LAB

Optimize AI-generated code using IBM Granite models

IBM **SkillsBuild**

Contents

Introduction

This lab focuses on optimizing AI-generated code using few-shot prompt techniques with the IBM Granite model on Google Colab to optimize code generation quality. Few-shot prompting allows the model to draw from multiple examples, enabling more accurate and nuanced outputs compared to the initial zero-shot prompt approach.

This lab builds on lab 1, which focused on generating code using Granite models, by shifting the focus to refining the model output through few-shot prompting techniques. The lab focuses on optimizing UI components to create a more intuitive interface for an online bookstore, demonstrating how incorporating examples can enhance the quality and precision of the generated code.

In this lab, you'll use the IBM Granite model family hosted on Replicate to optimize Python code that enhances the design of the landing page for an online bookstore.

Software requirements

To complete this lab, you must have access to an account with Replicate to make model inference calls. You must also obtain the API token from your Replicate account, which will be securely added as a Google Colab secret.

Familiarity with Python is not necessary but can be helpful for understanding the generated code.

Objective

After completing this lab, you should be able to:

- Optimize AI-generated code using IBM Granite Code models

Lab steps

This lab requires you to complete the following steps:

- Step 1: Confirm the software setup
- Step 2: Load the Jupyter notebook and initialize the model
- Step 3: Generate code using a few-shot prompt
- Step 4: Optimize the AI- generated code

Estimated duration to complete
30 minutes

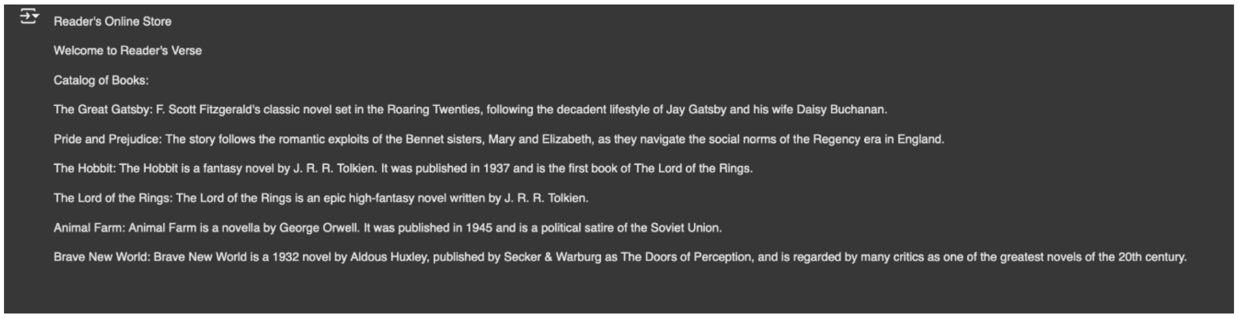
Scenario

Background

Reader's Verse is a local bookstore that plans to build an online presence by creating a website allowing its readers to search its catalog and check availability prior to visiting the physical store. You are the web designer assigned to this project and have created the initial draft of the website using zero-shot prompting technique.

Challenge

The client appreciates the initial output and requests for enhancements to improve the overall visual design and functionality of the website. You'll use an IBM Granite model to optimize the AI generated code from Lab 1 to enhance the initial design of the website.



```
Reader's Online Store
Welcome to Reader's Verse
Catalog of Books:
The Great Gatsby: F. Scott Fitzgerald's classic novel set in the Roaring Twenties, following the decadent lifestyle of Jay Gatsby and his wife Daisy Buchanan.
Pride and Prejudice: The story follows the romantic exploits of the Bennet sisters, Mary and Elizabeth, as they navigate the social norms of the Regency era in England.
The Hobbit: The Hobbit is a fantasy novel by J. R. R. Tolkien. It was published in 1937 and is the first book of The Lord of the Rings.
The Lord of the Rings: The Lord of the Rings is an epic high-fantasy novel written by J. R. R. Tolkien.
Animal Farm: Animal Farm is a novella by George Orwell. It was published in 1945 and is a political satire of the Soviet Union.
Brave New World: Brave New World is a 1932 novel by Aldous Huxley, published by Secker & Warburg as The Doors of Perception, and is regarded by many critics as one of the greatest novels of the 20th century.
```

Solution

You will use the few-shot prompting technique to optimize the AI-generated code using IBM Granite and enhance the visual design and functionality of the website.

Step 1: Confirm software setup

Overview
In this step, ensure that you have completed the following steps from lab 1:

- Step 1: Create a GitHub account
- Step 2: Create a Replicate account
- Step 3: Sign up for Google Colab

If you have not completed these steps, refer to the “Lab 1: Use IBM Granite models for code generation and programming tasks” document for details and complete the steps now.

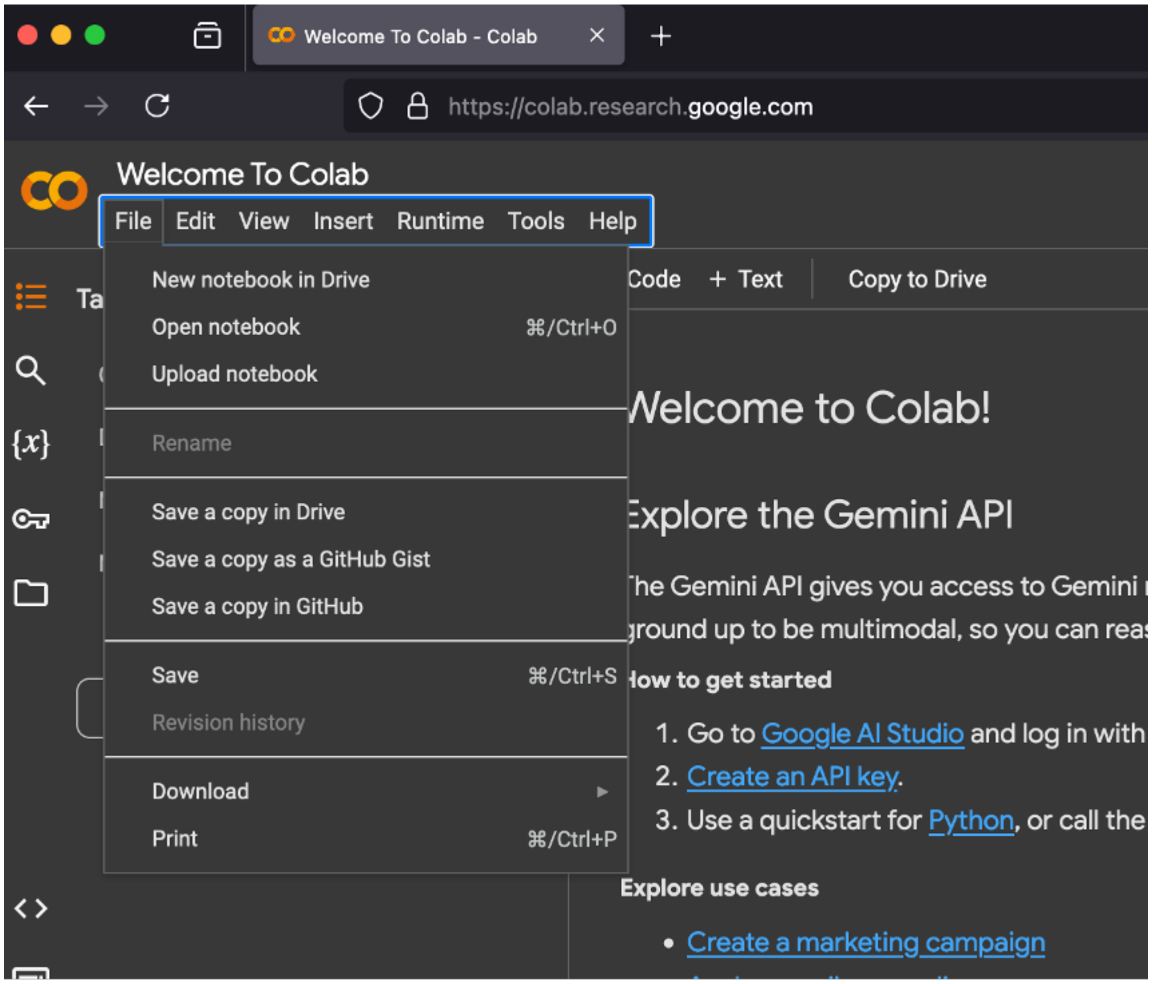
Step 2: Load the Jupyter notebook and initialize the model

Overview

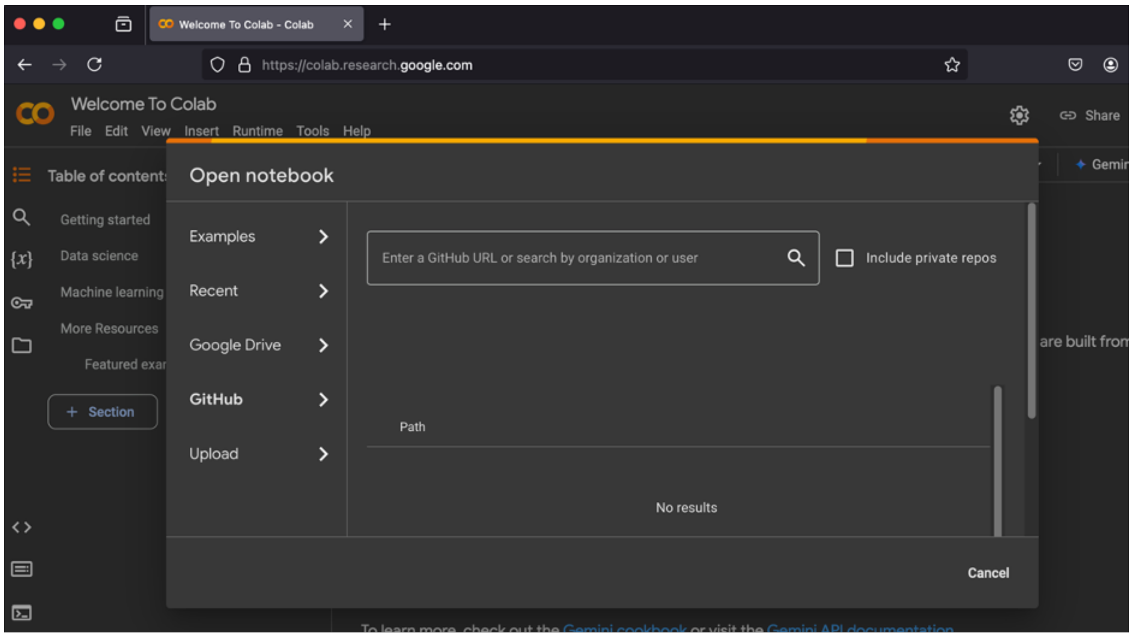
In this step, you will load a Jupyter notebook containing the code required to perform this lab. A Jupyter notebook is a shareable document that combines computer code, plain language descriptions, data, and visualizations. You'll load a Jupyter notebook file from GitHub into Google Colab and initialize the IBM Granite model to generate code for the given scenario. The notebook contains details of the few-shot prompts to optimize the initial output from lab 1.

