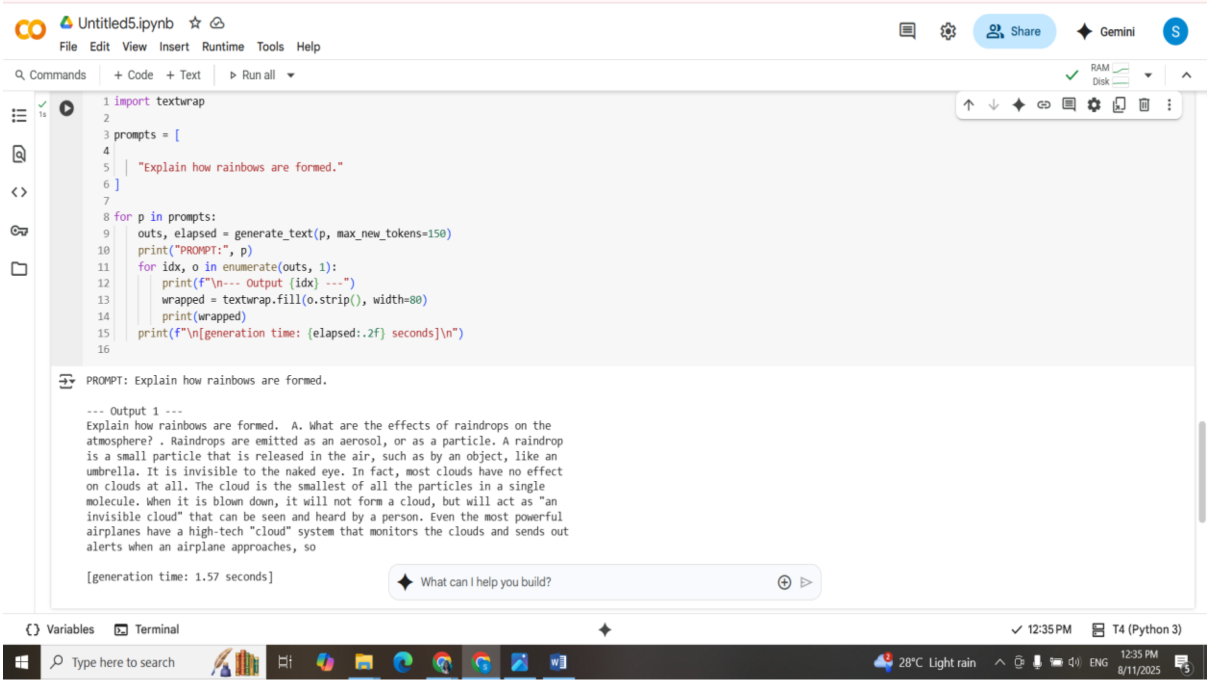
Assignment from Session 3 and session 4 (Basics of Gen AI and Basics of RAG)

1. Write a Python script to send a prompt "`Explain how rainbows are formed`" using OpenAI's GPT-3.5 Turbo or Hugging Face Transformers and print the response.

