LAB

Develop prompt templates using variables to generate output from LLMs

IBM **SkillsBuild**

Introduction 3
Software requirements 3
Objective 3
Lab steps 3
Scenario 4
Background information 4
Challenge 4
Solution 4
Create a GitHub account 5
Overview 5
Instructions 5
Create a Replicate account 9
Overview 9
Instructions 9
Sign up for Google Colab 13
Overview 13
Instructions 13
Load the Jupyter notebook and initialize the model 17
Overview 17
Instructions 17
Draft a baseline prompt for a recurring task 23
Overview 23
Instructions 23
Identify variable elements and convert to a template 25
Overview 25
Instructions 25
Test and finalize the prompt template 27
Overview 27
Instructions 27
Conclusion 31

Introduction

In this lab, you will identify variables and develop prompt templates to generate relevant output from large language models (LLMs).

Software requirements

To complete this lab, you don’t have to install any software on your computer. You only require a Google account to access Google Colab and a Replicate account to access and use various artificial intelligence (AI) models.

Objective

After completing this lab, you should be able to:

- Develop prompt templates using variables to generate output from LLMs

Lab steps

This lab requires you to complete the following procedures:

1. Create a GitHub account
2. Create a Replicate account
3. Sign up for Google Colab
4. Load the Jupyter notebook and initialize the model
5. Draft a baseline prompt for a recurring task
6. Identify variable elements and convert to a template
7. Test and finalize the prompt template

Estimated duration to complete

20 minutes

Scenario
Background information

Codal Technologies is expanding and offering consultation services to some of its major clients. With more stakeholders involved, analysts need to produce accurate weekly project updates for them.

Challenge

However, the variety of formats and prompting styles has led to inconsistencies in quality, tone, and completeness of these outputs.

Solution

To address this, your manager asks you, a senior analyst, to create the firm's first prompt template. This template will streamline the workflow by allowing analysts to update only a few project-specific variables while maintaining a uniform structure. The goal is to deliver high-quality and consistent communication, no matter who prepares the update or which client it is for.

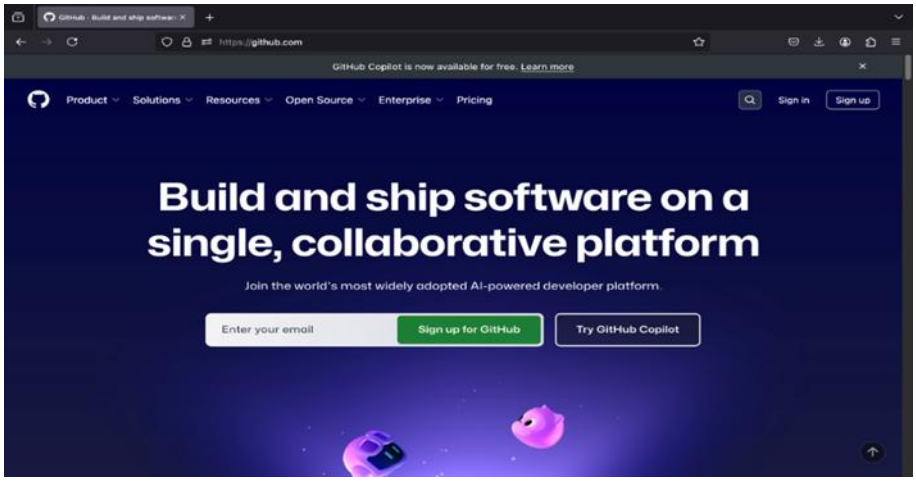
Create a GitHub account

Overview

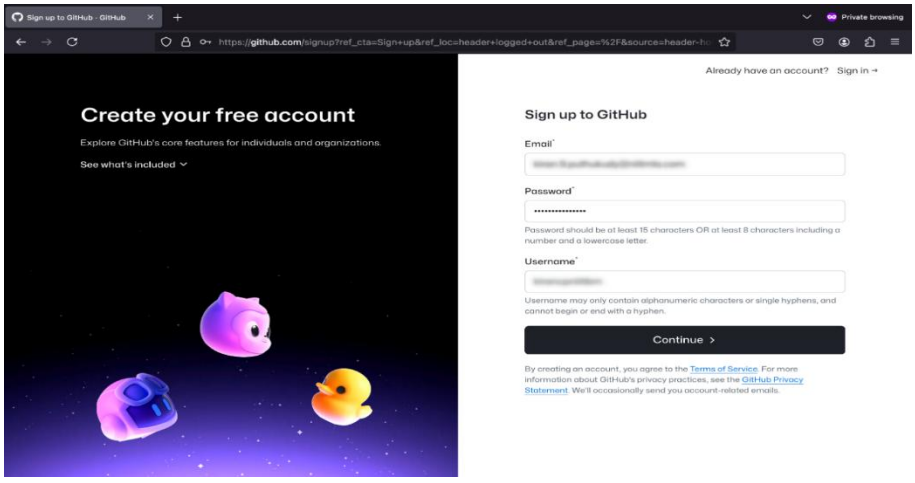
In this procedure, you'll set up a GitHub account. GitHub is a platform that helps developers store, manage, and share code, while also supporting collaboration through tools such as version control, bug tracking, and task management. Setting up a GitHub account provides access to the Replicate platform, which is required to complete the lab efficiently.

Instructions

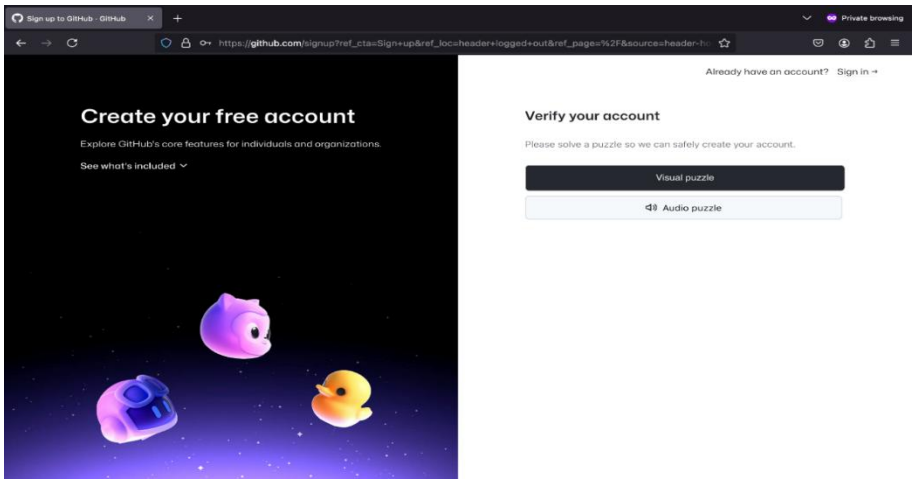
1. To create a GitHub account, go to the GitHub website.
2. Select the **Sign up for GitHub** button, as shown in the following screenshot.



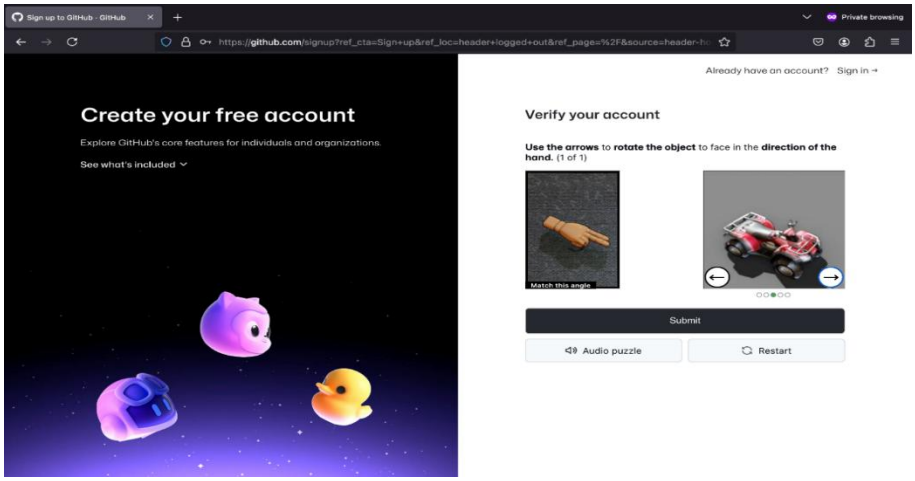
3. The following screenshot shows the "Sign up to GitHub" page. Enter your details in the **Email**, **Password**, and **Username** fields. Then, select **Continue**.



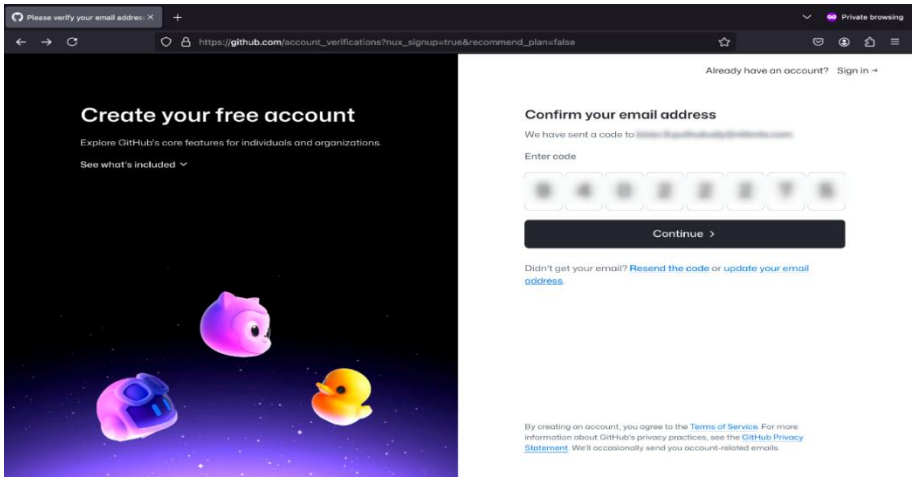
4. To verify your account, select the **Visual puzzle** button.