Instructions

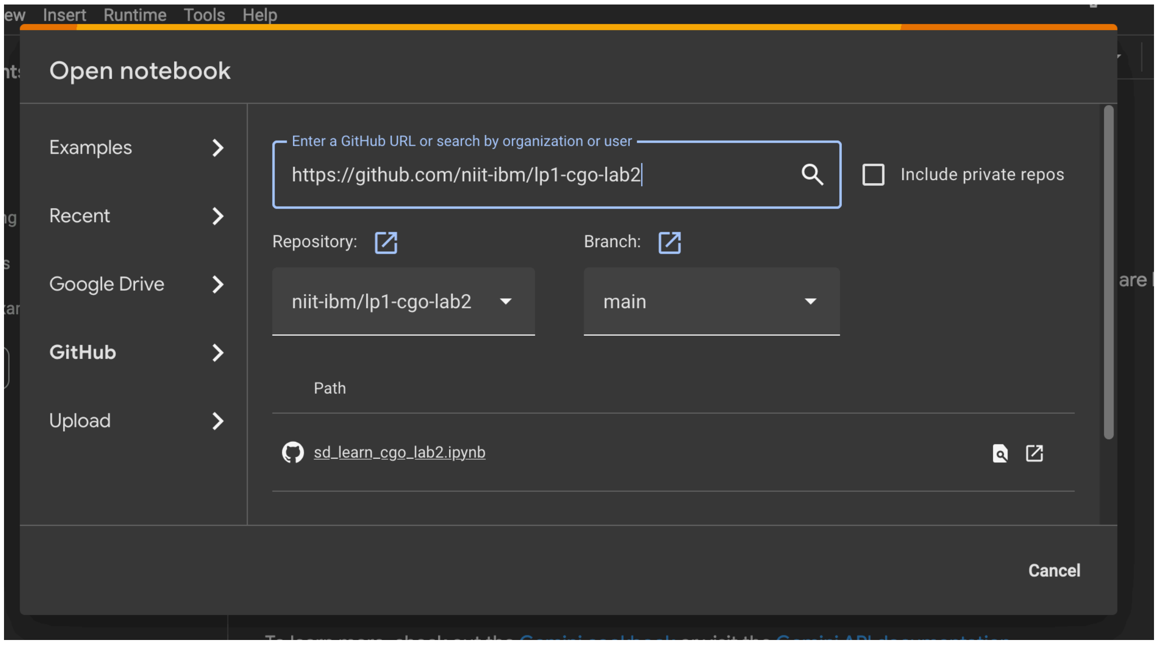
1. In your Google Colab workspace, on the **File** menu, select **Open notebook**.


Welcome To Colab - Colab
https://colab.research.google.com
Welcome To Colab
File Edit View Insert Runtime Tools Help
New notebook in Drive
Open notebook ⌘/Ctrl+O
Upload notebook
Rename
Save a copy in Drive
Save a copy as a GitHub Gist
Save a copy in GitHub
Save ⌘/Ctrl+S
Revision history
Download
Print ⌘/Ctrl+P
Code + Text Copy to Drive
Welcome to Colab!
Explore the Gemini API
The Gemini API gives you access to Gemini
ground up to be multimodal, so you can rea
How to get started
1. Go to Google AI Studio and log in with
2. Create an API key.
3. Use a quickstart for Python, or call the
Explore use cases
Create a marketing campaign

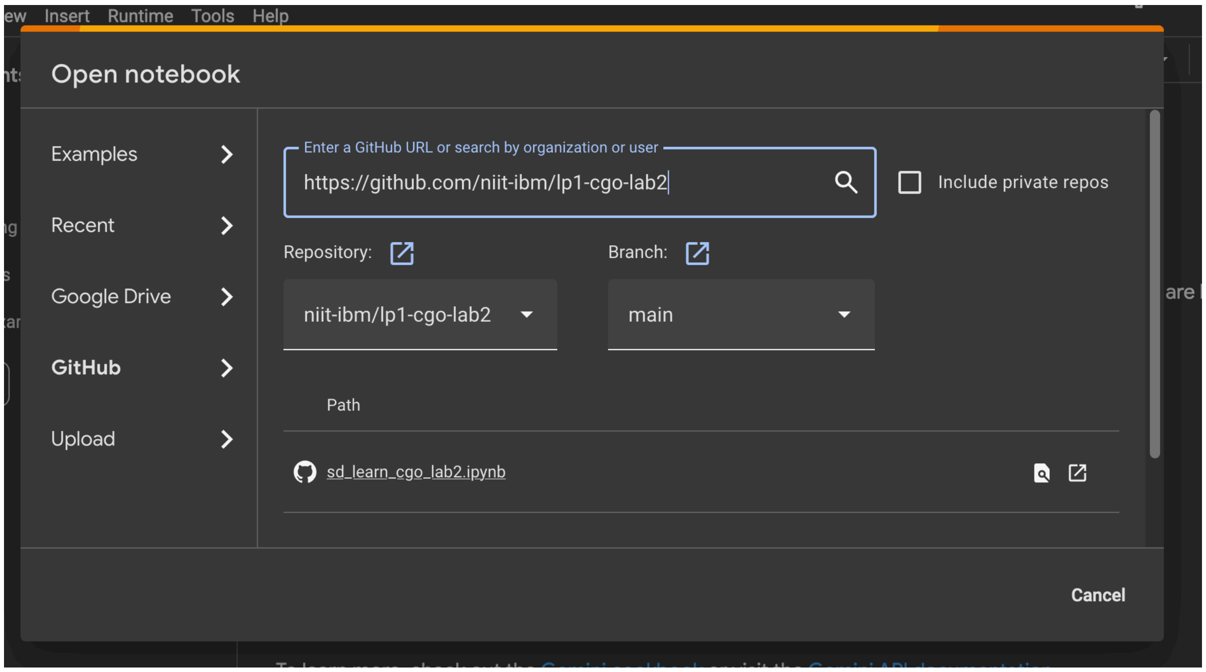
2. Select the **GitHub** tab.



3. In the **Enter a GitHub URL or search by organization or user** field, copy the following URL: https://github.com/niit-ibm/lp1-cgo-lab2 and select the **magnifying glass** icon.



4. Select **main** in the **Branch** section.


ew Insert Runtime Tools Help
Open notebook
Examples
Recent
Google Drive
GitHub
Upload
Enter a GitHub URL or search by organization or user
Include private repos
Cancel

https://github.com/niit-ibm/lp1-cgo-lab2

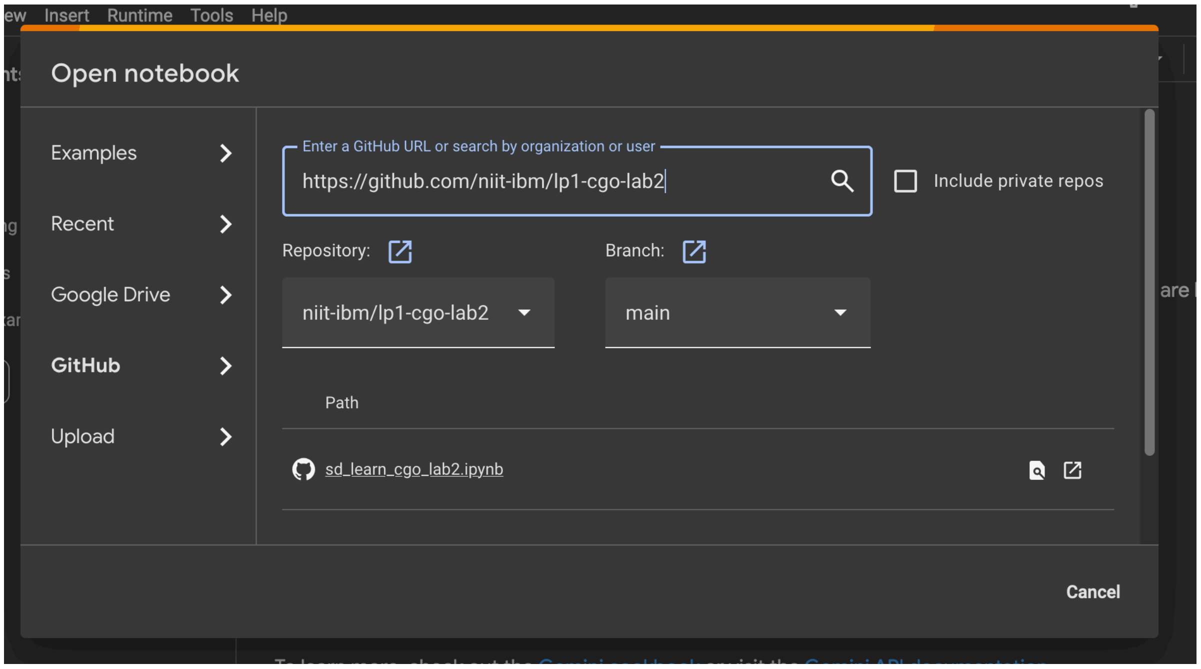
Repository: niit-ibm/lp1-cgo-lab2

Branch: main

Path

sd_learn_cgo_lab2.ipynb

5. Select the **sd_learn_cgo_lab2.ipynb** notebook in the **Path** section to open the notebook in Colab.


ew Insert Runtime Tools Help
Open notebook
Examples
Recent
Google Drive
GitHub
Upload
Enter a GitHub URL or search by organization or user
Include private repos
Cancel