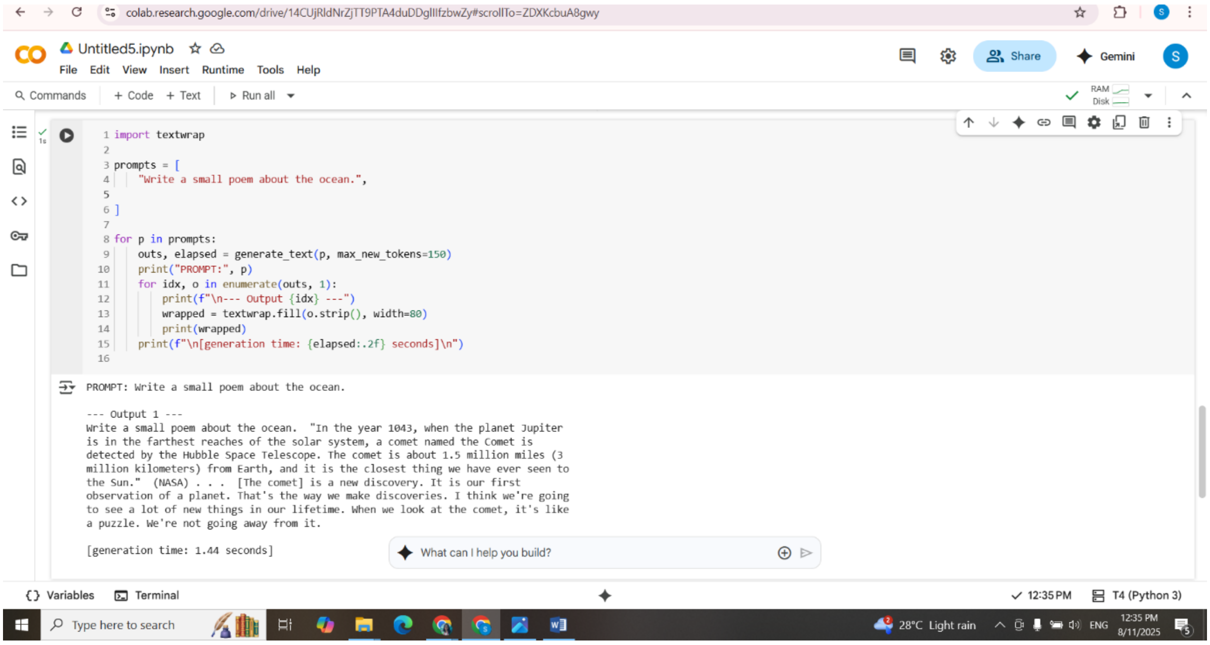


Untitled5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini
Variables Terminal
12:35 PM T4 (Python 3)
What can I help you build?
Type here to search
28°C Light rain ENG 12:35 PM 8/11/2025

```python
import textwrap

prompts = [

    "Explain how rainbows are formed."
]

for p in prompts:
    outs, elapsed = generate_text(p, max_new_tokens=150)
    print("PROMPT:", p)
    for idx, o in enumerate(outs, 1):
        print(f"\n--- Output {idx} ---")
        wrapped = textwrap.fill(o.strip(), width=80)
        print(wrapped)
    print(f"\n[generation time: {elapsed:.2f} seconds]\n")
```

```
PROMPT: Explain how rainbows are formed.

--- Output 1 ---
Explain how rainbows are formed.  A. What are the effects of raindrops on the
atmosphere? . Raindrops are emitted as an aerosol, or as a particle. A raindrop
is a small particle that is released in the air, such as by an object, like an
umbrella. It is invisible to the naked eye. In fact, most clouds have no effect
on clouds at all. The cloud is the smallest of all the particles in a single
molecule. When it is blown down, it will not form a cloud, but will act as "an
invisible cloud" that can be seen and heard by a person. Even the most powerful
airplanes have a high-tech "cloud" system that monitors the clouds and sends out
alerts when an airplane approaches, so

[generation time: 1.57 seconds]
```

2. Use the `transformers` library from Hugging Face to load a model and generate text from a prompt:
"Write a small poem about the ocean". Print the result.

