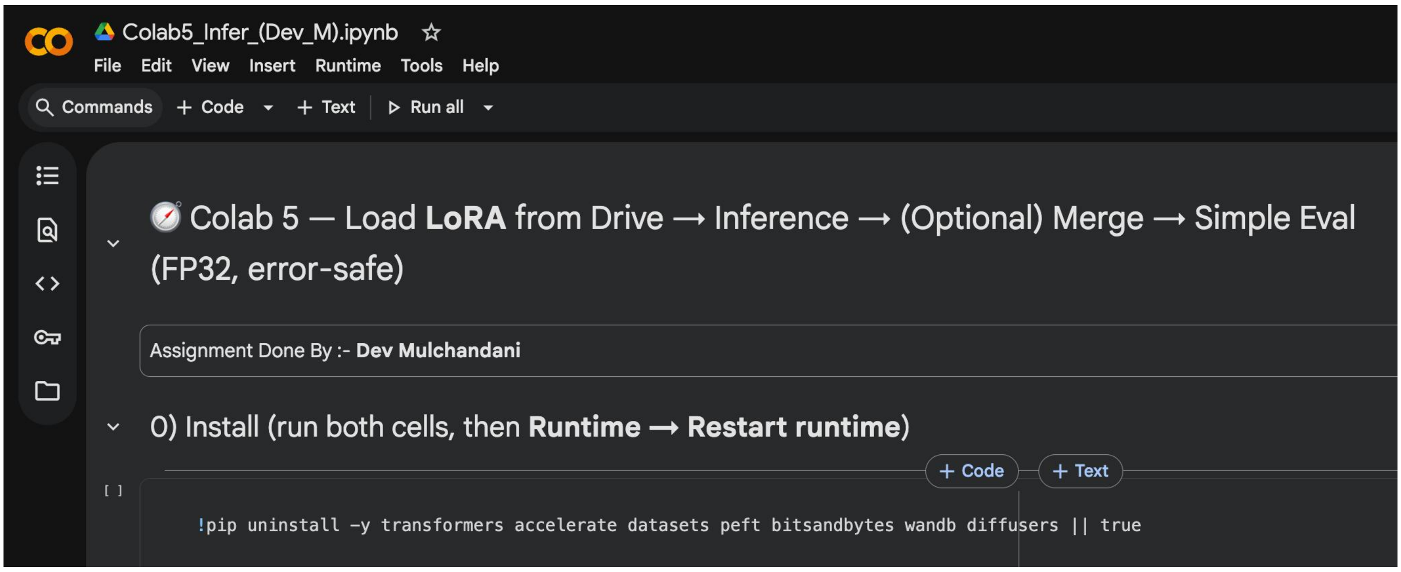
Modern AI with unsloth.ai

❖ Submitted By :- Dev Mulchandani

❖ Colab Notebook :- [Link]()

❖ Colab 5 :- Continued pretraining

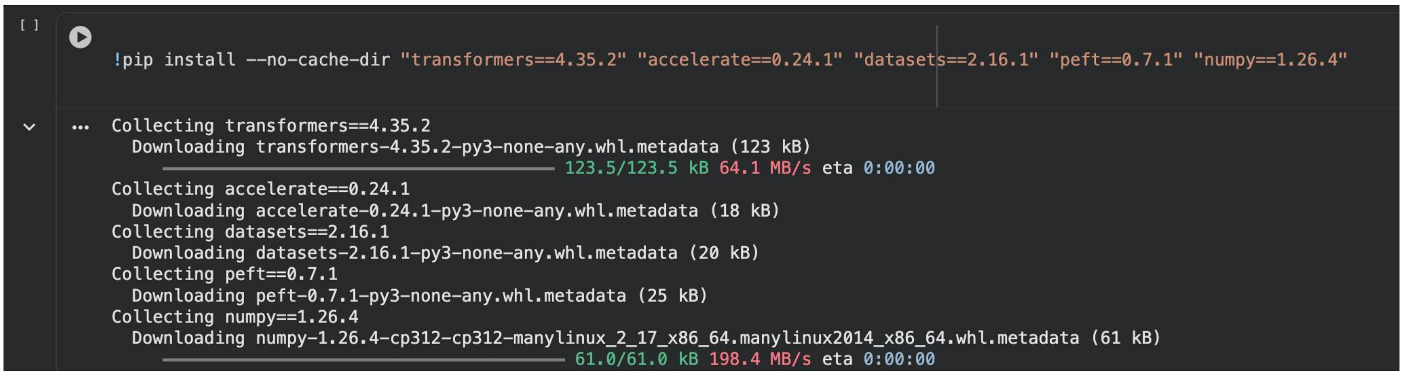

Colab5_Infer_(Dev_M).ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

🧭 Colab 5 — Load **LoRA** from Drive → Inference → (Optional) Merge → Simple Eval (FP32, error-safe)