https://github.com/niit-ibm/lp1-cgo-lab2

Repository: niit-ibm/lp1-cgo-lab2

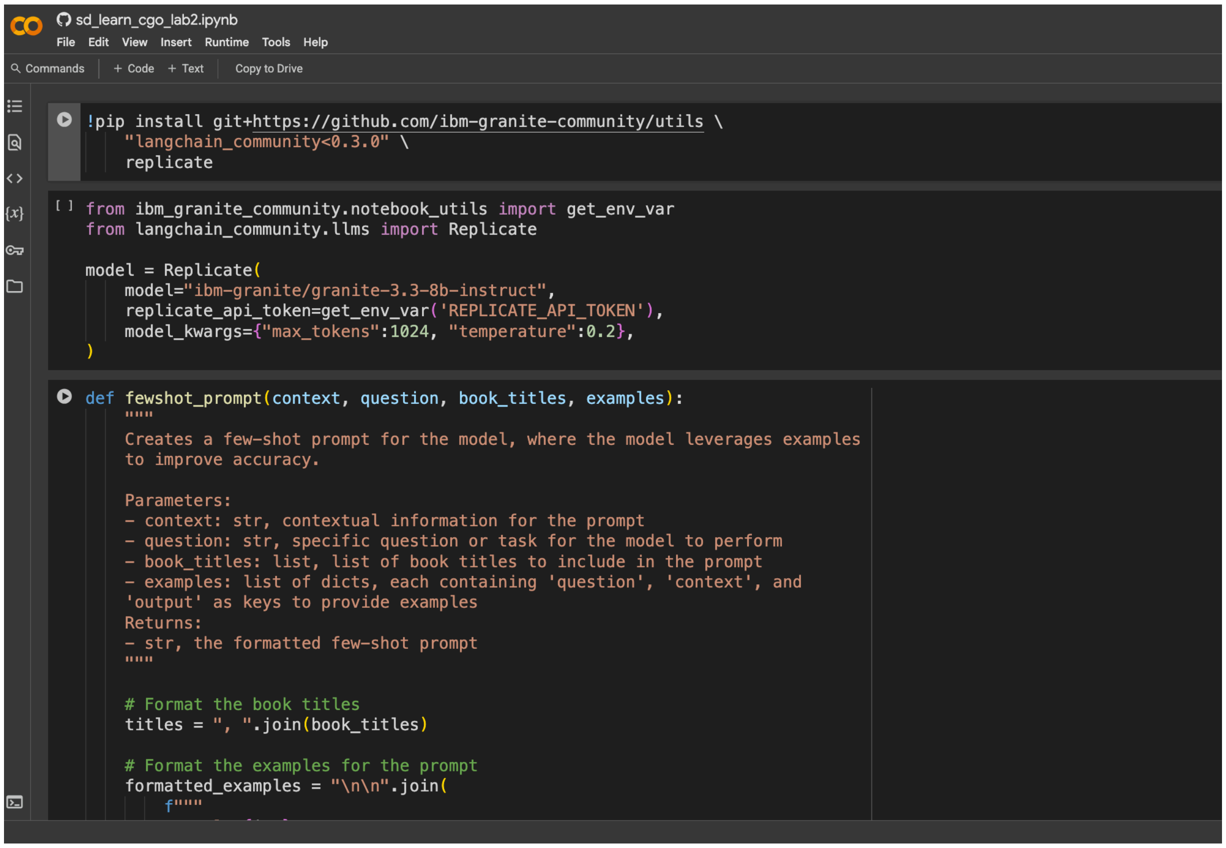
Branch: main

Path

sd_learn_cgo_lab2.ipynb

6. You'll now have the sd_learn_cgo_lab2 notebook opened in the Colab workspace.

   Note that each row in the notebook is referred to as a **cell**. The cells in the notebook are not numbered. The notebook is organized in such a way that you'll need to run the code in each cell sequentially, beginning from the first cell, to complete this lab.

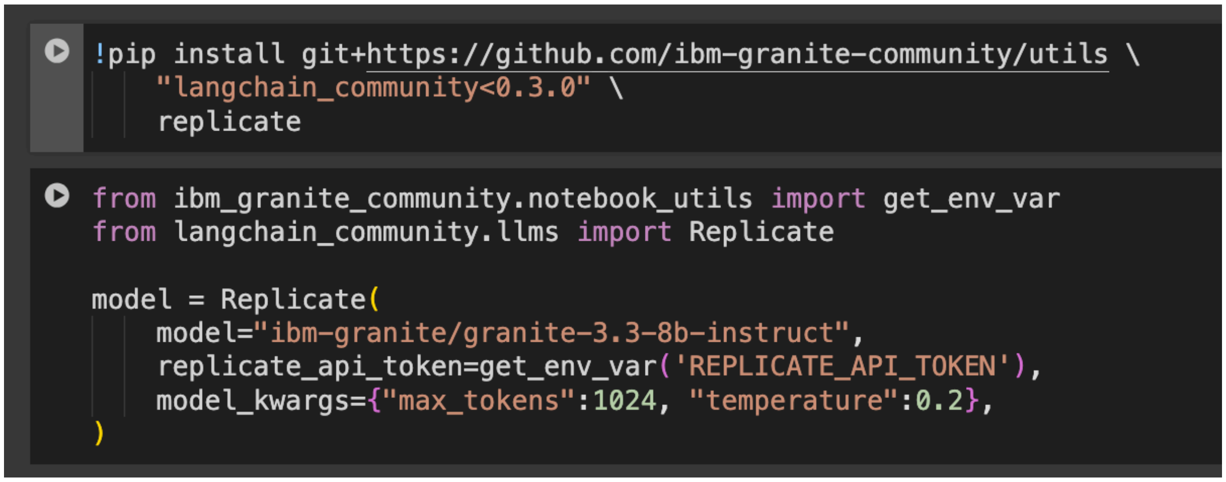

sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Copy to Drive

```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

```
def fewshot_prompt(context, question, book_titles, examples):
    """
    Creates a few-shot prompt for the model, where the model leverages examples
    to improve accuracy.

    Parameters:
    - context: str, contextual information for the prompt
    - question: str, specific question or task for the model to perform
    - book_titles: list, list of book titles to include in the prompt
    - examples: list of dicts, each containing 'question', 'context', and
    'output' as keys to provide examples
    Returns:
    - str, the formatted few-shot prompt
    """

    # Format the book titles
    titles = ", ".join(book_titles)

    # Format the examples for the prompt
    formatted_examples = "\n\n".join(
        f"""
```

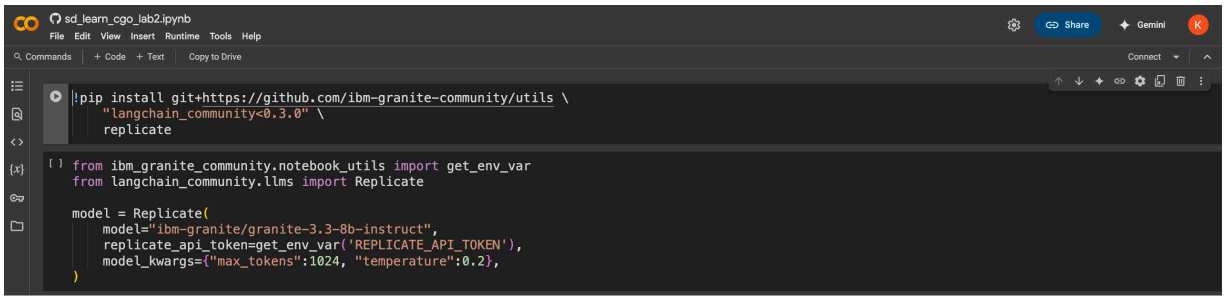
7. You'll need to start a new instance in the Google Colab runtime to run the code in the Jupyter notebook. Select **Connect** on the Google Colab navigation bar.



```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

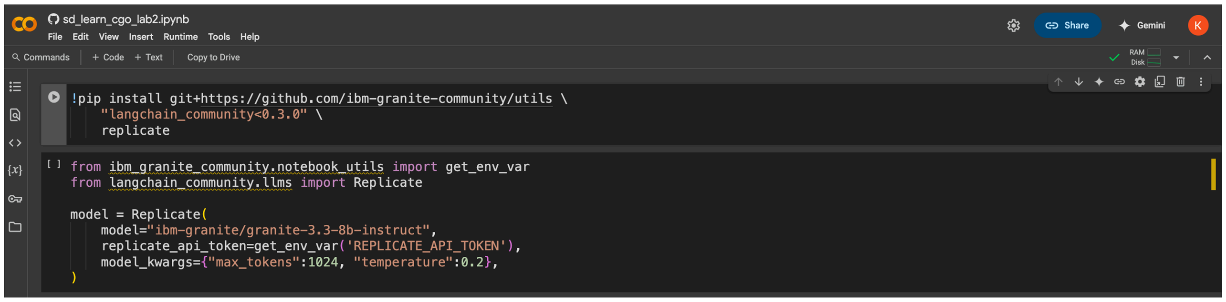
8. Select **Connect to a hosted runtime** on the **Connect** menu.


sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Copy to Drive
Share
Gemini
Connect

```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

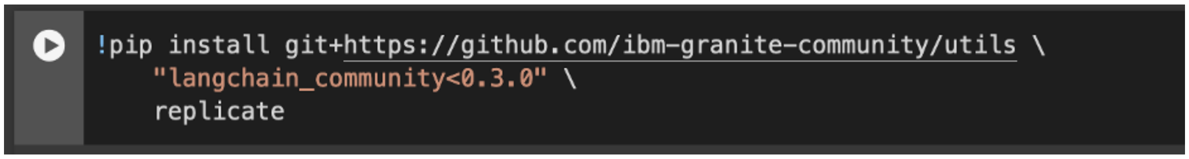
9. A green checkmark indicates that you have successfully connected to the hosted runtime.


sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Copy to Drive
Share
Gemini
RAM
Disk

```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

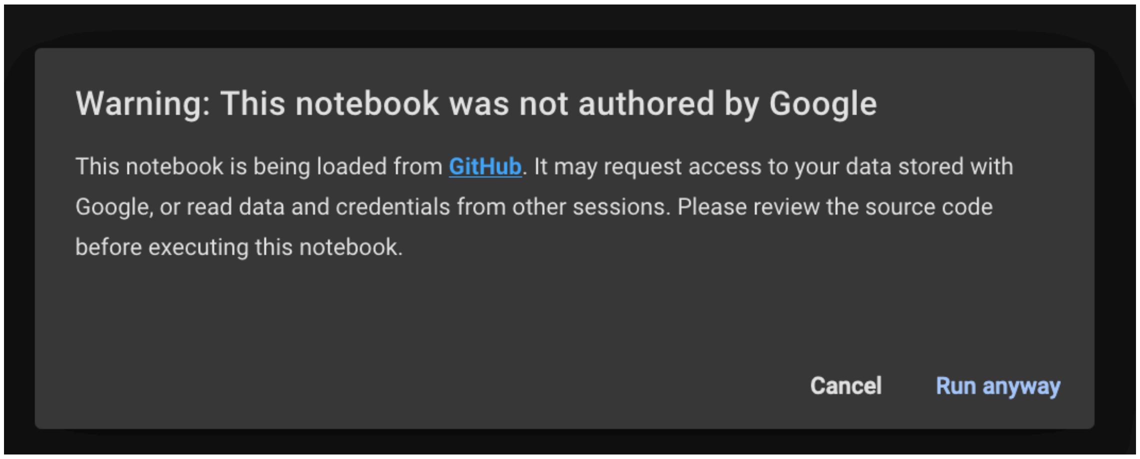
10. In the first cell of the notebook, select the **Play** button to install the required libraries from the Granite community.



```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

Note: You'll need to run the cells sequentially beginning from the first cell in the notebook.

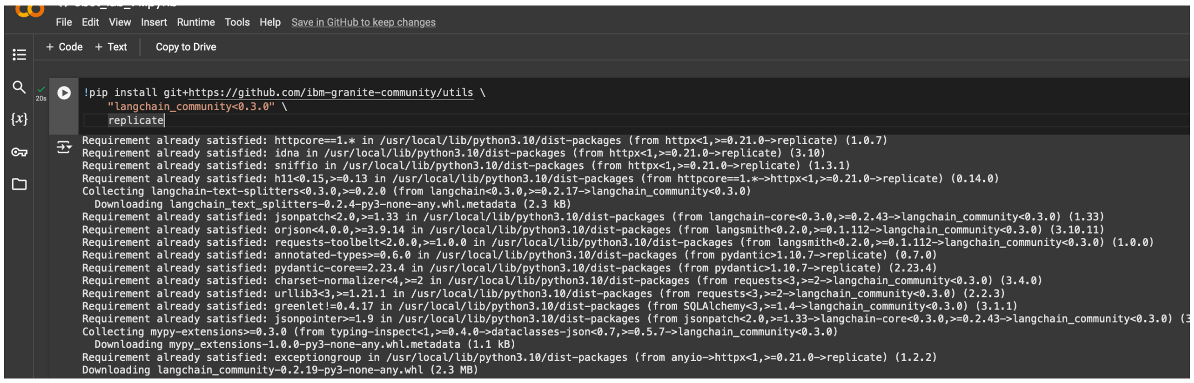
11. Select **Run anyway** to proceed loading the required libraries.



Warning: This notebook was not authored by Google

This notebook is being loaded from GitHub. It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook.

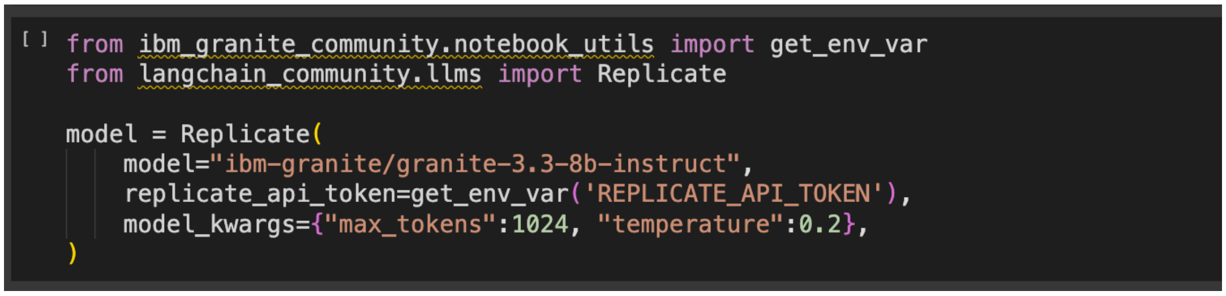
12. A green checkmark appears beside the **Play** button to indicate that the required libraries are successfully installed.


File Edit View Insert Runtime Tools Help Save in GitHub to keep changes
+ Code + Text Copy to Drive
20s