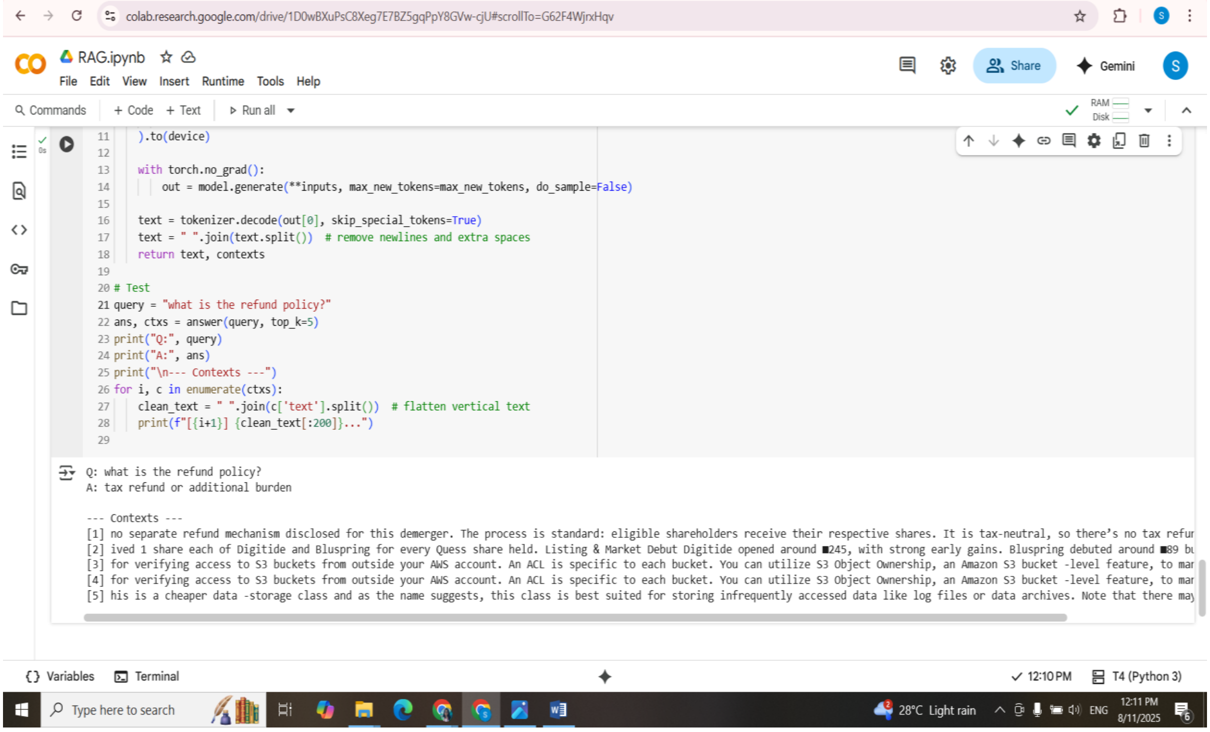


colab.research.google.com/drive/14CUjRldNrZjTT9PTA4duDDglllfzbwZy#scrollTo=ZDXKcbuA8gwy
Untitled5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini
Variables Terminal
12:35 PM T4 (Python 3)
What can I help you build?
Type here to search
28°C Light rain ENG 12:35 PM 8/11/2025

```python
import textwrap

prompts = [
    "Write a small poem about the ocean.",

]

for p in prompts:
    outs, elapsed = generate_text(p, max_new_tokens=150)
    print("PROMPT:", p)
    for idx, o in enumerate(outs, 1):
        print(f"\n--- Output {idx} ---")
        wrapped = textwrap.fill(o.strip(), width=80)
        print(wrapped)
    print(f"\n[generation time: {elapsed:.2f} seconds]\n")
```

```
PROMPT: Write a small poem about the ocean.

--- Output 1 ---
Write a small poem about the ocean.  "In the year 1043, when the planet Jupiter
is in the farthest reaches of the solar system, a comet named the Comet is
detected by the Hubble Space Telescope. The comet is about 1.5 million miles (3
million kilometers) from Earth, and it is the closest thing we have ever seen to
the Sun."  (NASA) . . .  [The comet] is a new discovery. It is our first
observation of a planet. That's the way we make discoveries. I think we're going
to see a lot of new things in our lifetime. When we look at the comet, it's like
a puzzle. We're not going away from it.

[generation time: 1.44 seconds]
```

