|--|
| 01 | Introduction to the PaLM API |
| 02 | Generative AI Powered Applications with Python |
| 03 | Using the PaLM API in Applications |



Python Flask Website example

- This is an example of using the text service with the PaLM API
 - Even though you may ask many questions, each one is independent
- Context must be added to tell the PaLM API to emulate a barista
- The coding is simple as you are just submitting an HTML form and making a request to the PaLM API for a response
 - The response is displayed on the screen

The image displays two screenshots of a web application titled "CoffeeBot".

Top Screenshot:

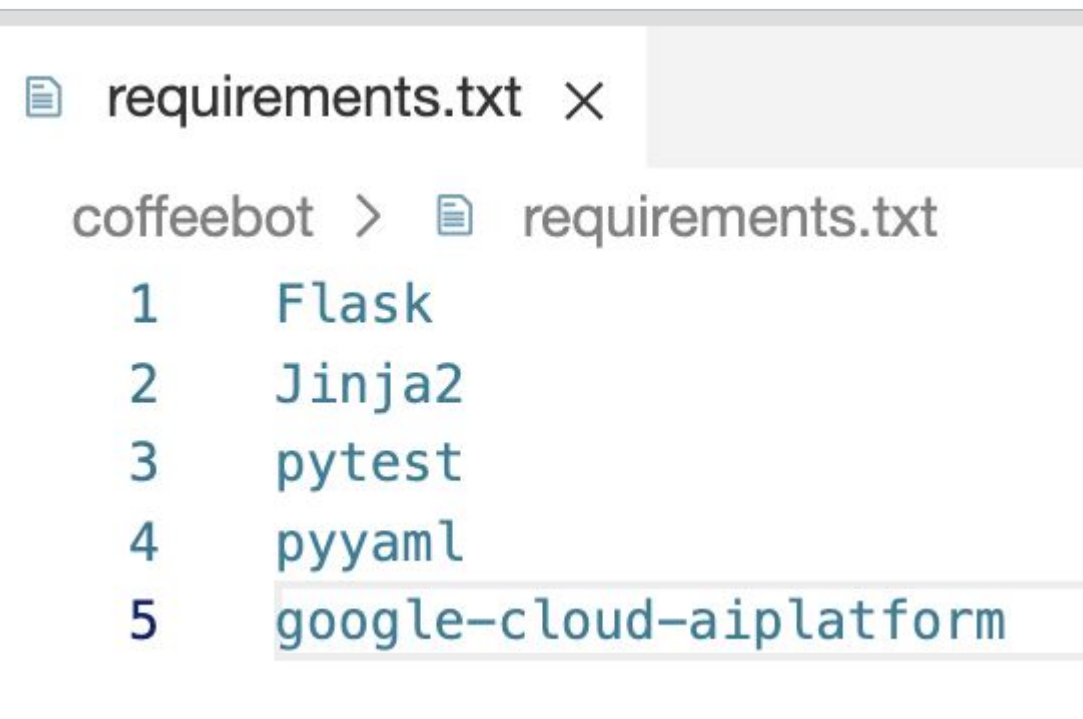
- Header: "CoffeeBot" (blue text)
- Section Header: "CoffeeBot" (large bold black text)
- Subtitle: "Your friendly online BaristaAI" (bold black text)
- Message Box (green background): "I am CoffeeBot, a barista and expert on all things related to coffee and tea. I can help you find the perfect coffee or tea for your taste, and I can also teach you how to make your own coffee and tea drinks at home."
- Form: "Ask CoffeeBot:" label, an empty text input field, and a blue "Submit" button.

Bottom Screenshot:

- Header: "CoffeeBot" (blue text)
- Section Header: "CoffeeBot" (large bold black text)
- Subtitle: "Your friendly online BaristaAI" (bold black text)
- Message Box (green background): "To make a latte, you will need: * 2 shots of espresso * 6 ounces of steamed milk * 1 tablespoon of foamed milk 1. Brew the espresso shots. 2. Steam the milk until it is hot and frothy. 3. Pour the espresso into a latte glass. 4. Add the steamed milk to the espresso. 5. Top with the foamed milk. 6. Enjoy!"
- Form: "Ask CoffeeBot:" label, a text input field containing "How do you make a Latte?", and a blue "Submit" button.

Add the Python requirements

- Add Google Cloud AI Platform to the requirements.txt file
- Add the required imports at the top of the code file



The screenshot shows a code editor window with a tab labeled 'requirements.txt'. The file path is 'coffeobot > requirements.txt'. The file contains five lines of requirements, with the fifth line, 'google-cloud-aiplatform', highlighted.

```
1 Flask
2 Jinja2
3 pytest
4 pyyaml
5 google-cloud-aiplatform
```

```
from flask import Flask, render_template, request
import os
import vertexai
from vertexai.language_models import TextGenerationModel
```