```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

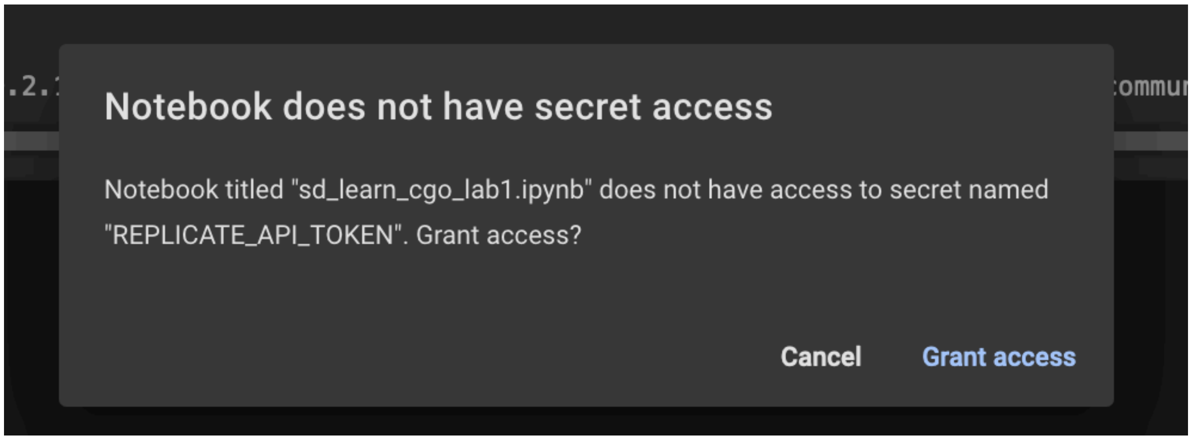
```
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.21.0->replicate) (1.0.7)
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.21.0->replicate) (3.10)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.21.0->replicate) (1.3.1)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx<1,>=0.21.0->replicate) (0.14.0)
Collecting langchain-text-splitters<0.3.0,>=0.2.0 (from langchain<0.3.0,>=0.2.17->langchain_community<0.3.0)
  Downloading langchain_text_splitters-0.2.4-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.43->langchain_community<0.3.0) (1.33)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.112->langchain_community<0.3.0) (3.10.11)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.112->langchain_community<0.3.0) (1.0.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>1.10.7->replicate) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (from pydantic>1.10.7->replicate) (2.23.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain_community<0.3.0) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain_community<0.3.0) (2.2.3)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain_community<0.3.0) (3.1.1)
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain-core<0.3.0,>=0.2.43->langchain_community<0.3.0) (3
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain_community<0.3.0)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx<1,>=0.21.0->replicate) (1.2.2)
Downloading langchain_community-0.2.19-py3-none-any.whl (2.3 MB)
```

13. In the second cell of the notebook, select the **Play** button to initialize the IBM Granite model using Replicate for code generation.



```
[ ] from ibm_granite_community.notebook_utils import get_env_var
    from langchain_community.llms import Replicate

    model = Replicate(
        model="ibm-granite/granite-3.3-8b-instruct",
        replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
        model_kwargs={"max_tokens":1024, "temperature":0.2},
    )
```

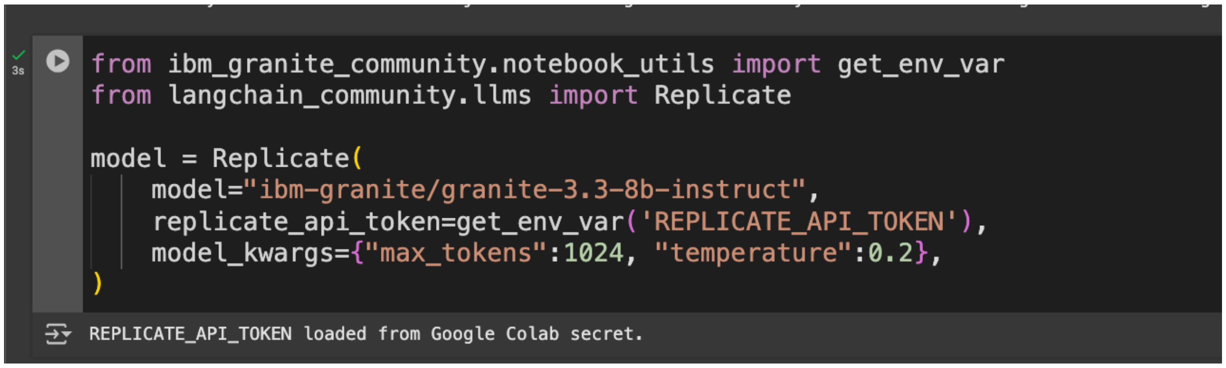
14. Select **Grant access** to proceed loading the model from Replicate.



Notebook does not have secret access

Notebook titled "sd_learn_cgo_lab1.ipynb" does not have access to secret named "REPLICATE_API_TOKEN". Grant access?

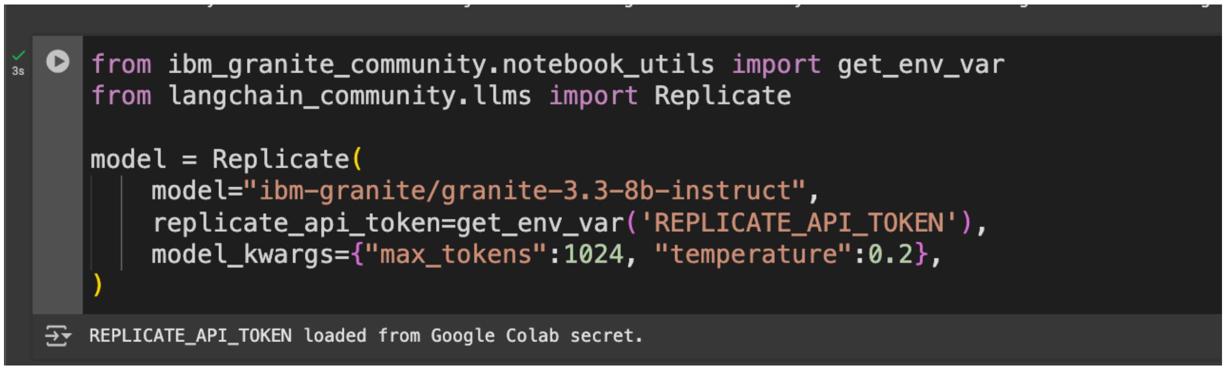
15. The following message displays at the bottom of the cell: “REPLICATE_API_TOKEN loaded from Google Colab secret”.



```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

```
REPLICATE_API_TOKEN loaded from Google Colab secret.
```

16. A green checkmark besides the **Play** button indicates that the IBM Granite model is initialized using Replicate. You are now ready to prompt the model to generate code for the given scenario.



```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

```
REPLICATE_API_TOKEN loaded from Google Colab secret.
```

Step 3: Generate code using a few-shot prompt

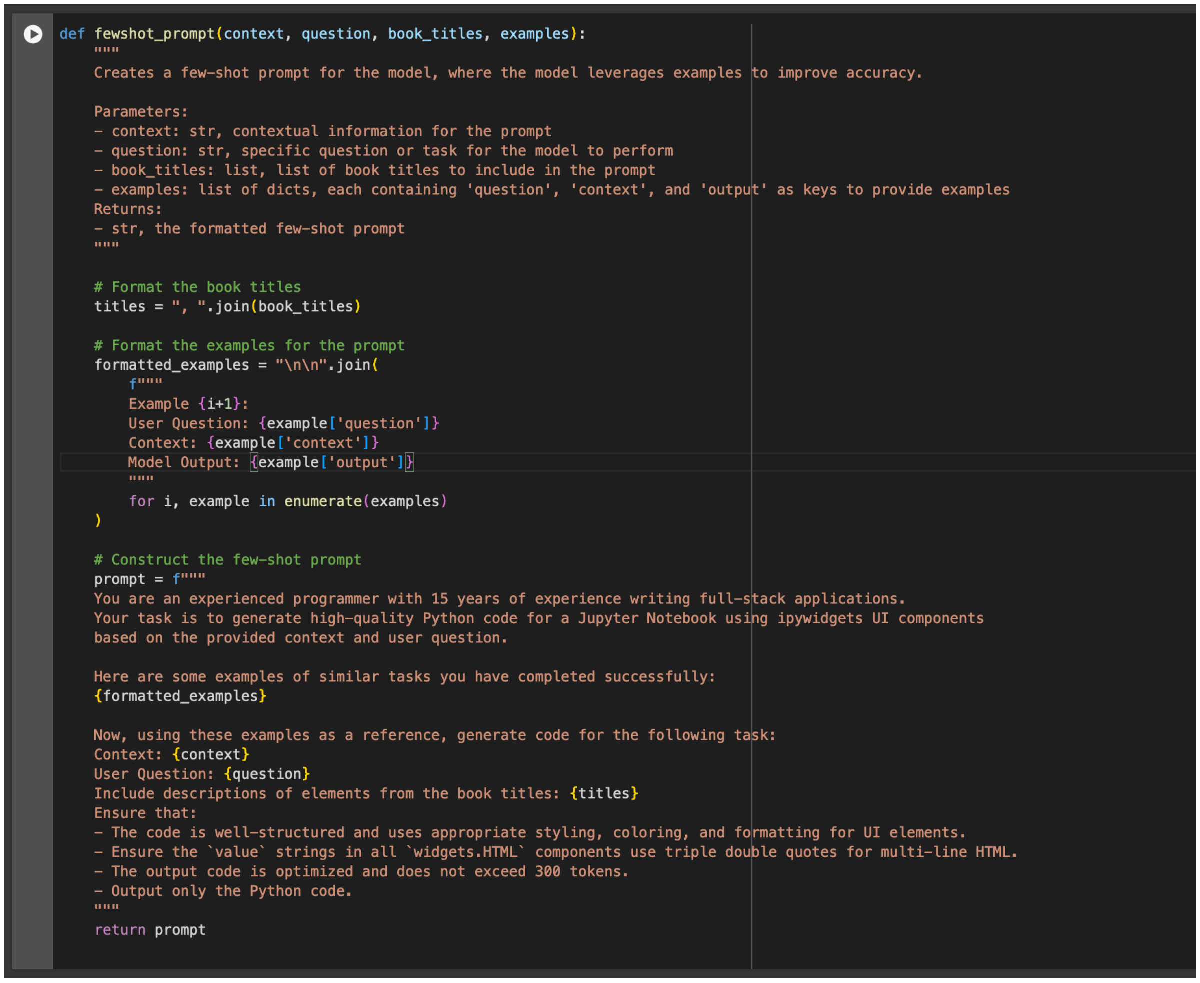
Overview

In this step, you'll prompt the IBM Granite model to generate the code for the given scenario using few-shot prompting. You'll define a few-shot prompt for the model and run the prompt to enhance the landing page of the Reader's Verse website by including interactive elements.