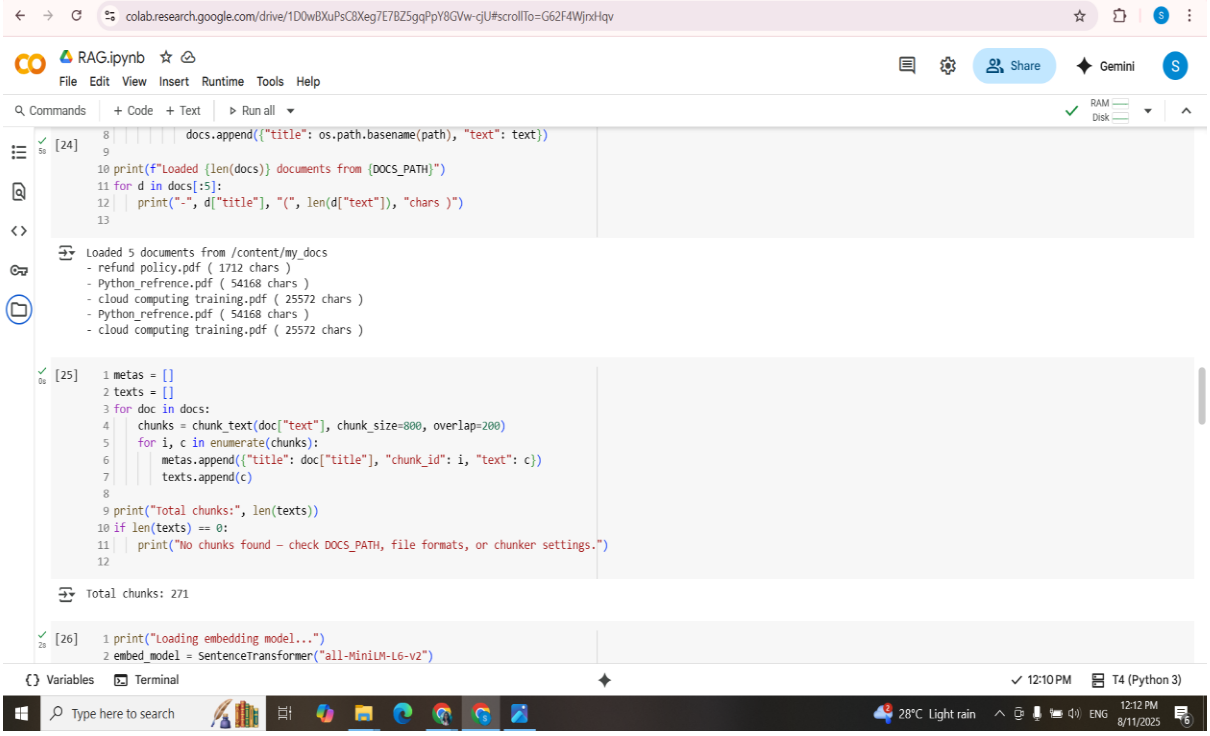
3. Using `RetrievalQA`, create a pipeline that can answer the question: “What is the refund policy?”



```python
    ).to(device)

    with torch.no_grad():
        out = model.generate(**inputs, max_new_tokens=max_new_tokens, do_sample=False)

    text = tokenizer.decode(out[0], skip_special_tokens=True)
    text = " ".join(text.split())  # remove newlines and extra spaces
    return text, contexts

# Test
query = "what is the refund policy?"
ans, ctxs = answer(query, top_k=5)
print("Q:", query)
print("A:", ans)
print("\n--- Contexts ---")
for i, c in enumerate(ctxs):
    clean_text = " ".join(c['text'].split())  # flatten vertical text
    print(f"[{i+1}] {clean_text[:200]}...")

```

```
Q: what is the refund policy?
A: tax refund or additional burden

--- Contexts ---
[1] no separate refund mechanism disclosed for this demerger. The process is standard: eligible shareholders receive their respective shares. It is tax-neutral, so there’s no tax refur
[2] ived 1 share each of Digitide and Bluspring for every Quess share held. Listing & Market Debut Digitide opened around ■245, with strong early gains. Bluspring debuted around ■89 bu
[3] for verifying access to S3 buckets from outside your AWS account. An ACL is specific to each bucket. You can utilize S3 Object Ownership, an Amazon S3 bucket -level feature, to mar
[4] for verifying access to S3 buckets from outside your AWS account. An ACL is specific to each bucket. You can utilize S3 Object Ownership, an Amazon S3 bucket -level feature, to mar
[5] his is a cheaper data -storage class and as the name suggests, this class is best suited for storing infrequently accessed data like log files or data archives. Note that there may
```

4. Load a `.txt` or `.pdf` file using LangChain’s `TextLoader` and split it using `RecursiveCharacterTextSplitter`. Print the total number of document chunks created.



[24]

```python
                docs.append({"title": os.path.basename(path), "text": text})

print(f"Loaded {len(docs)} documents from {DOCS_PATH}")
for d in docs[:5]:
    print("-", d["title"], "(", len(d["text"]), "chars )")

```

```
Loaded 5 documents from /content/my_docs
- refund policy.pdf ( 1712 chars )
- Python_refrence.pdf ( 54168 chars )
- cloud computing training.pdf ( 25572 chars )
- Python_refrence.pdf ( 54168 chars )
- cloud computing training.pdf ( 25572 chars )
```