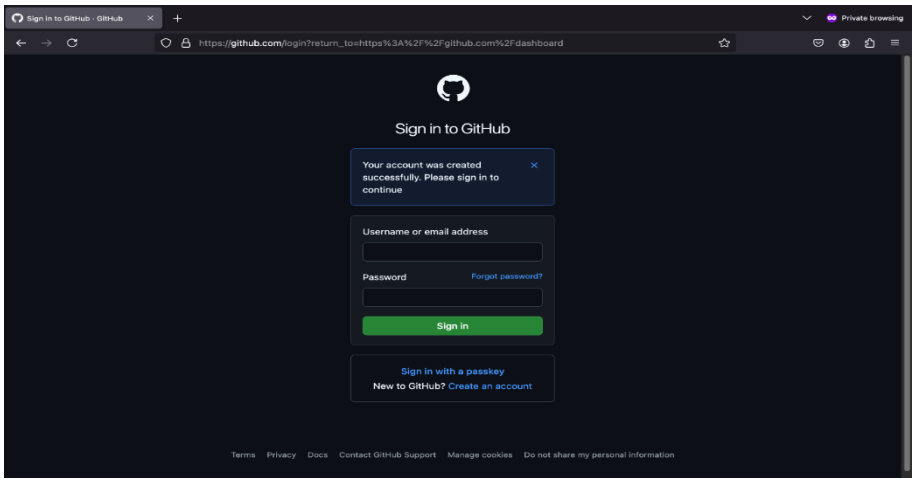
5. Solve the visual puzzle and select **Submit**.



6. To confirm your email, enter the confirmation code sent to your registered email in the **Enter code** field and select **Continue**.

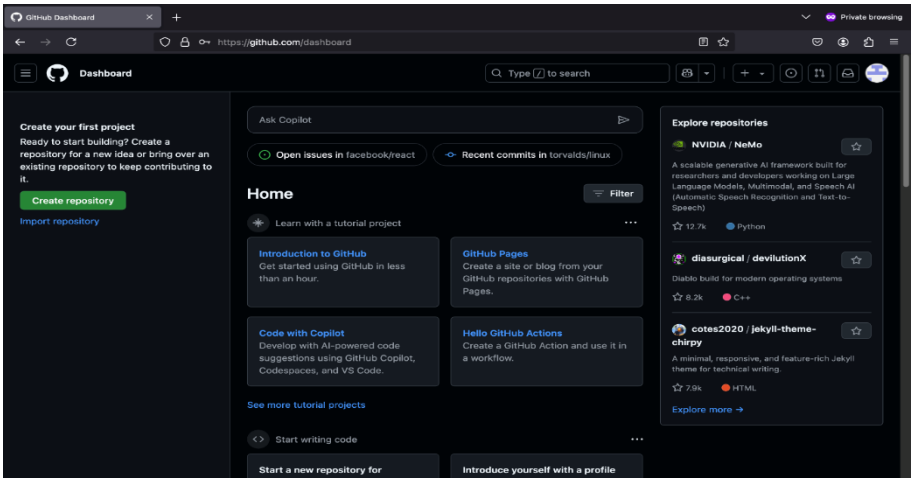


7. After your GitHub account has been successfully created, a confirmation message is displayed as shown in the following screenshot.



8. To sign in to your account, enter your credentials in the **Username or email address** and **Password** fields, and then select **Sign in**.

9. After you sign in, the GitHub dashboard displays as shown in the following screenshot.



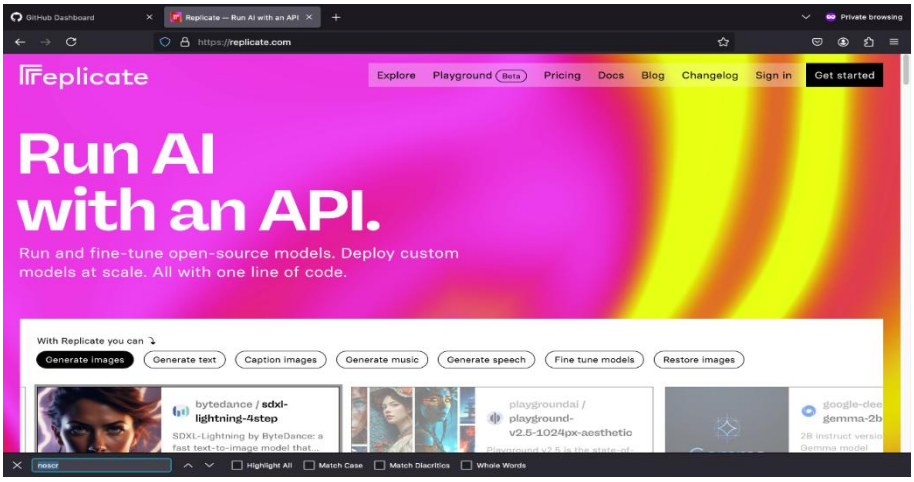
Create a Replicate account

Overview

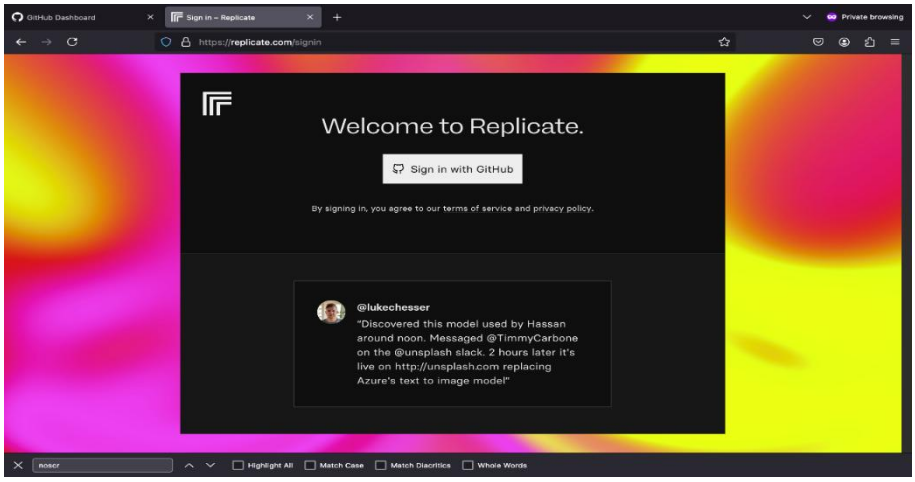
In this procedure, you'll use your GitHub account to register for a Replicate account. Replicate is a cloud-based platform that lets you use AI models such as IBM Granite without requiring advanced hardware. As part of this procedure, you'll create a Replicate token, a secure access key that allows the lab environment to authenticate with Replicate and run models from your account in Google Colab.

Instructions

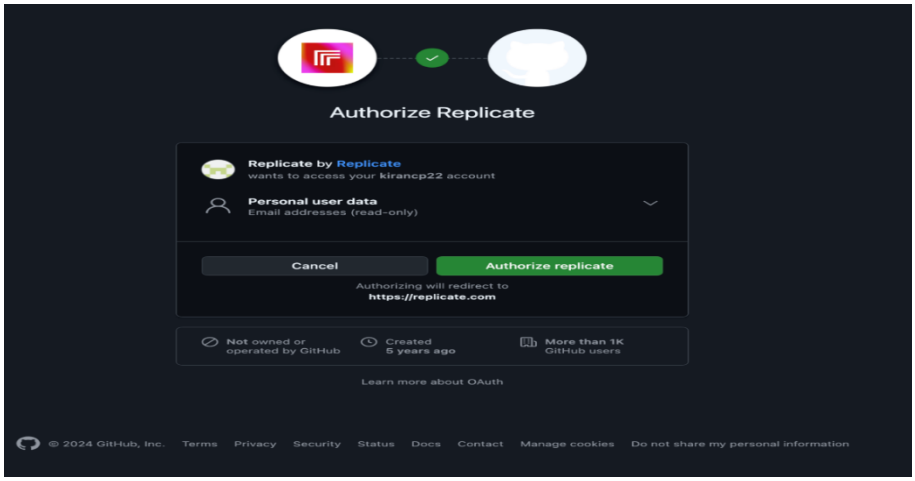
1. Go to the Replicate website.
2. Select **Get started**, as shown in the following screenshot.



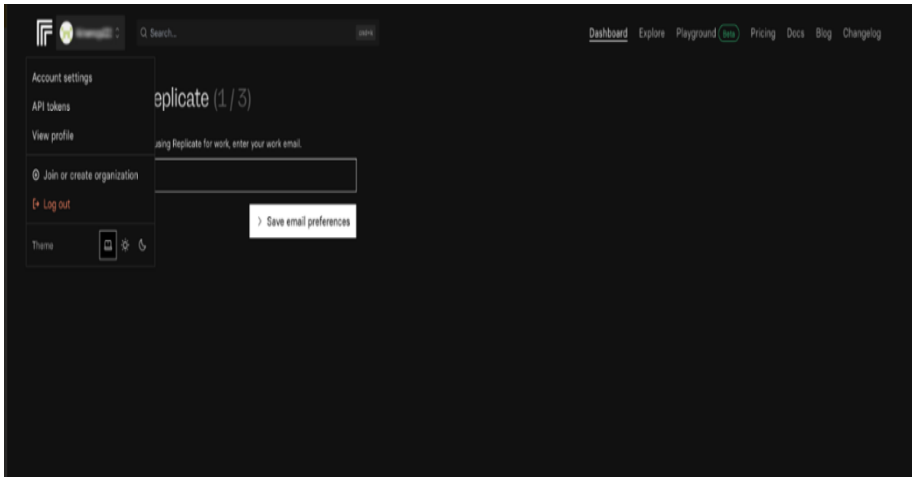
3. Select the **Sign in with GitHub** button, as shown in the following screenshot.