Instructions

1. Define a function that creates a few-shot prompt for the AI model using examples to improve the quality of the output. This function will help optimize the code to create interactive UI components for the landing page of the Reader's Verse website.

   In the third cell of the notebook, select the **Play** button to define the **fewshot_prompt** function, which contains a set of instructions guiding the model on the output to generate. In this scenario, the instructions inform the model to optimize the UI for the online bookstore based on the given parameters: context, question, book titles, and examples.

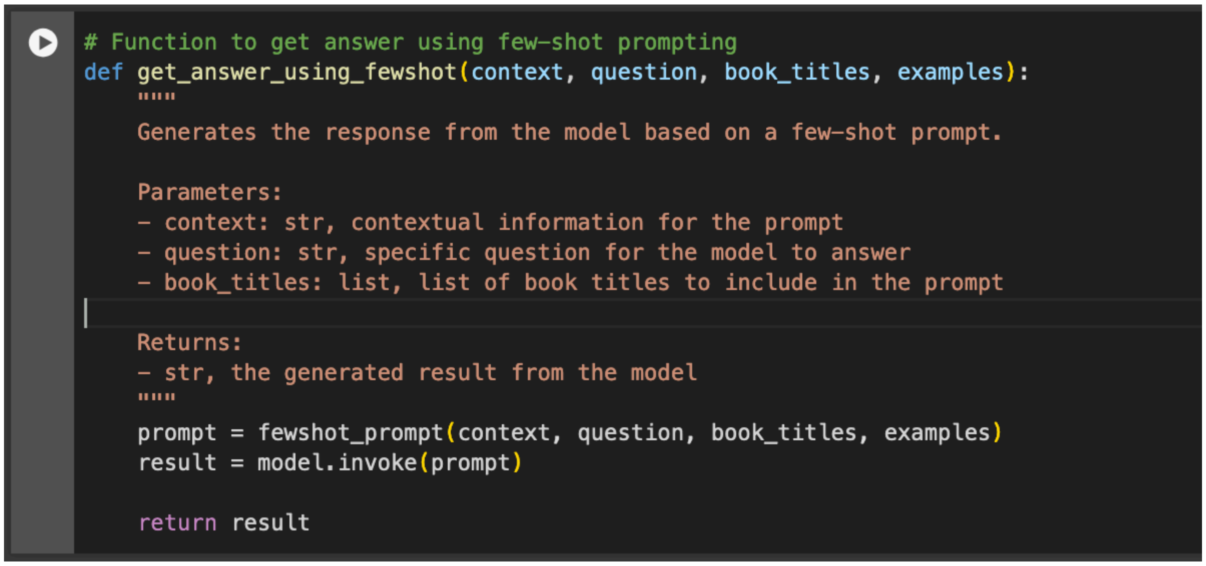


```python
def fewshot_prompt(context, question, book_titles, examples):
    """
    Creates a few-shot prompt for the model, where the model leverages examples to improve accuracy.

    Parameters:
    - context: str, contextual information for the prompt
    - question: str, specific question or task for the model to perform
    - book_titles: list, list of book titles to include in the prompt
    - examples: list of dicts, each containing 'question', 'context', and 'output' as keys to provide examples
    Returns:
    - str, the formatted few-shot prompt
    """

    # Format the book titles
    titles = ", ".join(book_titles)

    # Format the examples for the prompt
    formatted_examples = "\n\n".join(
        f"""
        Example {i+1}:
        User Question: {example['question']}
        Context: {example['context']}
        Model Output: {example['output']}
        """
        for i, example in enumerate(examples)
    )

    # Construct the few-shot prompt
    prompt = f"""
    You are an experienced programmer with 15 years of experience writing full-stack applications.
    Your task is to generate high-quality Python code for a Jupyter Notebook using ipywidgets UI components
    based on the provided context and user question.

    Here are some examples of similar tasks you have completed successfully:
    {formatted_examples}

    Now, using these examples as a reference, generate code for the following task:
    Context: {context}
    User Question: {question}
    Include descriptions of elements from the book titles: {titles}
    Ensure that:
    - The code is well-structured and uses appropriate styling, coloring, and formatting for UI elements.
    - Ensure the `value` strings in all `widgets.HTML` components use triple double quotes for multi-line HTML.
    - The output code is optimized and does not exceed 300 tokens.
    - Output only the Python code.
    """
    return prompt
```

Note: You won't see any output when you run this cell in the notebook.

2. Next, define a function to generate a response from the model using the few-shot prompt. This function sends the instructions defined in the few-shot prompt to the model and gets the desired response from the model.

   In the fourth cell of the notebook, select the **Play** button to define the **get_answer_using_fewshot** function.

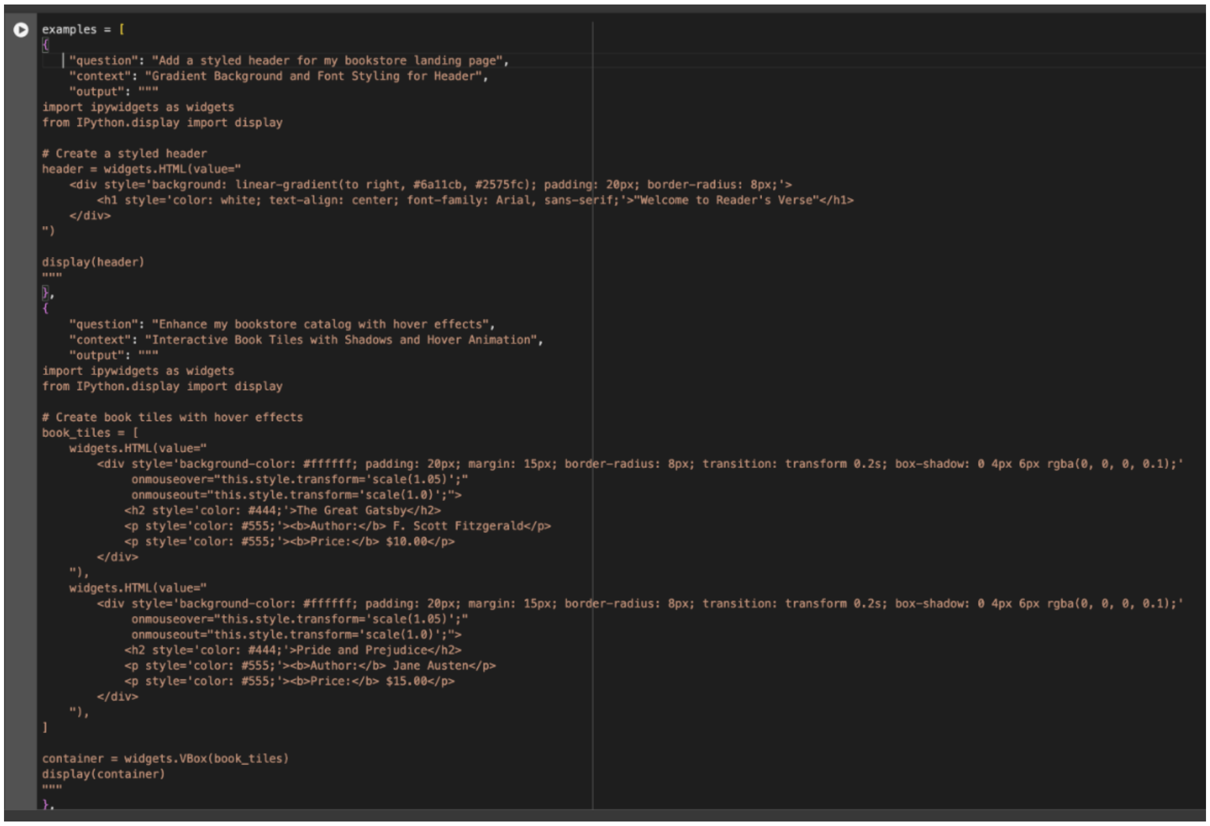


```python
# Function to get answer using few-shot prompting
def get_answer_using_fewshot(context, question, book_titles, examples):
    """
    Generates the response from the model based on a few-shot prompt.

    Parameters:
    - context: str, contextual information for the prompt
    - question: str, specific question for the model to answer
    - book_titles: list, list of book titles to include in the prompt

    Returns:
    - str, the generated result from the model
    """
    prompt = fewshot_prompt(context, question, book_titles, examples)
    result = model.invoke(prompt)

    return result
```

   Note: You won't see any output when you run this cell in the notebook.

3. Next, you'll prepare examples of interactive elements with questions, context, and output parameters to guide the IBM Granite model in generating an output that matches the style and format requested by the client. In this scenario, examples include formats for header styles, hover effects, and other interactive UI elements such as those requested by the client.

   In the fifth cell of the notebook, select the **Play** button to prepare the examples that the model will use while invoking the **get_answer_using_fewshot** function in the next step.