Handling web requests in Flask

- The default route will handle HTTP posts and gets
 - Post means a question was submitted from the HTML form
 - Get means there is no question (have CoffeeBot introduce itself)
- The code for using the PaLM API is in the **get_response()** function


```
@app.route("/", methods = ['POST', 'GET'])
def main():
    if request.method == 'POST':
        input = request.form['input']
        response = get_response(input)
    else:
        input = ""
        response = get_response("Who are you and what can you do?")

    model = {"title": "CoffeeBot", "message": response, "input": input}
    return render_template('index.html', model=model)
```



Making a request to the PaLM API

```
def get_response(input):  
    vertexai.init(project="vertext-ai-dar", location="us-central1")  
    parameters = {  
        "temperature": 0.8,  
        "max_output_tokens": 256,  
        "top_p": 0.8,  
        "top_k": 40  
    }  
  
    model = TextGenerationModel.from_pretrained("text-bison@001")
```

Initialize the API
and set up the
parameters



Create the model
using the correct
version of the
PaLM API



<< CODE ON NEXT SLIDE OMITTED >>

Making a request to the PaLM API (continued)

```
def get_response(input):  
    << CODE FROM PREVIOUS SLIDE OMITTED >>  
  
    model = TextGenerationModel.from_pretrained("text-bison@001")  
    request = """Your name is CoffeeBot. You are a barista and expert on  
all things related to coffee and tea..  
  
input: {}  
output:  
"""  
    response = model.predict(  
        request.format(input),  
        **parameters  
    )  
    return response
```

The context tells the API to emulate a bartender

The input is the question

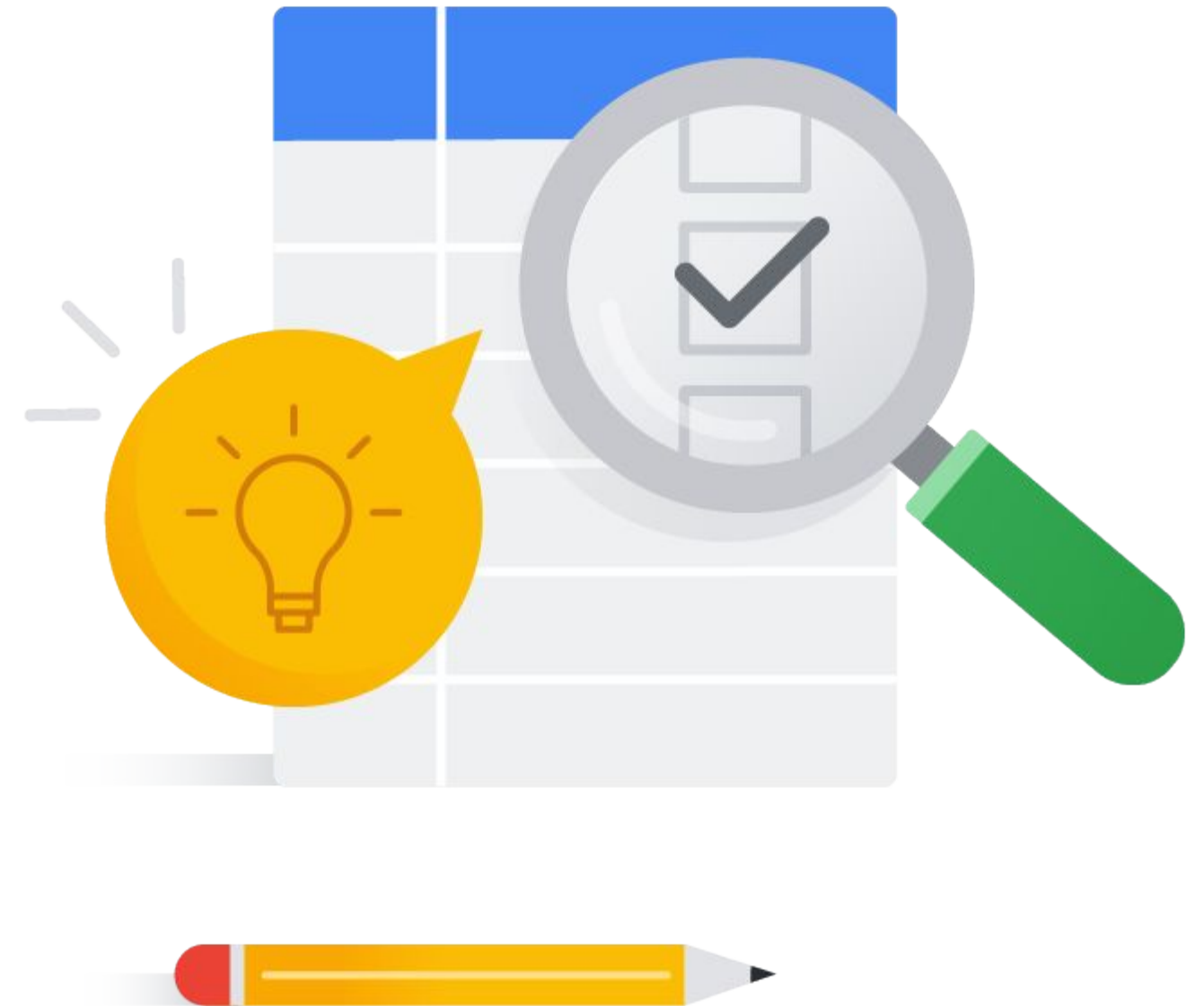
Call the model.predict() function to send the request