4. Select **Authorize replicate** to continue, as shown in the following screenshot.

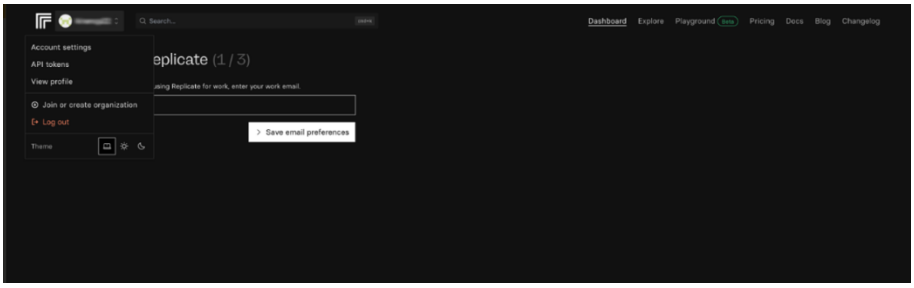


5. After your Replicate account is created, the Replicate dashboard displays as shown in the following screenshot. To create a Replicate token, select the **Account settings** option from the navigation menu.

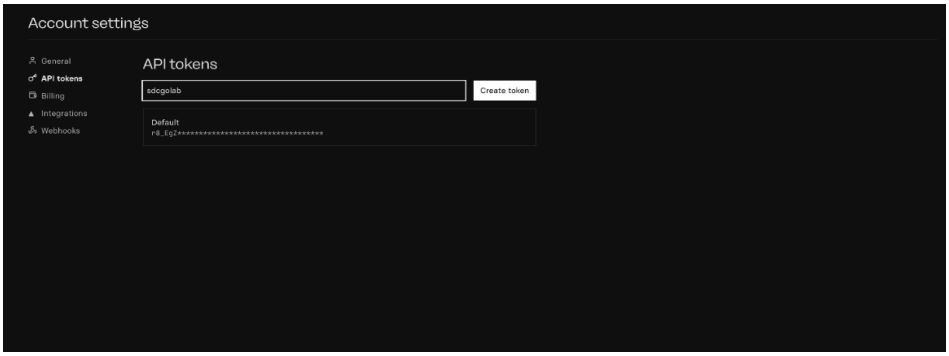


6. The following screenshot shows the Replicate dashboard. Select **API tokens** from the “Account settings” panel.

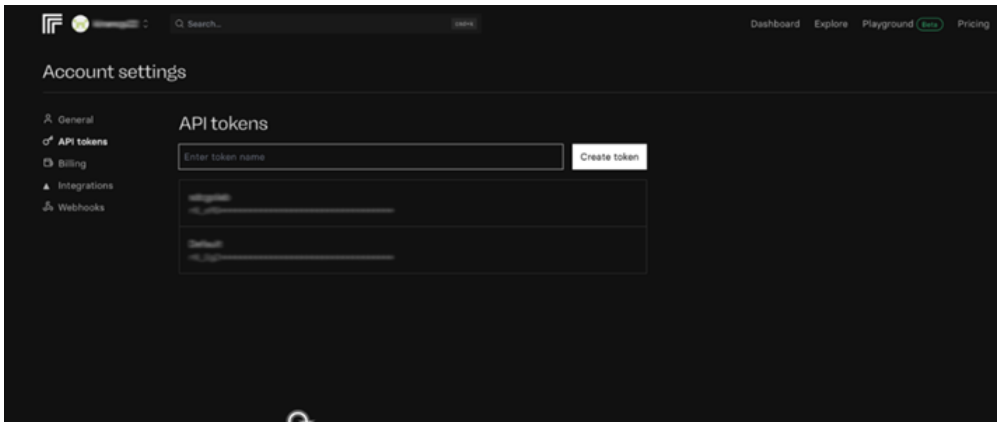
.



7. Enter a name for the token in the **API tokens** field and then select **Create token**, as shown in the following screenshot.

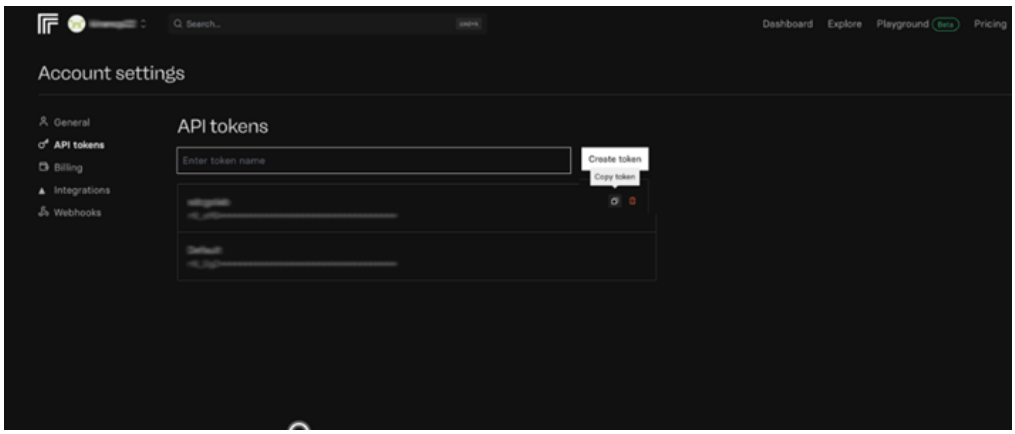


8. After your API token is created, it is displayed on the “Account settings” page as shown in the following screenshot.



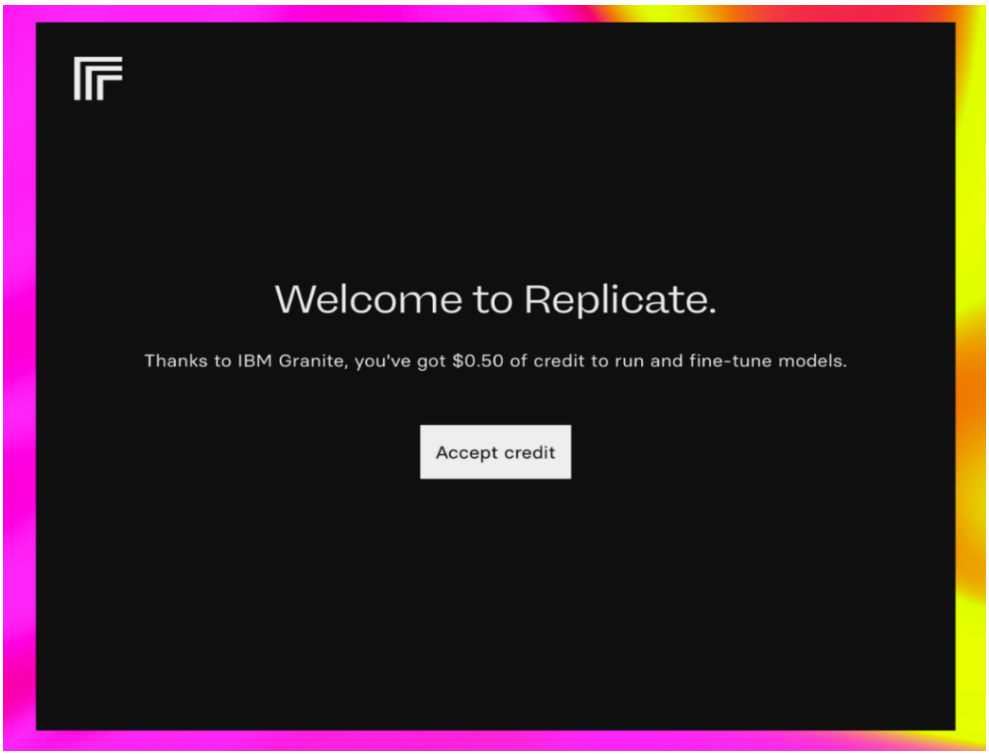
9. To copy the Replicate API, select the **Copy token** icon, as shown in the following screenshot.

   Note: Save the Replicate token because you’ll need it to authenticate with the Replicate API when running code in the Google Colab environment later in this lab.

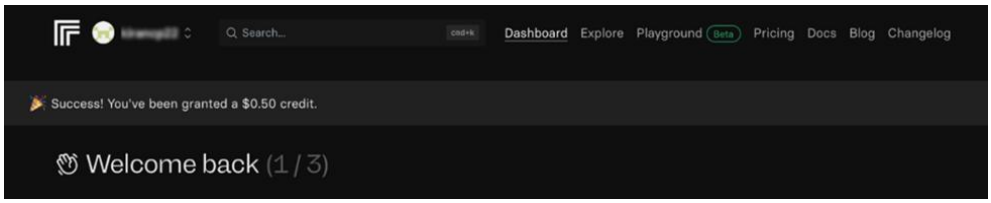


10. To run the lab without interruptions, you'll require some Replicate credits. Go to the Replicate invite link to claim your free $0.50 credit.

11. To claim the amount, select **Accept credit**.



12. A confirmation message is displayed indicating that a $0.50 credit has been added to your Replicate account as shown in the following screenshot.