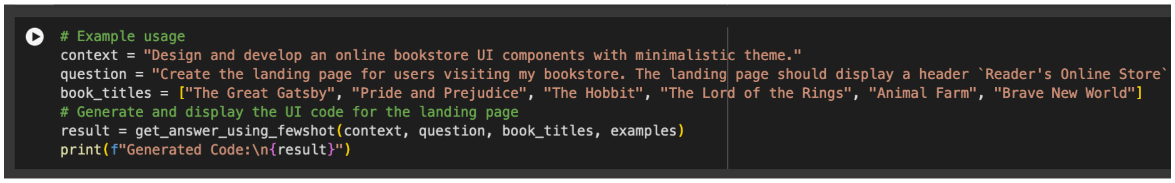


```
examples = [
{
    "question": "Add a styled header for my bookstore landing page",
    "context": "Gradient Background and Font Styling for Header",
    "output": """
import ipywidgets as widgets
from IPython.display import display

# Create a styled header
header = widgets.HTML(value="
    <div style='background: linear-gradient(to right, #6a11cb, #2575fc); padding: 20px; border-radius: 8px;'>
        <h1 style='color: white; text-align: center; font-family: Arial, sans-serif;'>"Welcome to Reader's Verse"</h1>
    </div>
")

display(header)
"""
},
{
    "question": "Enhance my bookstore catalog with hover effects",
    "context": "Interactive Book Tiles with Shadows and Hover Animation",
    "output": """
import ipywidgets as widgets
from IPython.display import display

# Create book tiles with hover effects
book_tiles = [
    widgets.HTML(value="
        <div style='background-color: #ffffff; padding: 20px; margin: 15px; border-radius: 8px; transition: transform 0.2s; box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);'
             onmouseover="this.style.transform='scale(1.05)';"
             onmouseout="this.style.transform='scale(1.0)';">
            <h2 style='color: #444;'>The Great Gatsby</h2>
            <p style='color: #555;'><b>Author:</b> F. Scott Fitzgerald</p>
            <p style='color: #555;'><b>Price:</b> $10.00</p>
        </div>
    "),
    widgets.HTML(value="
        <div style='background-color: #ffffff; padding: 20px; margin: 15px; border-radius: 8px; transition: transform 0.2s; box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);'
             onmouseover="this.style.transform='scale(1.05)';"
             onmouseout="this.style.transform='scale(1.0)';">
            <h2 style='color: #444;'>Pride and Prejudice</h2>
            <p style='color: #555;'><b>Author:</b> Jane Austen</p>
            <p style='color: #555;'><b>Price:</b> $15.00</p>
        </div>
    "),
]

container = widgets.VBox(book_tiles)
display(container)
"""
},
```

   Note: You won't see any output when you run this cell in the notebook.

4. Next, you'll run the **get_answer_using_fewshot** function to generate the code output for the Reader's Verse landing page. The few-shot response refines the zero-shot output by using examples and contextual guidance to generate the code for the given scenario. The few-shot prompt includes pre-provided examples that demonstrate how to generate Python code for creating an interactive UI for the Reader's Verse landing page.

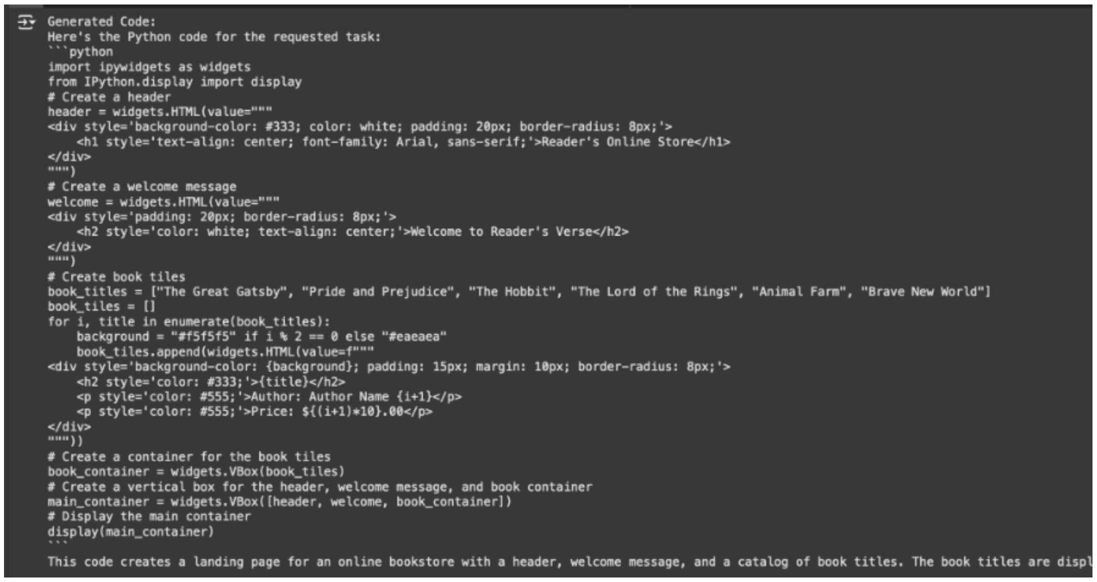
   In the sixth cell of the notebook, select the **Play** button to run the **get_answer_using_fewshot** function.



```
# Example usage
context = "Design and develop an online bookstore UI components with minimalistic theme."
question = "Create the landing page for users visiting my bookstore. The landing page should display a header `Reader's Online Store`
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The Lord of the Rings", "Animal Farm", "Brave New World"]
# Generate and display the UI code for the landing page
result = get_answer_using_fewshot(context, question, book_titles, examples)
print(f"Generated Code:\n{result}")
```

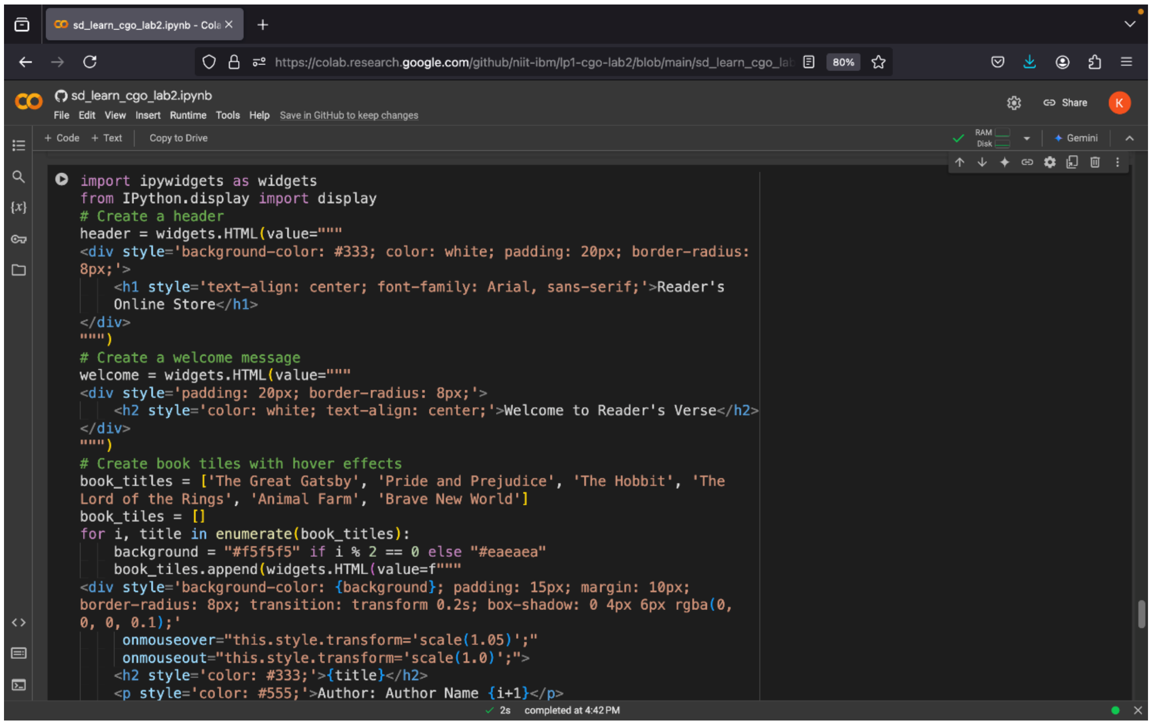
5. The **get_answer_using_fewshot** function generates and displays the Python code for the landing page UI following the code cell. This code will create the landing page including a title, a catalog of book titles, and interactive UI elements.

   Note: When you generate the code, it might be different from the following code output. These minor differences are normal and will not affect the overall functionality or outcome of the lab.



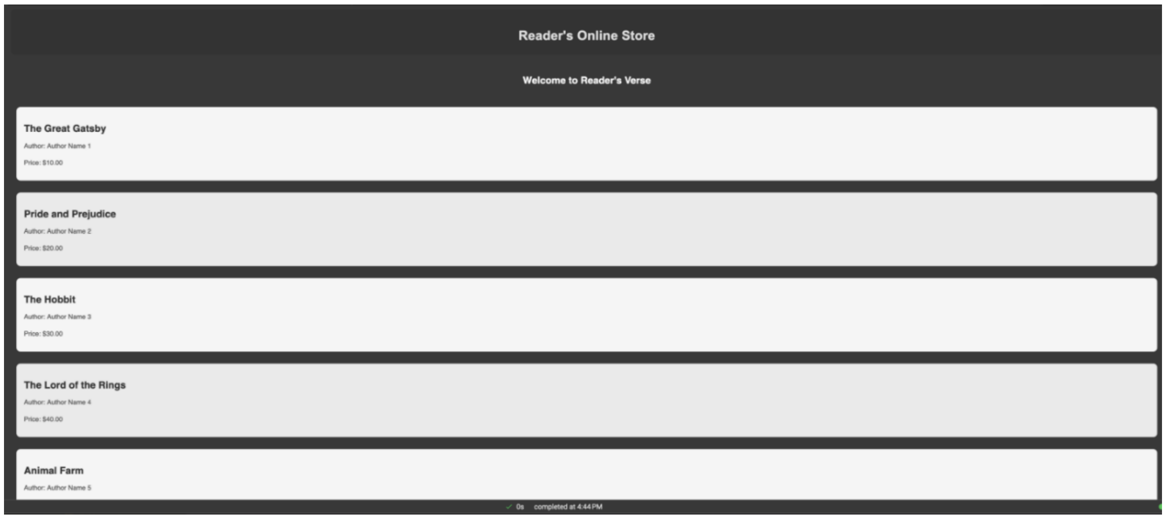
~~~
Generated Code:
Here's the Python code for the requested task:
```python
import ipywidgets as widgets
from IPython.display import display
# Create a header
header = widgets.HTML(value="""
<div style='background-color: #333; color: white; padding: 20px; border-radius: 8px;'>
    <h1 style='text-align: center; font-family: Arial, sans-serif;'>Reader's Online Store</h1>
</div>
""")
# Create a welcome message
welcome = widgets.HTML(value="""
<div style='padding: 20px; border-radius: 8px;'>
    <h2 style='color: white; text-align: center;'>Welcome to Reader's Verse</h2>
</div>
""")
# Create book tiles
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The Lord of the Rings", "Animal Farm", "Brave New World"]
book_tiles = []
for i, title in enumerate(book_titles):
    background = "#f5f5f5" if i % 2 == 0 else "#eaeaea"
    book_tiles.append(widgets.HTML(value=f"""
<div style='background-color: {background}; padding: 15px; margin: 10px; border-radius: 8px;'>
    <h2 style='color: #333;'>{title}</h2>
    <p style='color: #555;'>Author: Author Name {i+1}</p>
    <p style='color: #555;'>Price: ${(i+1)*10}.00</p>
</div>
"""))
# Create a container for the book tiles
book_container = widgets.VBox(book_tiles)
# Create a vertical box for the header, welcome message, and book container
main_container = widgets.VBox([header, welcome, book_container])
# Display the main container
display(main_container)
```
This code creates a landing page for an online bookstore with a header, welcome message, and a catalog of book titles. The book titles are displ
~~~