The format function injects the question into the prompt

Lab

🕒 30 min ⚙️

Integrating the PaLM API into Applications



In this module, you learned to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

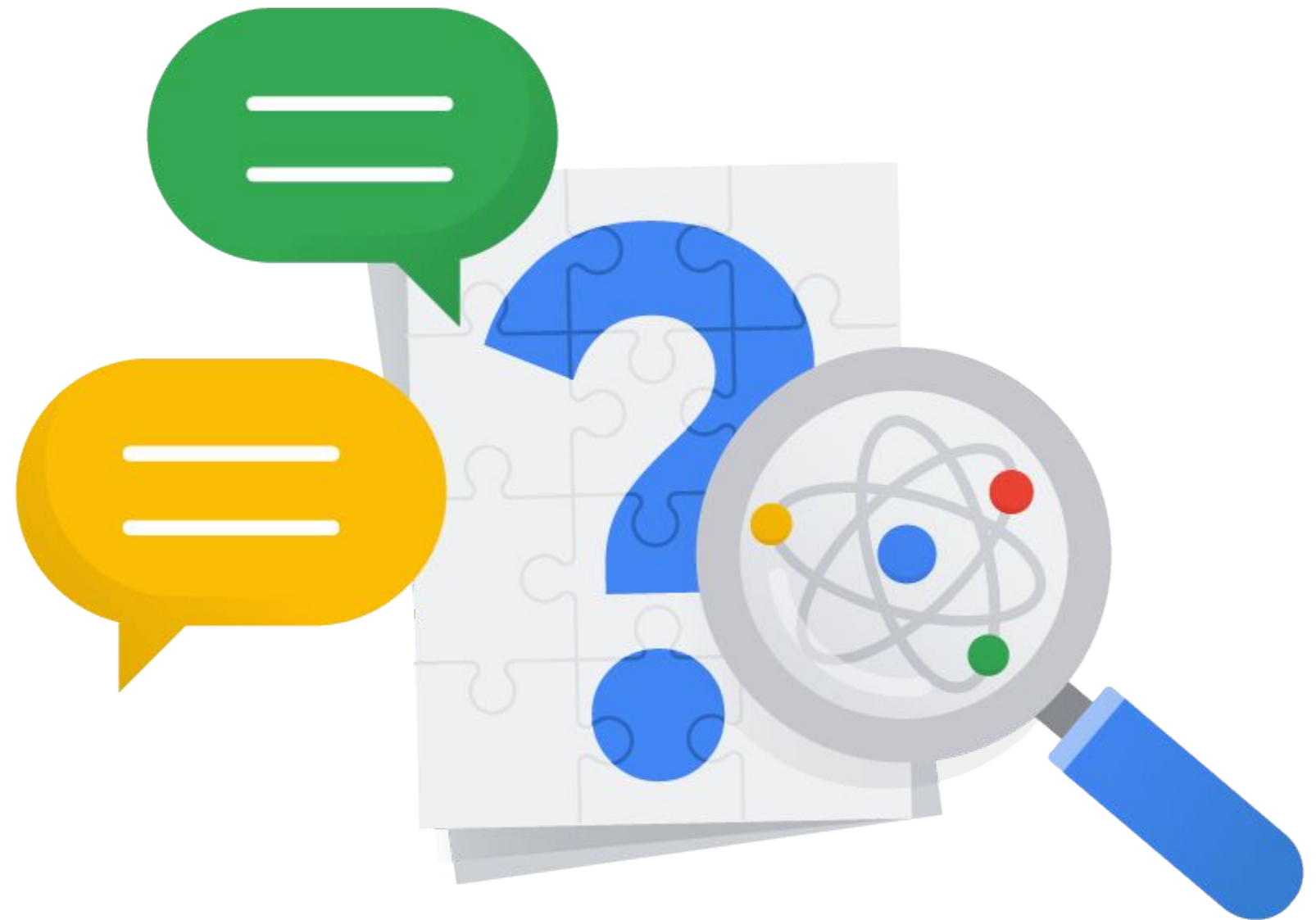
Program Python applications the use PaLM to generate content

03

Integrate PaLM and GenAI into your applications



Questions and answers



Quiz question

How do you authenticate a request to the PaLM API ?

A: Using a API Key

B: Using an authorization token

C: With a Service Account

D: All of the above would work

Quiz question

How do you authenticate a request to the PaLM API ?

A: Using a API Key

B: Using an authorization token

C: With a Service Account

D: All of the above would work

Quiz question

What programming languages are
Supported by the PaLM API ?

A: Python

B: Node.js

C: Swift

D: Java

E: All of the above

Quiz question

What programming languages are
Supported by the PaLM API ?

A: Python

B: Node.js

C: Swift

D: Java

E: All of the above