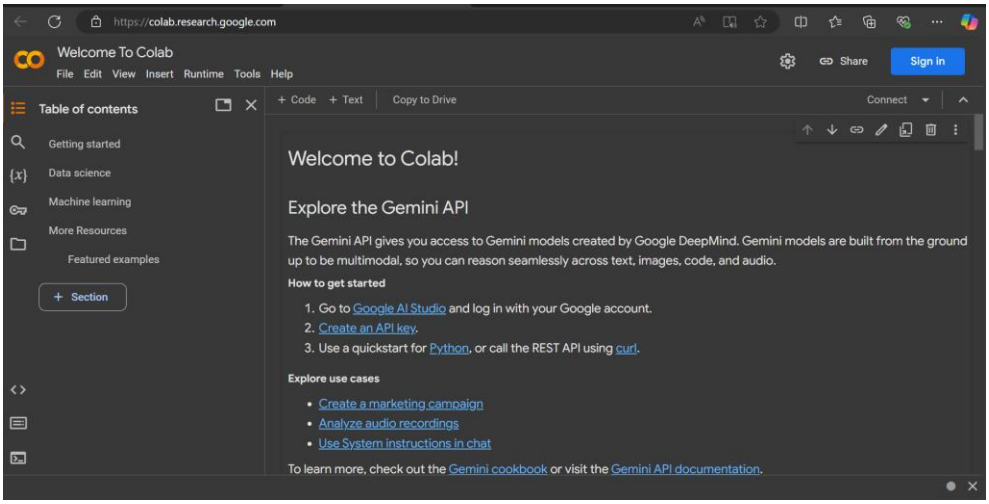
Sign up for Google Colab

Overview

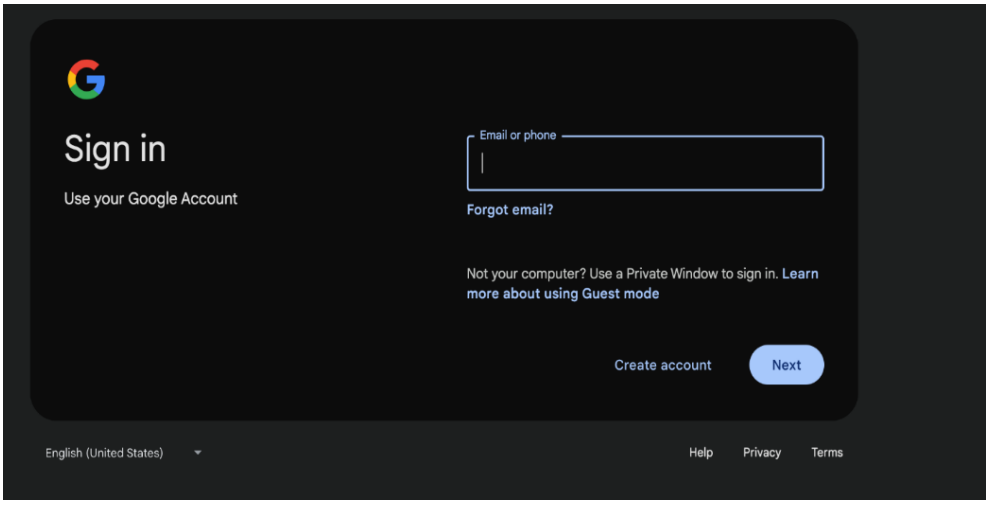
In this procedure, you'll set up a Google Colab account. Google Colab is a free cloud platform that lets you run code in notebooks, which are commonly used for machine learning, data science, and AI tasks. A Google Colab account allows you to install and use the tools needed to complete this lab.

Instructions

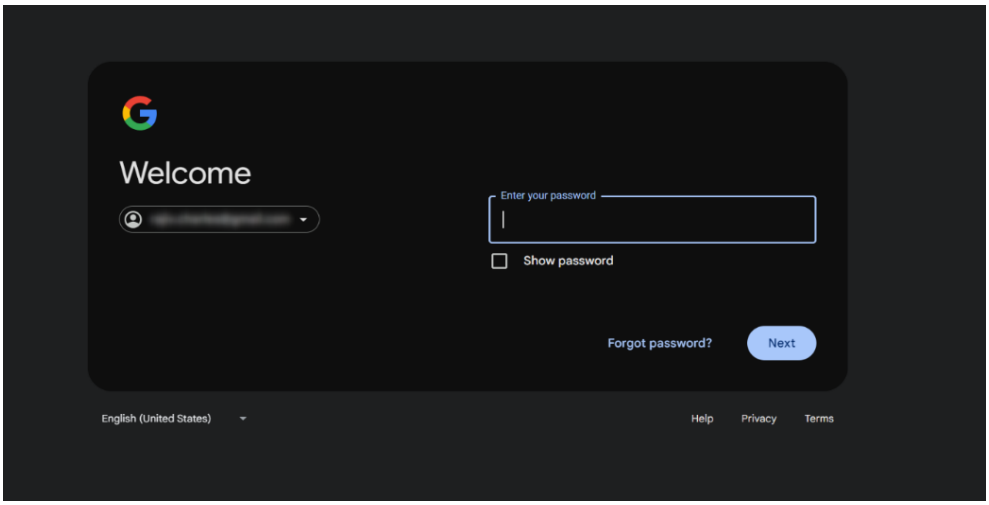
1. To sign up, go to the Google Colab website.
2. Select **Sign in**, as shown in the following screenshot.



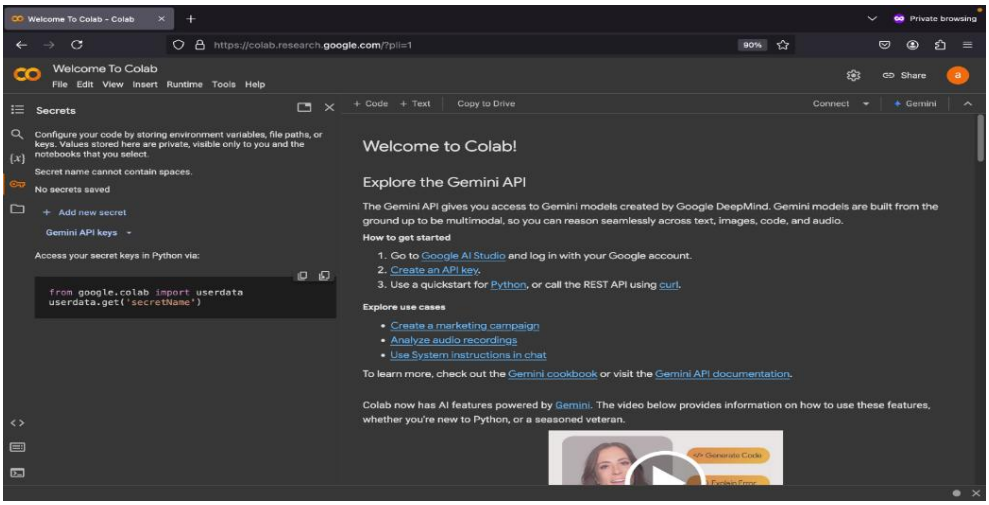
3. Enter your email or phone number in the **Email or phone** field, and then select **Next**. The following screenshot shows the Google sign-in page.



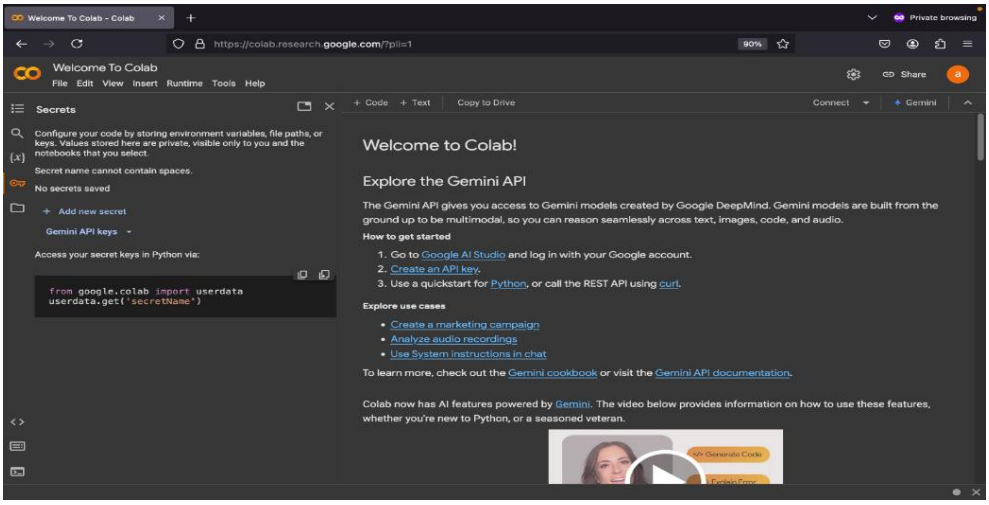
4. Enter your password in the **Enter your password** field, and then select **Next**.



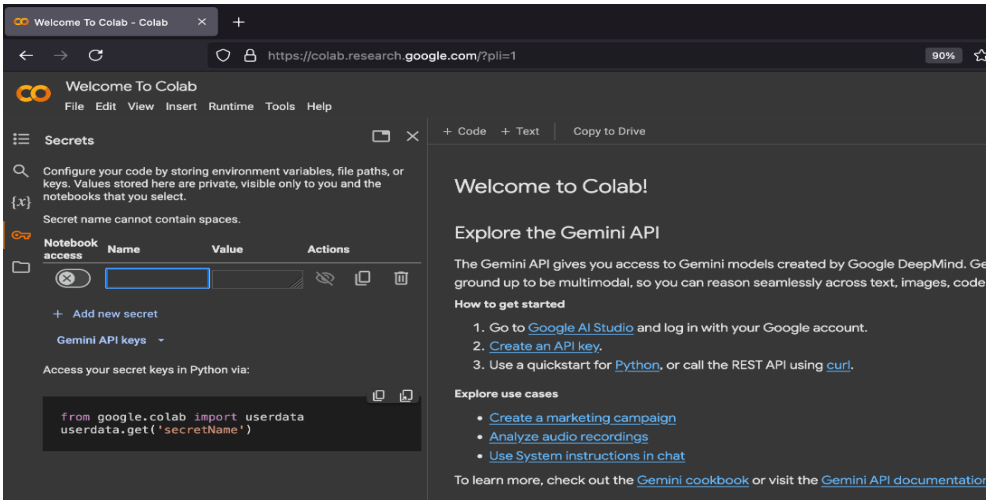
5. To store your Replicate API token, select the key icon from the Secrets tab on the Welcome to Colab page, as shown in the following screenshot.