6. For ease of execution, the AI-generated output code is pre-populated in a new cell of the Jupyter notebook. Select the **Play** button in the seventh cell of the notebook to validate the Python code output generated by the IBM Granite model for the given scenario.


sd_learn_cgo_lab2.ipynb - Cola
https://colab.research.google.com/github/niit-ibm/lp1-cgo-lab2/blob/main/sd_learn_cgo_lab
80%
sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help Save in GitHub to keep changes
Share
+ Code + Text Copy to Drive
RAM Disk
Gemini
2s completed at 4:42 PM

```
import ipywidgets as widgets
from IPython.display import display
# Create a header
header = widgets.HTML(value="""
<div style='background-color: #333; color: white; padding: 20px; border-radius:
8px;'>
    <h1 style='text-align: center; font-family: Arial, sans-serif;'>Reader's
    Online Store</h1>
</div>
""")
# Create a welcome message
welcome = widgets.HTML(value="""
<div style='padding: 20px; border-radius: 8px;'>
    <h2 style='color: white; text-align: center;'>Welcome to Reader's Verse</h2>
</div>
""")
# Create book tiles with hover effects
book_titles = ['The Great Gatsby', 'Pride and Prejudice', 'The Hobbit', 'The
Lord of the Rings', 'Animal Farm', 'Brave New World']
book_tiles = []
for i, title in enumerate(book_titles):
    background = "#f5f5f5" if i % 2 == 0 else "#eaeaea"
    book_tiles.append(widgets.HTML(value=f"""
<div style='background-color: {background}; padding: 15px; margin: 10px;
border-radius: 8px; transition: transform 0.2s; box-shadow: 0 4px 6px rgba(0,
0, 0, 0.1);'
     onmouseover="this.style.transform='scale(1.05)';"
     onmouseout="this.style.transform='scale(1.0)';">
    <h2 style='color: #333;'>{title}</h2>
    <p style='color: #555;'>Author: Author Name {i+1}</p>
```

7. The output should display the bookstore's landing page UI following the code cell, featuring the following components:
    - A stylized header
    - A formatted and organized listing of book titles
    - A minimalistic theme



Reader's Online Store

Welcome to Reader's Verse

The Great Gatsby
Author: Author Name 1
Price: $10.00

Pride and Prejudice
Author: Author Name 2
Price: $20.00

The Hobbit
Author: Author Name 3
Price: $30.00

The Lord of the Rings
Author: Author Name 4
Price: $40.00

Animal Farm
Author: Author Name 5

✓ 0s completed at 4:44 PM

Step 4: Optimize the AI-generated code

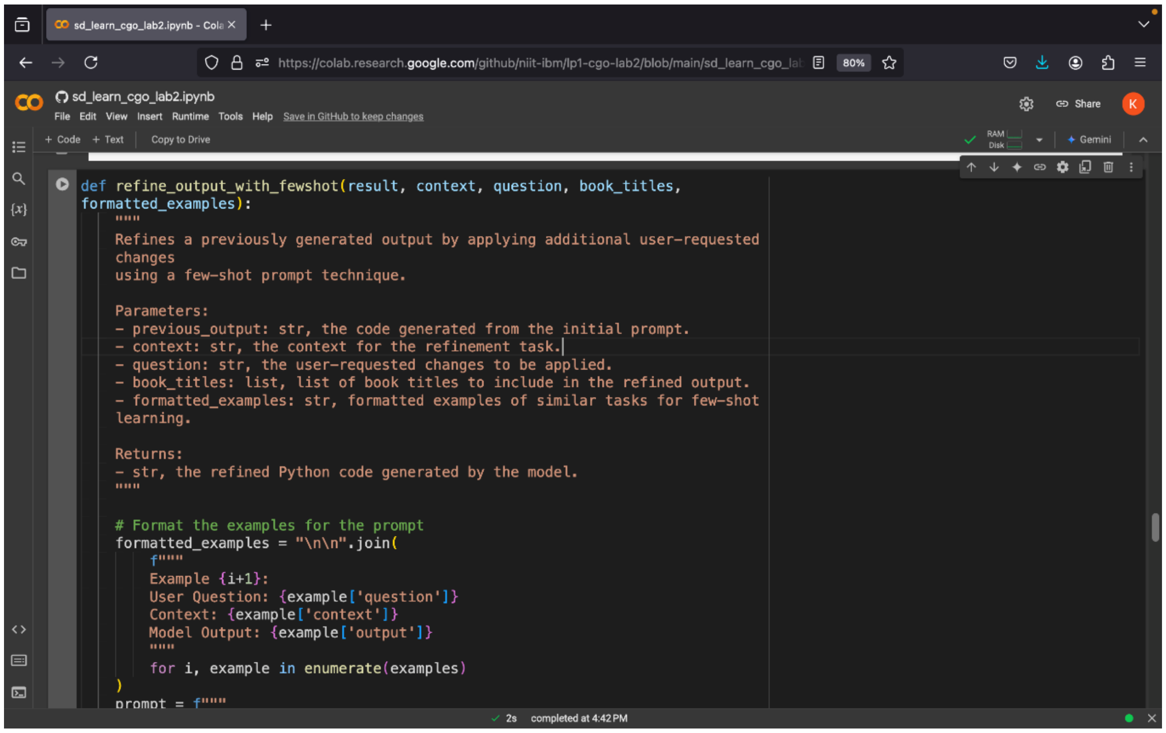
Overview

In this step, you'll optimize the AI-generated output of the few-shot prompt by adding design elements to the landing page of the Reader's Verse website.

Instructions

1. Define and execute a function that creates a few-shot prompt for the model by providing examples to guide the model. The **refine_output_with_fewshot** function refines the code to improve the formatting, styling, and functionality of the landing page.

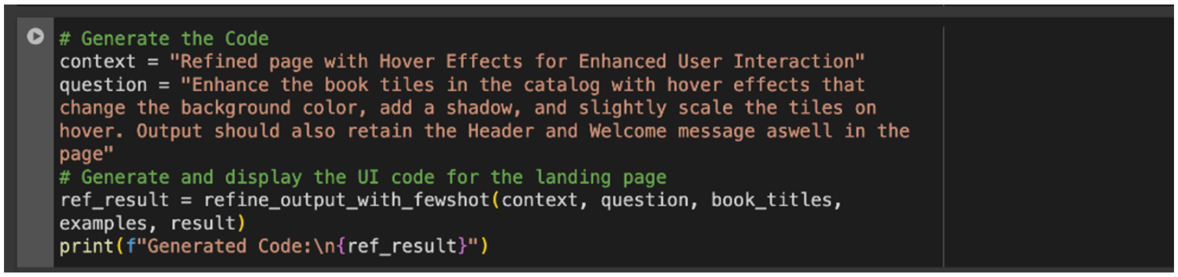
   In the eighth cell of the notebook, select the **Play** button to define the **refine_output_with_fewshot** function which contains a set of instructions guiding the model on the output to generate. In this scenario, the instructions inform the model to refine code for the landing page based on the given parameters: output from the few-shot prompt, context, question, book titles, and formatted examples.



```python
def refine_output_with_fewshot(result, context, question, book_titles,
formatted_examples):
    """
    Refines a previously generated output by applying additional user-requested
    changes
    using a few-shot prompt technique.

    Parameters:
    - previous_output: str, the code generated from the initial prompt.
    - context: str, the context for the refinement task.
    - question: str, the user-requested changes to be applied.
    - book_titles: list, list of book titles to include in the refined output.
    - formatted_examples: str, formatted examples of similar tasks for few-shot
    learning.

    Returns:
    - str, the refined Python code generated by the model.
    """

    # Format the examples for the prompt
    formatted_examples = "\n\n".join(
        f"""
        Example {i+1}:
        User Question: {example['question']}
        Context: {example['context']}
        Model Output: {example['output']}
        """
        for i, example in enumerate(examples)
    )
    prompt = f"""
```

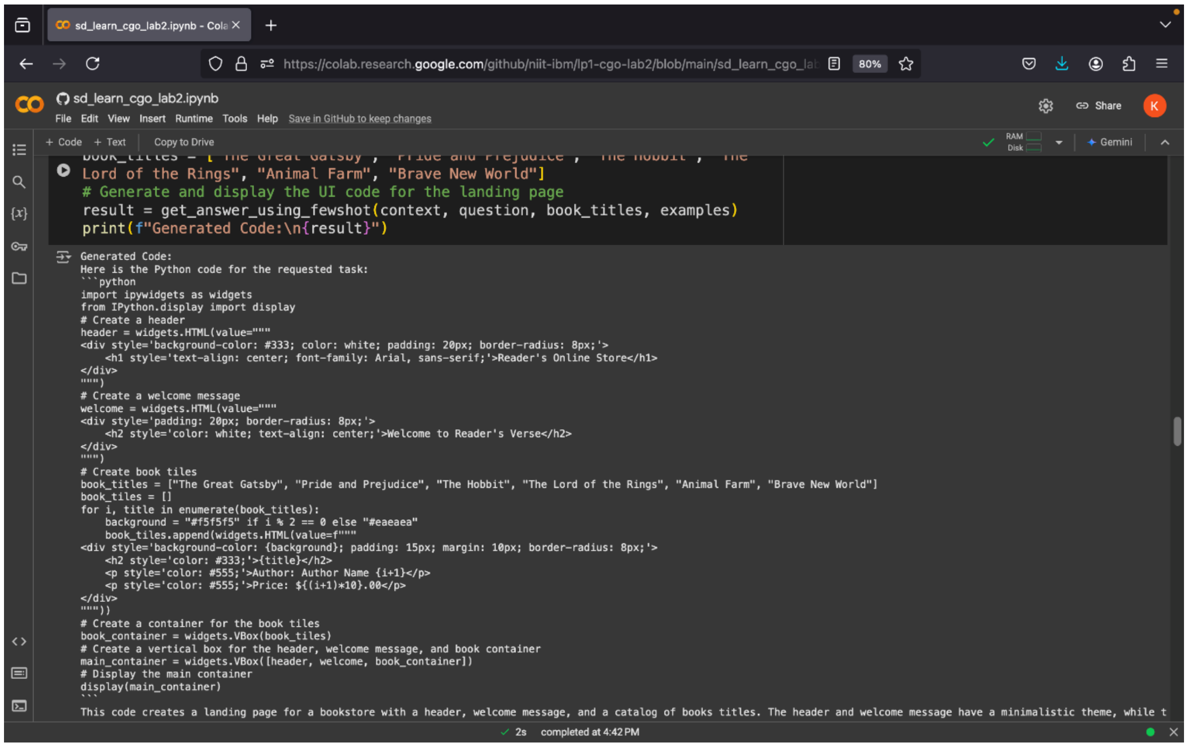
sd_learn_cgo_lab2.ipynb - Cola
https://colab.research.google.com/github/niit-ibm/lp1-cgo-lab2/blob/main/sd_learn_cgo_la
80%
sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help Save in GitHub to keep changes
Share
+ Code + Text Copy to Drive
RAM Disk
Gemini
2s completed at 4:42 PM

2. In the ninth cell of the notebook, select the **Play** button to run the **refine_output_with_fewshot** function. This function refines the initial output of the few-shot prompt for generating a Python code that adds shadow and hover effects to enhance the design of the landing page for the Reader's Verse website.



```
# Generate the Code
context = "Refined page with Hover Effects for Enhanced User Interaction"
question = "Enhance the book tiles in the catalog with hover effects that
change the background color, add a shadow, and slightly scale the tiles on
hover. Output should also retain the Header and Welcome message aswell in the
page"
# Generate and display the UI code for the landing page
ref_result = refine_output_with_fewshot(context, question, book_titles,
examples, result)
print(f"Generated Code:\n{ref_result}")
```

3. The generated output includes the refined Python code that will create the bookstore's landing page UI, including a title and a catalog of book titles with the design enhancements requested by the client.