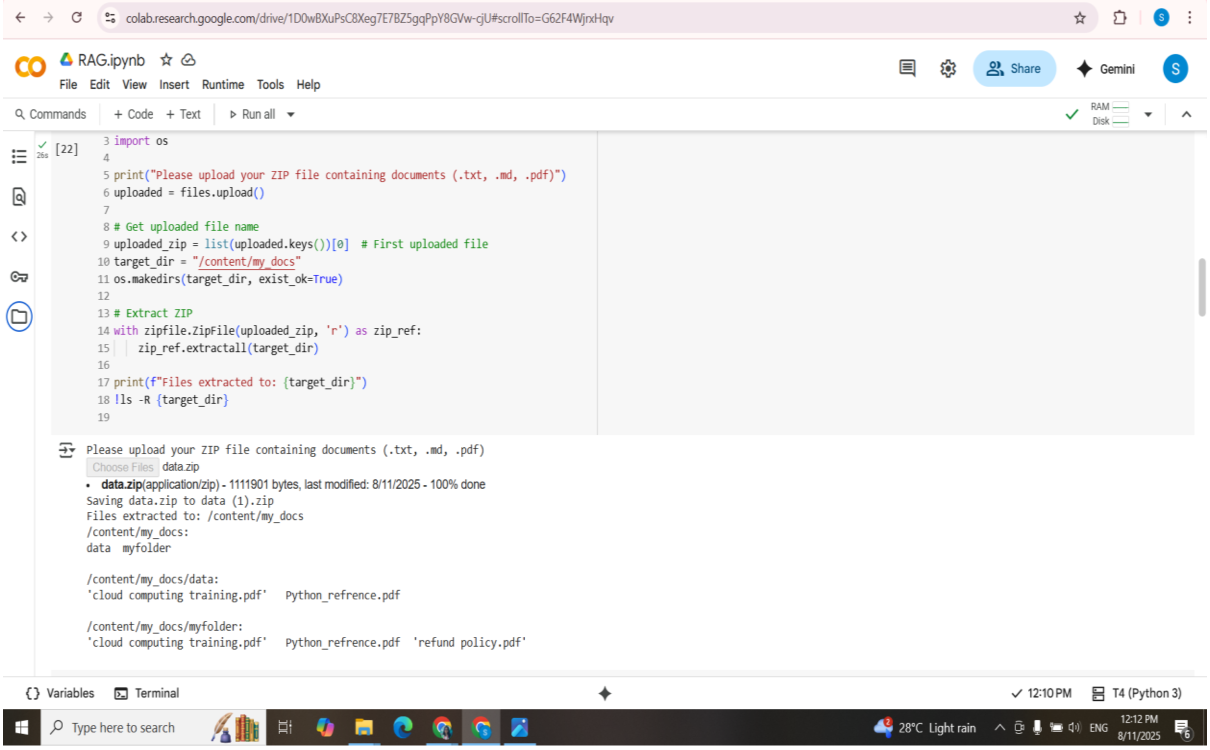
[25]

```python
metas = []
texts = []
for doc in docs:
    chunks = chunk_text(doc["text"], chunk_size=800, overlap=200)
    for i, c in enumerate(chunks):
        metas.append({"title": doc["title"], "chunk_id": i, "text": c})
        texts.append(c)

print("Total chunks:", len(texts))
if len(texts) == 0:
    print("No chunks found — check DOCS_PATH, file formats, or chunker settings.")

```

```
Total chunks: 271
```

[26]

```python
print("Loading embedding model...")
embed_model = SentenceTransformer("all-MiniLM-L6-v2")
```



```
import os

print("Please upload your ZIP file containing documents (.txt, .md, .pdf)")
uploaded = files.upload()

# Get uploaded file name
uploaded_zip = list(uploaded.keys())[0]  # First uploaded file
target_dir = "/content/my_docs"
os.makedirs(target_dir, exist_ok=True)

# Extract ZIP
with zipfile.ZipFile(uploaded_zip, 'r') as zip_ref:
    zip_ref.extractall(target_dir)

print(f"Files extracted to: {target_dir}")
!ls -R {target_dir}

```

```
Please upload your ZIP file containing documents (.txt, .md, .pdf)
Choose Files data.zip
• data.zip(application/zip) - 1111901 bytes, last modified: 8/11/2025 - 100% done
Saving data.zip to data (1).zip
Files extracted to: /content/my_docs
/content/my_docs:
data  myfolder

/content/my_docs/data:
'cloud computing training.pdf'   Python_refrence.pdf

/content/my_docs/myfolder:
'cloud computing training.pdf'   Python_refrence.pdf  'refund policy.pdf'
```