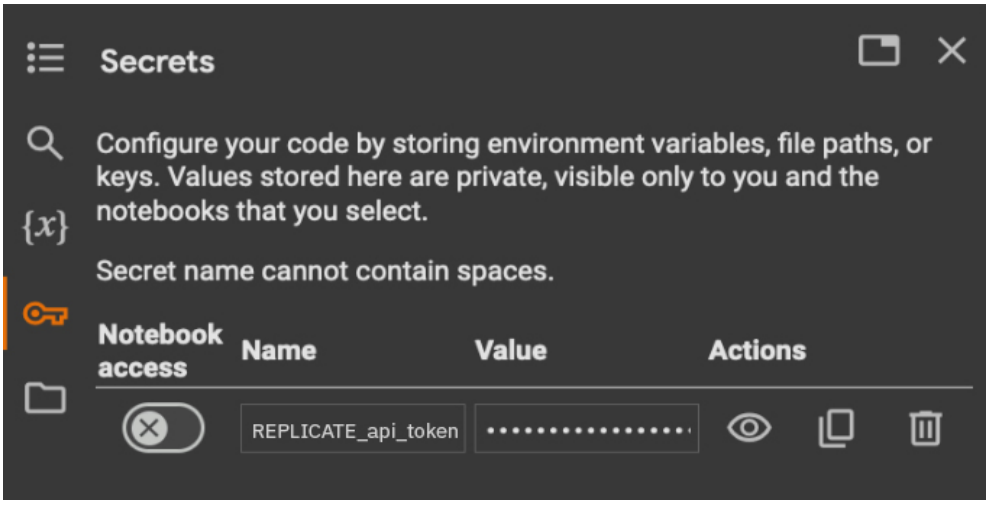
6. Select **Add new secret**.



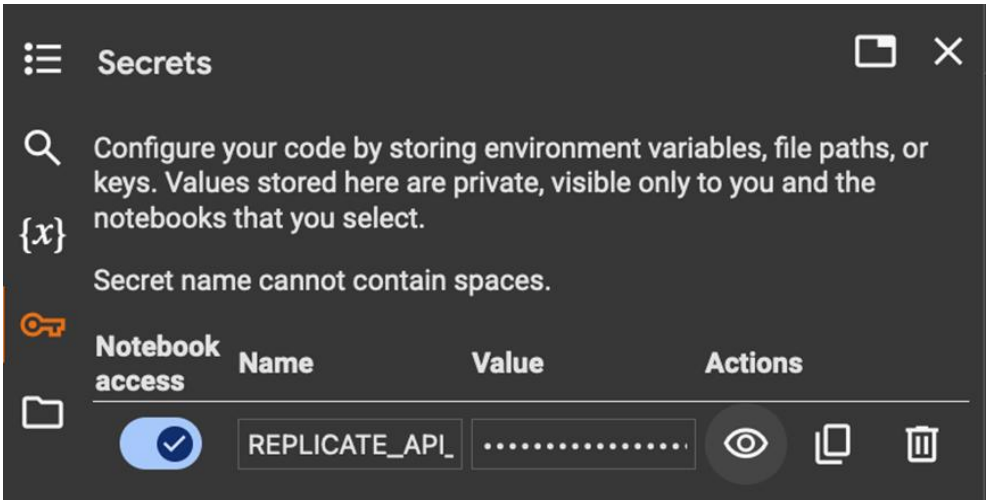
7. Type `REPLICATE_api_token` in the **Name** field.



8. Paste your Replicate API token you copied into the **Value** field, as shown in the following screenshot.



9. Set the **Notebook access** switch on, as shown in the following screenshot. Then, select the close icon to exit the configuration.



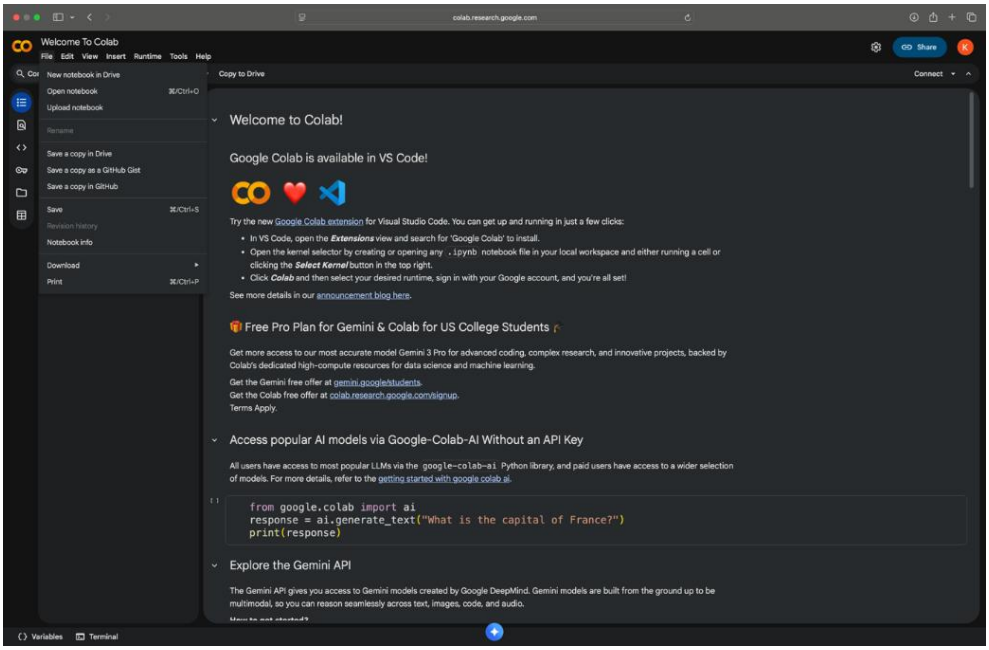
Load the Jupyter notebook and initialize the model

Overview

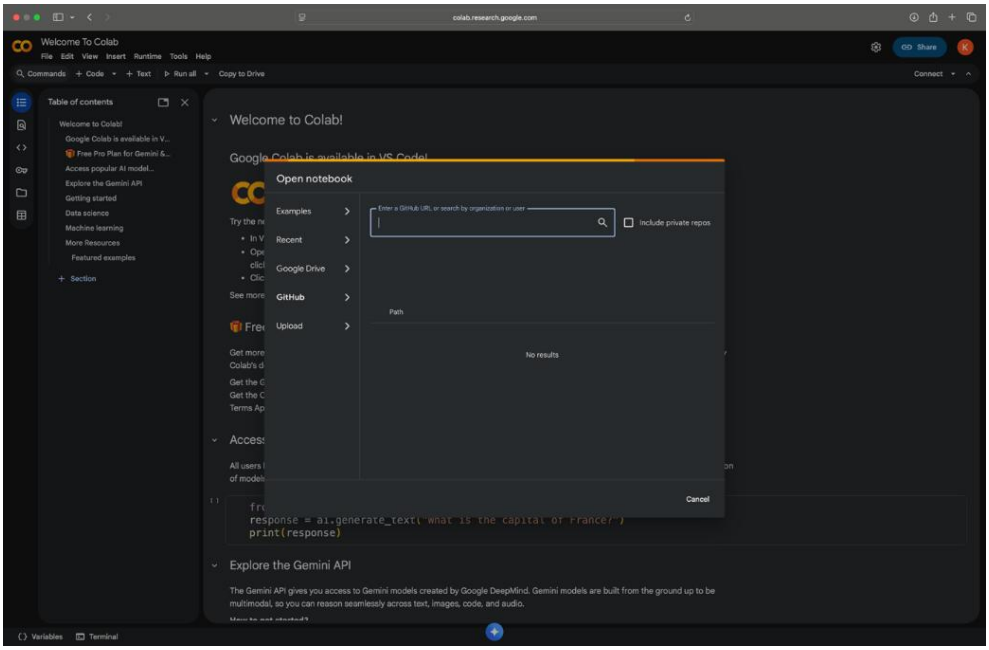
In this procedure, you'll load the Jupyter notebook and initialize the AI model. This sets up your environment before you can interact with the AI model.

Instructions

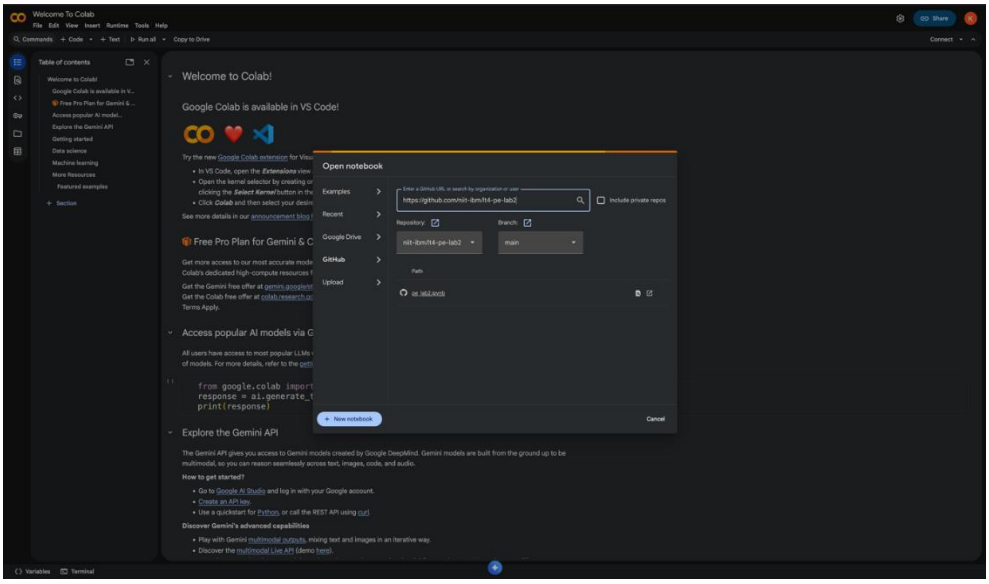
1. In your Google Colab workspace, select **File > Open notebook**.