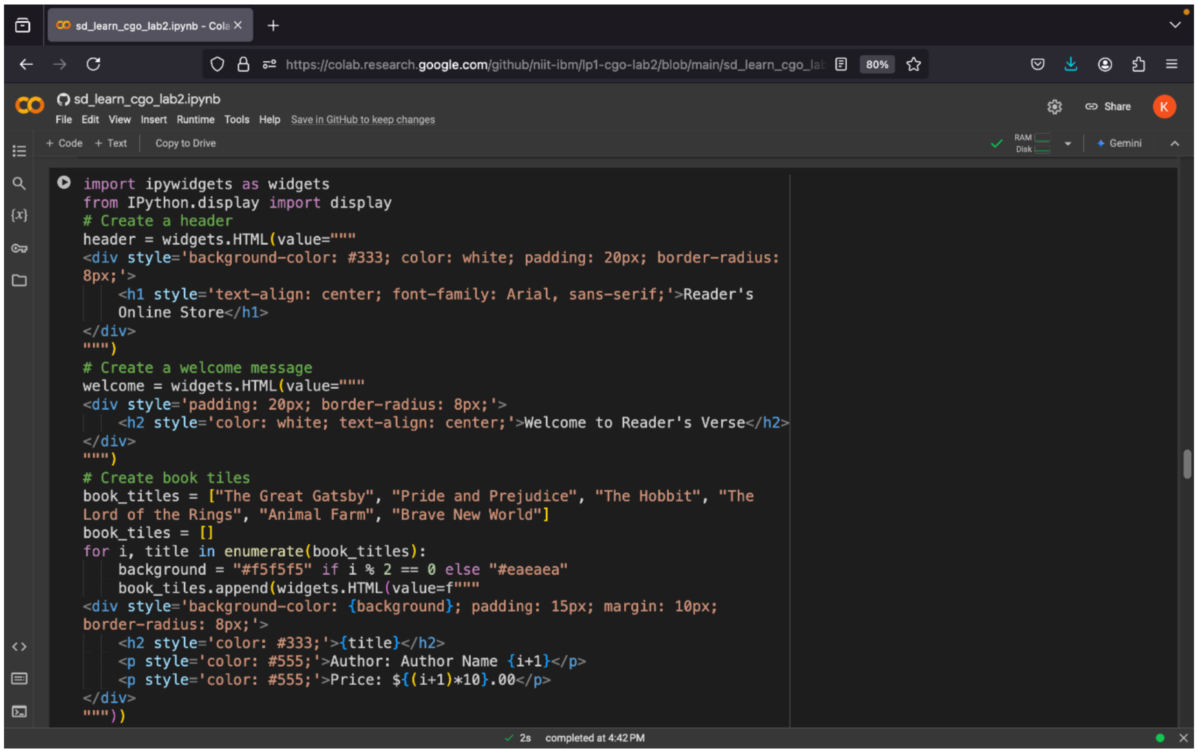


```
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The
Lord of the Rings", "Animal Farm", "Brave New World"]
# Generate and display the UI code for the landing page
result = get_answer_using_fewshot(context, question, book_titles, examples)
print(f"Generated Code:\n{result}")
```

~~~
Generated Code:
Here is the Python code for the requested task:
```python
import ipywidgets as widgets
from IPython.display import display
# Create a header
header = widgets.HTML(value="""
<div style='background-color: #333; color: white; padding: 20px; border-radius: 8px;'>
    <h1 style='text-align: center; font-family: Arial, sans-serif;'>Reader's Online Store</h1>
</div>
""")
# Create a welcome message
welcome = widgets.HTML(value="""
<div style='padding: 20px; border-radius: 8px;'>
    <h2 style='color: white; text-align: center;'>Welcome to Reader's Verse</h2>
</div>
""")
# Create book tiles
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The Lord of the Rings", "Animal Farm", "Brave New World"]
book_tiles = []
for i, title in enumerate(book_titles):
    background = "#f5f5f5" if i % 2 == 0 else "#eaeaea"
    book_tiles.append(widgets.HTML(value=f"""
<div style='background-color: {background}; padding: 15px; margin: 10px; border-radius: 8px;'>
    <h2 style='color: #333;'>{title}</h2>
    <p style='color: #555;'>Author: Author Name {i+1}</p>
    <p style='color: #555;'>Price: ${(i+1)*10}.00</p>
</div>
"""))
# Create a container for the book tiles
book_container = widgets.VBox(book_tiles)
# Create a vertical box for the header, welcome message, and book container
main_container = widgets.VBox([header, welcome, book_container])
# Display the main container
display(main_container)
```
This code creates a landing page for a bookstore with a header, welcome message, and a catalog of books titles. The header and welcome message have a minimalistic theme, while t
~~~

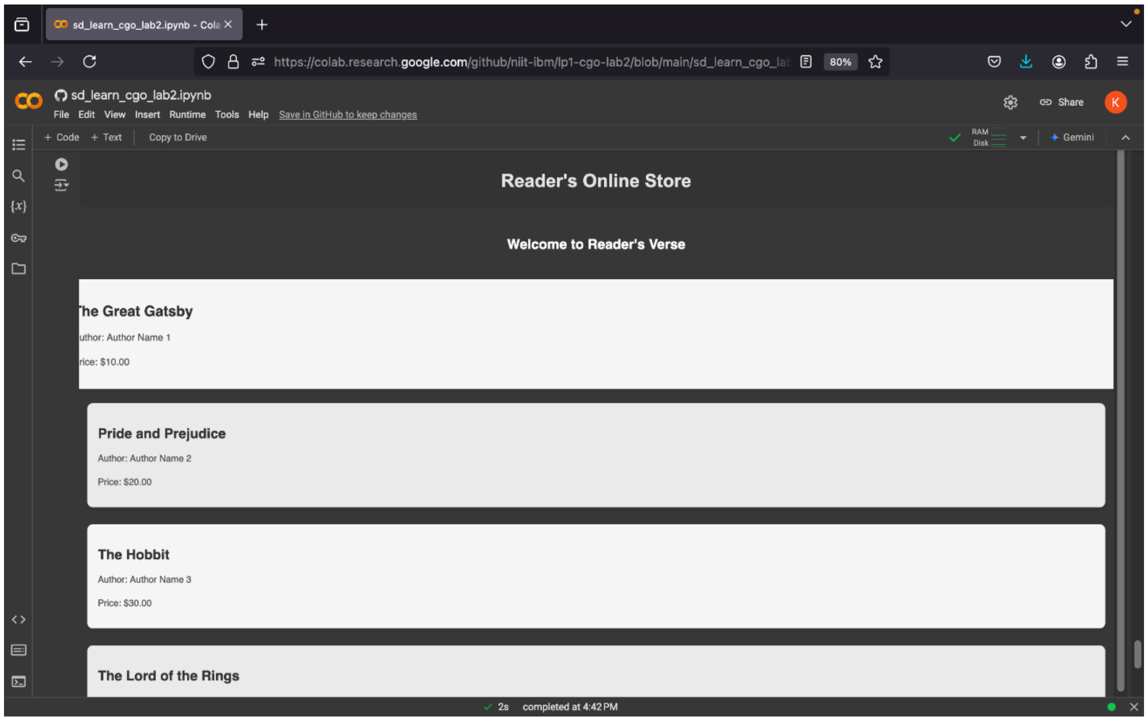
Note: When you generate the code, it might be different from this code output. These minor differences are normal and will not affect the overall functionality or outcome of the lab.

4. For ease of execution, the output code is pre-populated in a new cell of the Jupyter notebook. Select the **Play** button in the tenth cell of the notebook to validate the refined Python code output generated by the IBM Granite model for the given scenario.


sd_learn_cgo_lab2.ipynb - Cola
https://colab.research.google.com/github/niit-ibm/lp1-cgo-lab2/blob/main/sd_learn_cgo_lal
80%
sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help Save in GitHub to keep changes
Share
+ Code + Text Copy to Drive
RAM Disk
Gemini

```
import ipywidgets as widgets
from IPython.display import display
# Create a header
header = widgets.HTML(value="""
<div style='background-color: #333; color: white; padding: 20px; border-radius:
8px;'>
    <h1 style='text-align: center; font-family: Arial, sans-serif;'>Reader's
    Online Store</h1>
</div>
""")
# Create a welcome message
welcome = widgets.HTML(value="""
<div style='padding: 20px; border-radius: 8px;'>
    <h2 style='color: white; text-align: center;'>Welcome to Reader's Verse</h2>
</div>
""")
# Create book tiles
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The
Lord of the Rings", "Animal Farm", "Brave New World"]
book_tiles = []
for i, title in enumerate(book_titles):
    background = "#f5f5f5" if i % 2 == 0 else "#eaeaea"
    book_tiles.append(widgets.HTML(value=f"""
<div style='background-color: {background}; padding: 15px; margin: 10px;
border-radius: 8px;'>
    <h2 style='color: #333;'>{title}</h2>
    <p style='color: #555;'>Author: Author Name {i+1}</p>
    <p style='color: #555;'>Price: ${(i+1)*10}.00</p>
</div>
"""))
```

5. The output should display the bookstore's landing page UI following the code cell. You'll observe the following refinements in the design of the landing page as compared to the initial output of the few-shot prompt:
   - Hover effects for book tiles
   - Shadow effects and smooth transitions
   - Consistent formatting


sd_learn_cgo_lab2.ipynb - Cola
https://colab.research.google.com/github/niit-ibm/lp1-cgo-lab2/blob/main/sd_learn_cgo_la
80%
sd_learn_cgo_lab2.ipynb
File Edit View Insert Runtime Tools Help Save in GitHub to keep changes
Share
+ Code + Text Copy to Drive
RAM Disk
Gemini
2s completed at 4:42 PM

Reader's Online Store

Welcome to Reader's Verse

The Great Gatsby
Author: Author Name 1
Price: $10.00

Pride and Prejudice
Author: Author Name 2
Price: $20.00

The Hobbit
Author: Author Name 3
Price: $30.00

The Lord of the Rings

Conclusion

You used an IBM Granite model to optimize code for creating the landing page of a website for Reader’s Verse, a hypothetical online bookstore. The client is delighted to see how quickly you created a working prototype of the UI for the landing page of the website. You used few-shot prompts to generate Python code output from the IBM Granite model and validated the AI-generated code to ensure that it correctly renders the expected output.