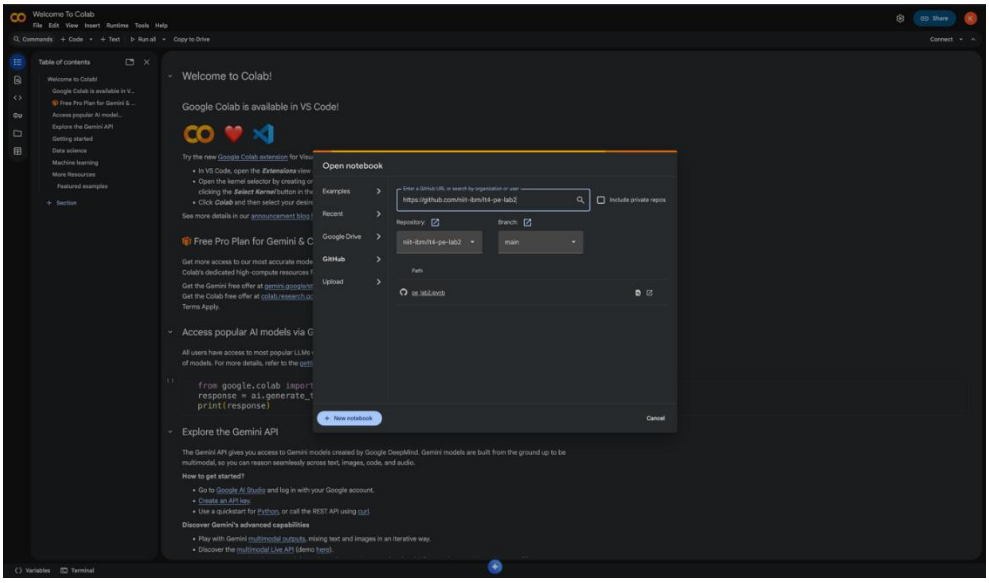
2. From the navigation menu, select **GitHub**.



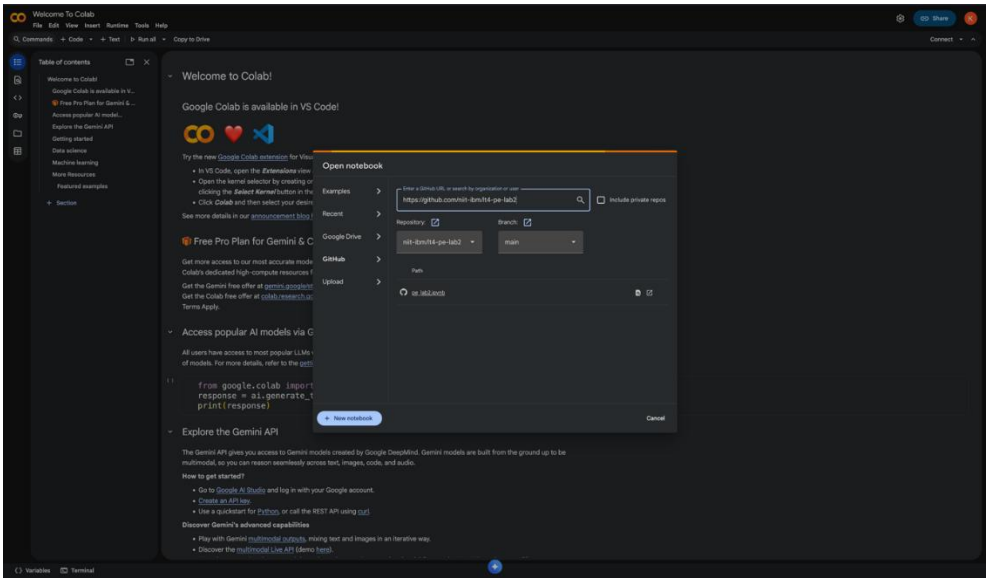
3. In the **Enter a GitHub URL or search by organization or user** field, copy and paste the following URL: https://github.com/niit-ibm/lt4-pe-lab2 and then, select the **search** icon.



4. In the Branch section, select **main**.



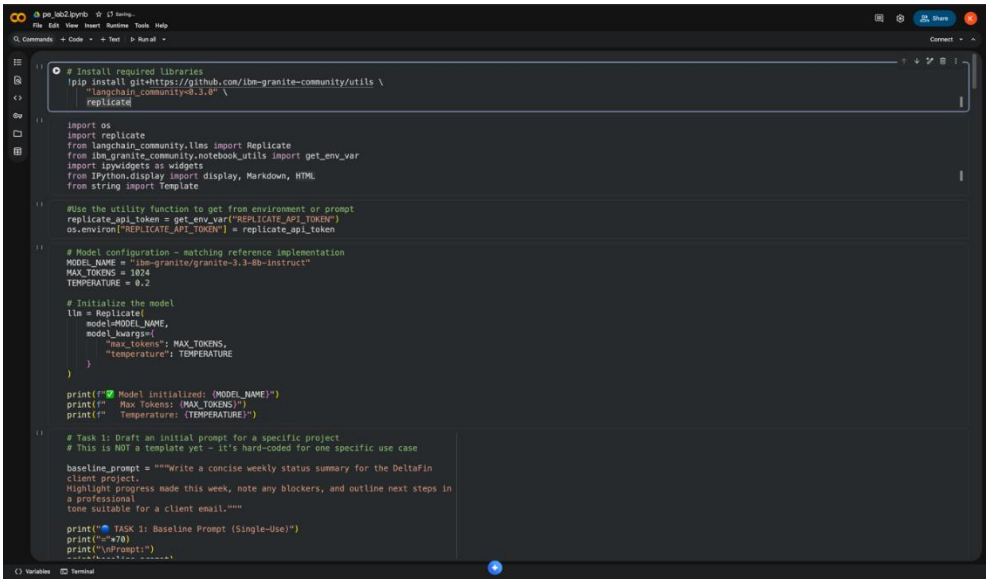
5. Select the **pe_lab2.ipynb** notebook in the Path section to open the notebook in Google Colab.



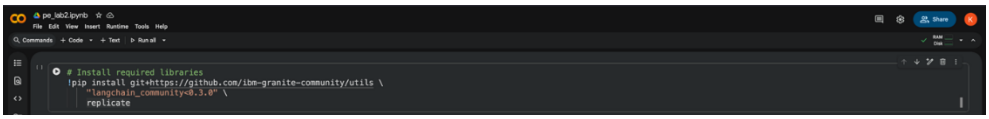
Result: A notebook titled "**pe-1ab2**" that you need to use for this lab activity opens in the Colab Workspace. Note that each row in the notebook is referred to as a **cell**.

6. To run the code in the Jupyter notebook, you'll need to start a new instance in Google Colab runtime. For this, select the "**Connect**" menu on the Google Colab navigation bar, and then, select the **Connect to a hosted runtime** option.

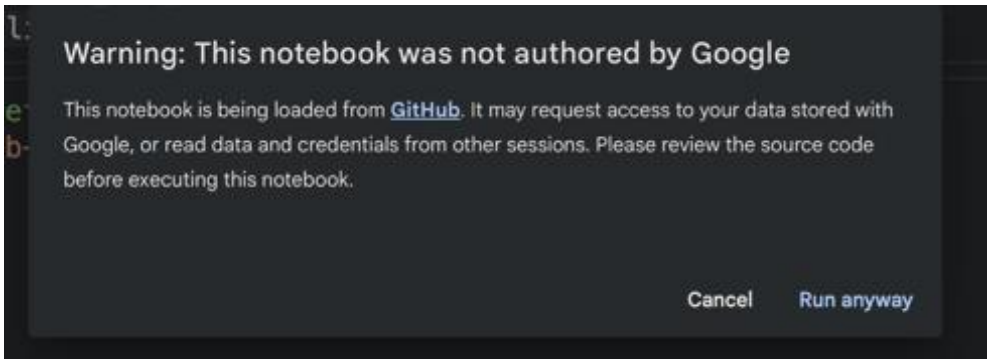
Result: A green checkmark indicates that you have successfully connected to the hosted runtime.



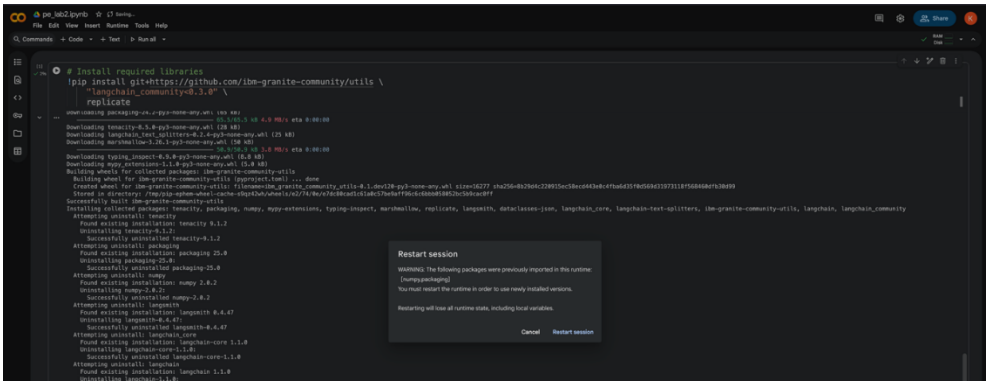
7. In the first cell of the notebook, select the **Play** button to install the required libraries from the Granite suite.



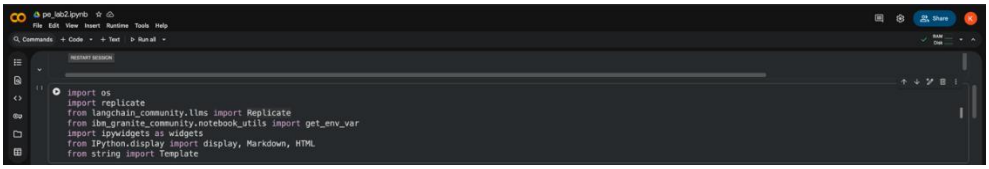
8. Select **Run anyway** to proceed loading the required libraries.



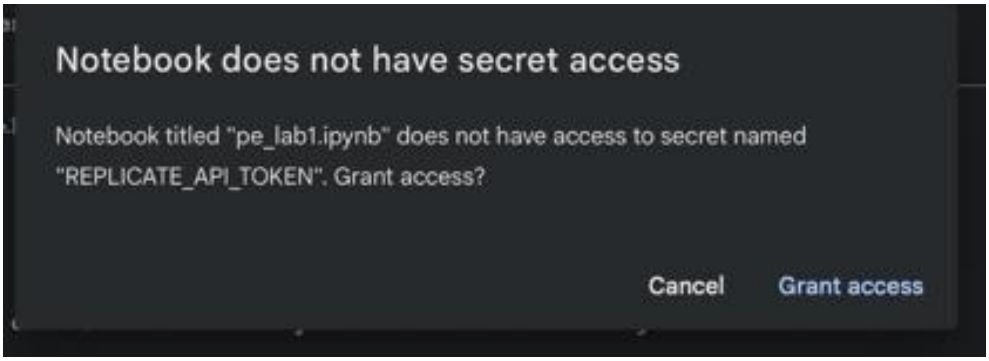
9. Once installation is complete you are prompted to restart the runtime session. Select **Restart session**.



10. Select **Run** to import the libraries.



11. From the next cell, select **Grant access** to let the model access the Replicate Token from the secrets.



12. The following message displays at the bottom of the cell: REPLICATE_API_TOKEN loaded from Google Colab secret.

13. Select the **Play** button to initialize the granite-3.3-8B-instruct model.

Draft a baseline prompt for a recurring task

Overview

In this procedure, you'll draft an initial, working prompt for a specific use case.

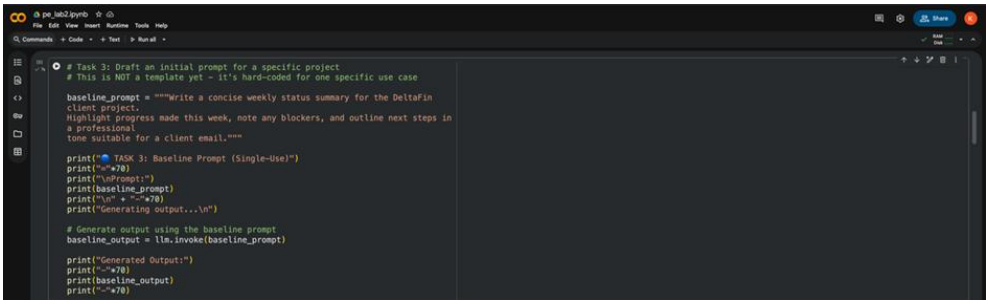
Instructions

1. Begin by writing a baseline prompt for generating weekly status summaries of the projects.

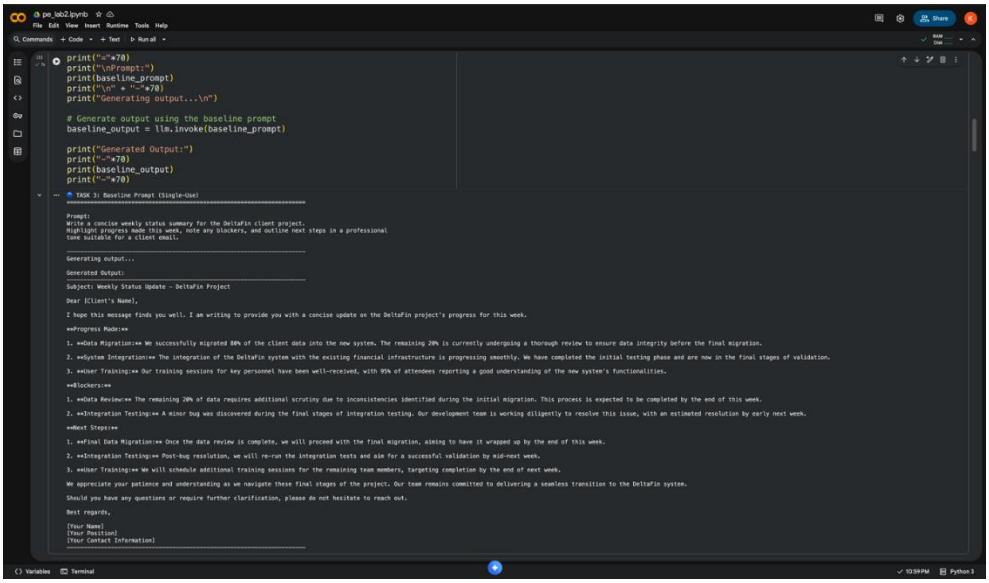
   To do this, copy the following code, paste it into a new code cell, and then select the **Play** icon to execute it.

```
# Task 3: Draft an initial prompt for a specific project
# This is NOT a template yet - it's hard-coded for one specific use
case
baseline_prompt = """Write a concise weekly status summary for the
DeltaFin
client project.
Highlight progress made this week, note any blockers, and outline next
steps in a
professional
tone suitable for a client email."""

print("TASK 3: Baseline Prompt (Single-Use)")
print("-" * 70)
print("\nPrompt:")
print(baseline_prompt)
print("\n" + "-" * 70)
print("Generating output...")

# Generate output using the baseline prompt
baseline_output = lm.invoke(baseline_prompt)

print("Generated Output:")
print("-" * 70)
print(baseline_output)
print("-" * 70)
```

2. The following screenshot shows the code pasted into the code cell.



Result: After you enter the baseline prompt, the LLM generates an output as shown in the following screenshot.



Note that the output addresses all required elements and produces a complete output. However, it gives a vague time reference (“this week”) and it cannot be reused for other projects without a complete rewrite. This prompt works for one specific situation, but it's not scalable.

Identify variable elements and convert to a template

Overview

In this procedure, you'll identify variables such as client name, time period, progress items, blockers, next steps, tone, and audience and convert your baseline prompt to a template.

Instructions

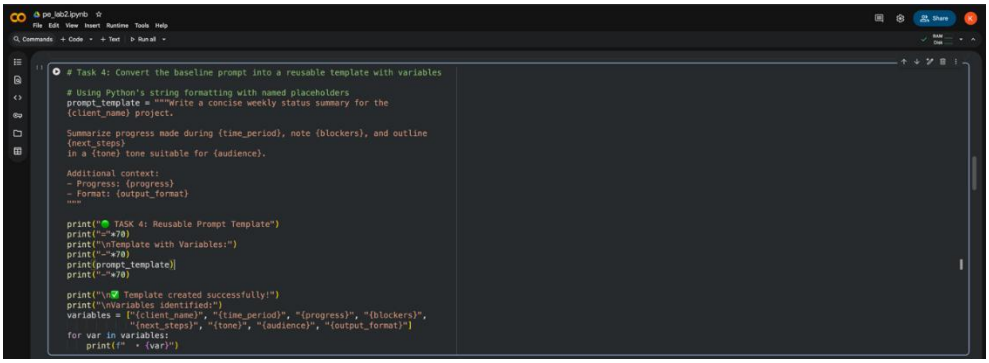
1. Enter the details of the variable of your prompt template in the LLM.

   To do this, copy the following code, paste it into a new code cell, and then select the **Play** icon to execute it.

```
# Task 4: Convert the baseline prompt into a reusable template
with variables
# Using Python's string formatting with named placeholders
prompt_template = """Write a concise weekly status summary for
the
{client_name} project.

Summarize progress made during {time_period}, note {blockers},
and outline
{next_steps}
in a {tone} tone suitable for {audience}.

Additional context:
- progress: {progress}
- format: {output_format}"""


print("TASK 4: Reusable Prompt Template")
print("-" * 70)
print("\nTemplate with Variables:")
print("-" * 70)
print(prompt_template)
print("-" * 70)

print("\n Template created successfully!")
print("\nVariables identified:")
variables = ["{client_name}", "{time_period}", "{progress}",
"{blockers}",
             "{next_steps}", "{tone}", "{audience}",
"{output_format}"]
```

```
for var in variables:
    print(f" - {var}")
```

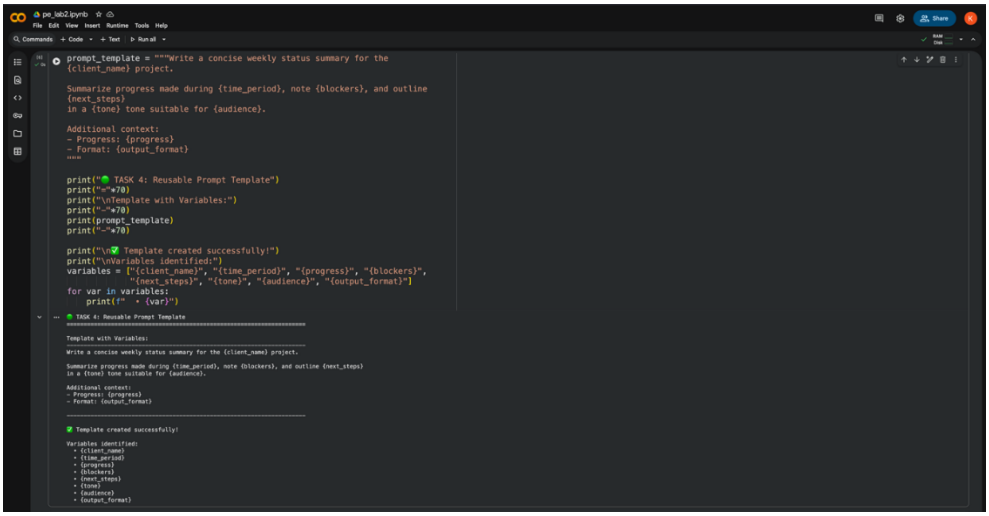
Result: The following screenshot shows the code pasted into the code cell.



The message “Template created successfully!” is displayed. At the end, the identified variables are displayed as follows:

- {client_name}
- {time_period}
- {progress}
- {blockers}
- {next_steps}
- {tone}
- {audience}
- {output_format}

This prompt template can be used to generate summaries for any project. It produces outputs with a consistent structure and is scalable for use across teams. You only need to supply values for the variables. The structure and instructions remain fixed, producing reliable and consistent output quality.



Test and finalize the prompt template

Overview

In this procedure, you'll test your prompt template with real inputs to verify if it produces reliable outputs. You'll test the template with two set of inputs to verify:

- Consistent structure across different inputs
- Proper tone adaptation
- Audience-appropriate content
- Complete coverage of requirements

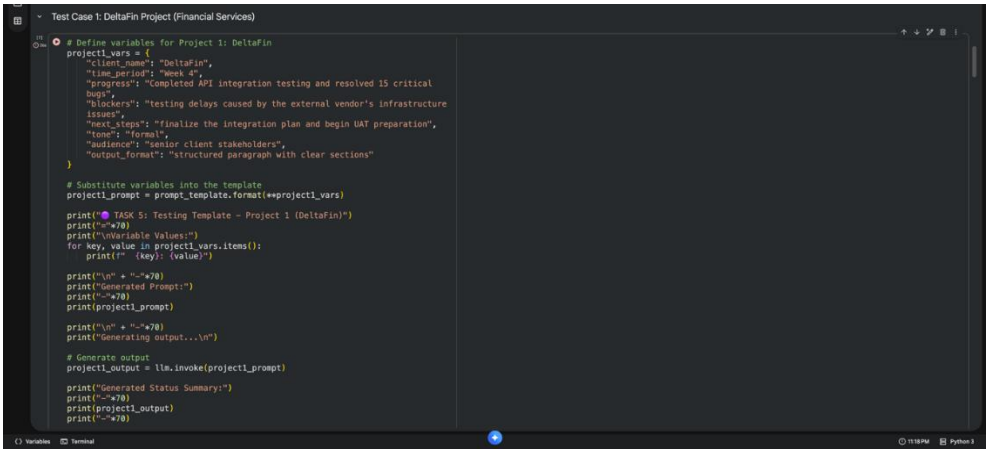
Instructions

1. Begin by testing the template for defining and loading the variables for the DeltaFin Project, a Financial services project. To do this, copy the following code, paste it into a new code cell, and then select the **Play** icon to execute it.

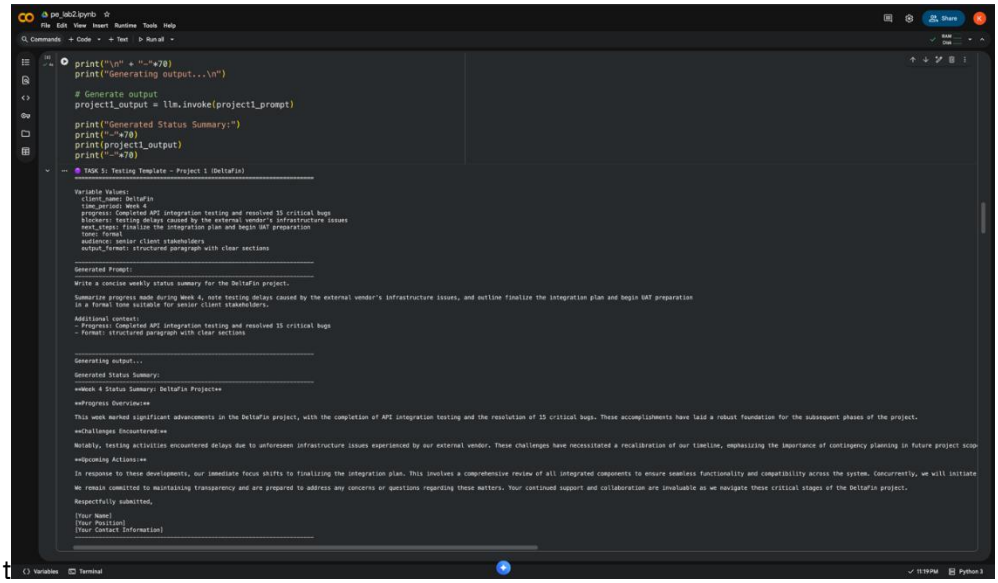
```
# Test Case 1: DeltaFin Project (Financial Services)
# Define variables for Project 1: DeltaFin
project1_vars = {
    "client_name": "DeltaFin",
    "time_period": "Week 4",
    "progress": "Completed API integration testing and resolved 15
critical bugs",
    "blockers": "Testing delays caused by the external vendor's
infrastructure issues",
    "next_steps": "Finalize the integration plan and begin UAT
preparation",
    "tone": "formal",
    "audience": "senior client stakeholders",
    "output_format": "structured paragraph with clear sections"
}

# Substitute variables into the template
project1_prompt = prompt_template.format(**project1_vars)

print("TASK 5: Testing Template - Project 1 (DeltaFin)")
print("-" * 70)
print("\nVariable Values:")
for key, value in project1_vars.items():
    print(f"{key}: {value}")

print("\n" + "-" * 70)
print("\nGenerated Prompt:")
print("-" * 70)
```

```
print(project1_prompt)
print("-" * 70)

print("\n" + "-" * 70)
print("Generating output... \n")

# Generate output
project1_output = lm.invoke(project1_prompt)

print("Generated Status Summary:")
print("-" * 70)
print(project1_output)
print("-" * 70)
```

The following screenshot shows the code pasted into the code cell.



Result: After you enter the template and edit it as per DeltaFin's project requirements, the generated result is tailored to this use case.

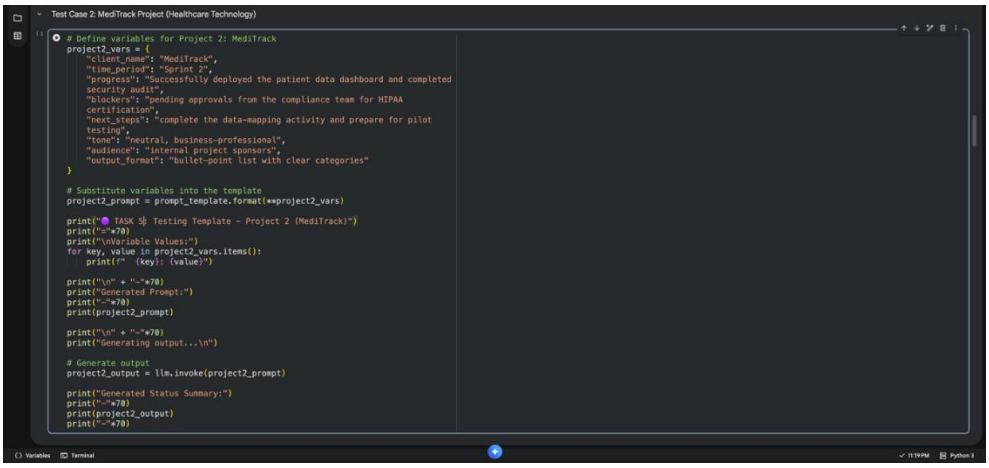


2. Now, try defining and loading the variables for the MediTrack, a project for a Healthcare organization. To do this, copy the following code, paste it into a new code cell, and then select the **Play** icon to execute it.

```
# Test Case 2: MediTrack Project (Healthcare Technology)
# Define variables for Project 2: MediTrack
project2_vars = {
    "client_name": "MediTrack",
    "time_period": "Sprint 2",
    "progress": "Successfully deployed the patient data dashboard
and completed
                security audit",
    "blockers": "pending approvals from the compliance team for
HIPAA
                certification",
    "next_steps": "complete the data-mapping activity and prepare
for pilot
                testing",
    "tone": "neutral, business-professional",
    "audience": "internal project sponsors",
    "output_format": "bullet-point list with clear categories"
}

# Substitute variables into the template
project2_prompt = prompt_template.format(**project2_vars)
```

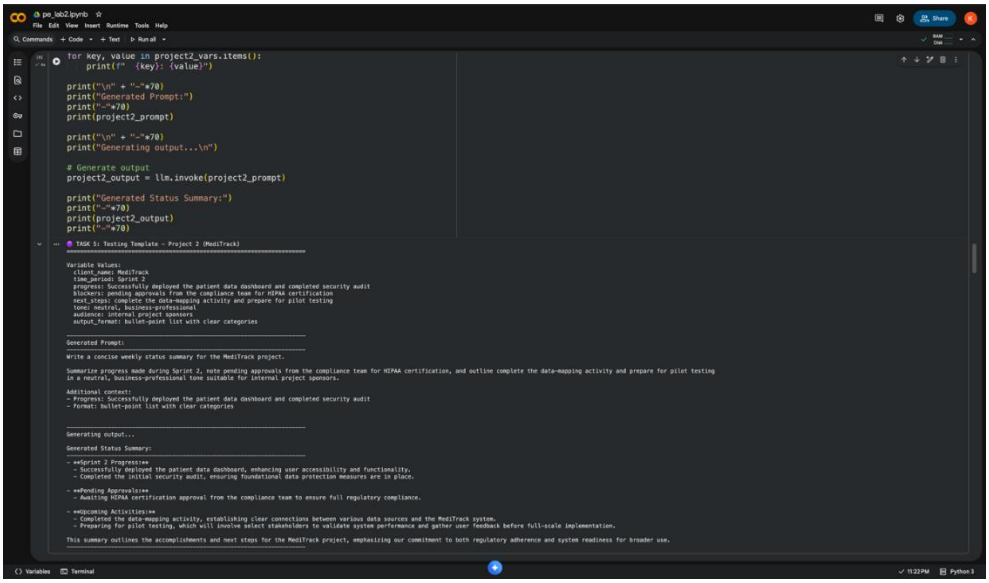
```
print("TASK 5: Testing Template - Project 2 (MediTrack)")
print("-" * 70)
print("\nVariable Values:")
for key, value in project2_vars.items():
    print(f"{key}: {value}")

print("\n" + "-" * 70)
print("\nGenerated Prompt:")
print("-" * 70)
print(project2_prompt)
print("-" * 70)

print("\n" + "-" * 70)
print("Generating output... \n")

# Generate output
project2_output = lm.invoke(project2_prompt)

print("Generated Status Summary:")
print("-" * 70)
print(project2_output)
print("-" * 70)
2. Context: The Reusable Prompt Template Definition

# Task 4: Convert the baseline prompt into a reusable template
with variables
# Using Python's string formatting with named placeholders
prompt_template = """Write a concise weekly status summary for
the
{client_name} project.

Summarize progress made during {time_period}, note {blockers},
and outline
{next_steps}
in a {tone} tone suitable for {audience}.

Additional context:
- progress: {progress}
- format: {output_format}"""
```

The following screenshot shows the code pasted into the code cell.



Result: After you enter the variables as per MediTrack's project requirements, the following output is generated.

This template produces consistent, high-quality, and well-formatted summaries that adapt to different contexts, tones, and audiences, while maintaining professional standards.



Conclusion

Congratulations! You have learned how to develop a prompt template using variables to generate output from LLMs. Your template now produces clear, consistent outputs for any project, and can be used across teams.