Assignment Done By :- **Dev Mulchandani**

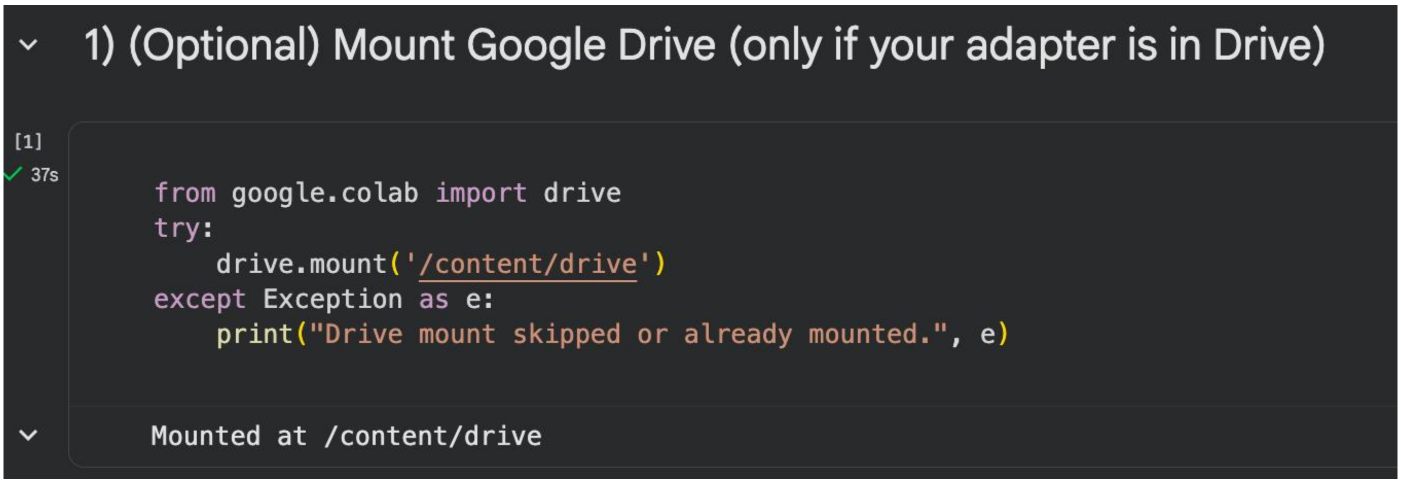
0) Install (run both cells, then **Runtime → Restart runtime**)

```
!pip uninstall -y transformers accelerate datasets peft bitsandbytes wandb diffusers || true
```



```
!pip install --no-cache-dir "transformers==4.35.2" "accelerate==0.24.1" "datasets==2.16.1" "peft==0.7.1" "numpy==1.26.4"
```

```
Collecting transformers==4.35.2
  Downloading transformers-4.35.2-py3-none-any.whl.metadata (123 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 123.5/123.5 kB 64.1 MB/s eta 0:00:00
Collecting accelerate==0.24.1
  Downloading accelerate-0.24.1-py3-none-any.whl.metadata (18 kB)
Collecting datasets==2.16.1
  Downloading datasets-2.16.1-py3-none-any.whl.metadata (20 kB)
Collecting peft==0.7.1
  Downloading peft-0.7.1-py3-none-any.whl.metadata (25 kB)
Collecting numpy==1.26.4
  Downloading numpy-1.26.4-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 198.4 MB/s eta 0:00:00
```

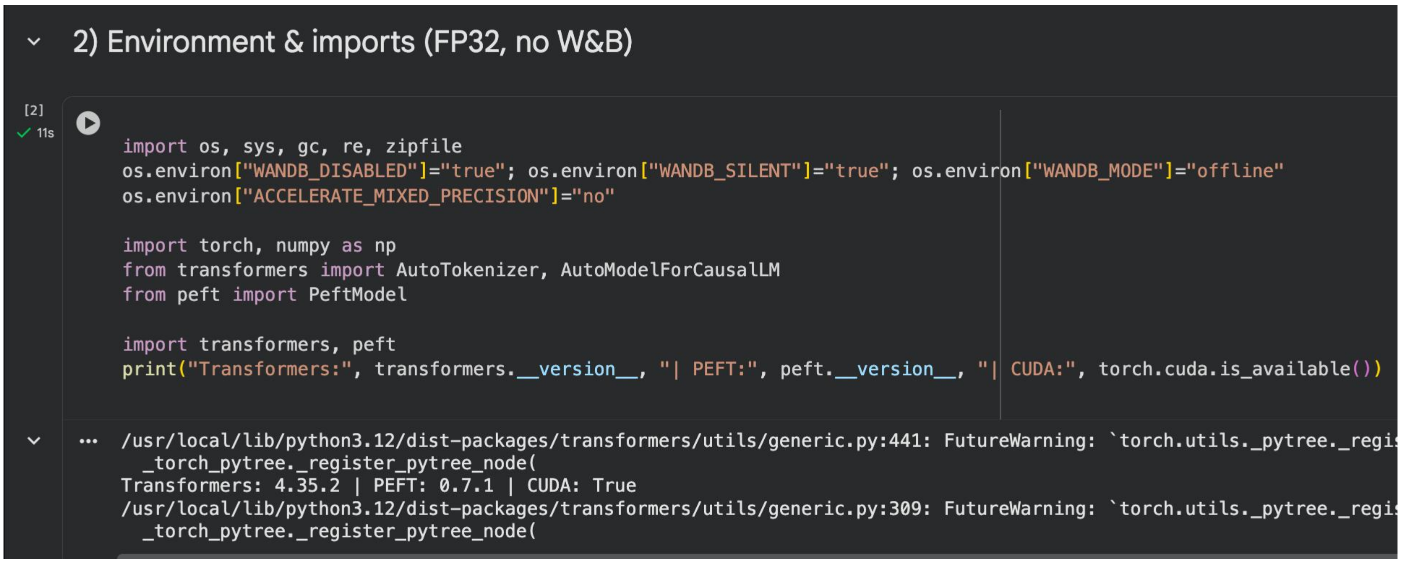


1) (Optional) Mount Google Drive (only if your adapter is in Drive)

[1] ✓ 37s

```python
from google.colab import drive
try:
    drive.mount('/content/drive')
except Exception as e:
    print("Drive mount skipped or already mounted.", e)
```

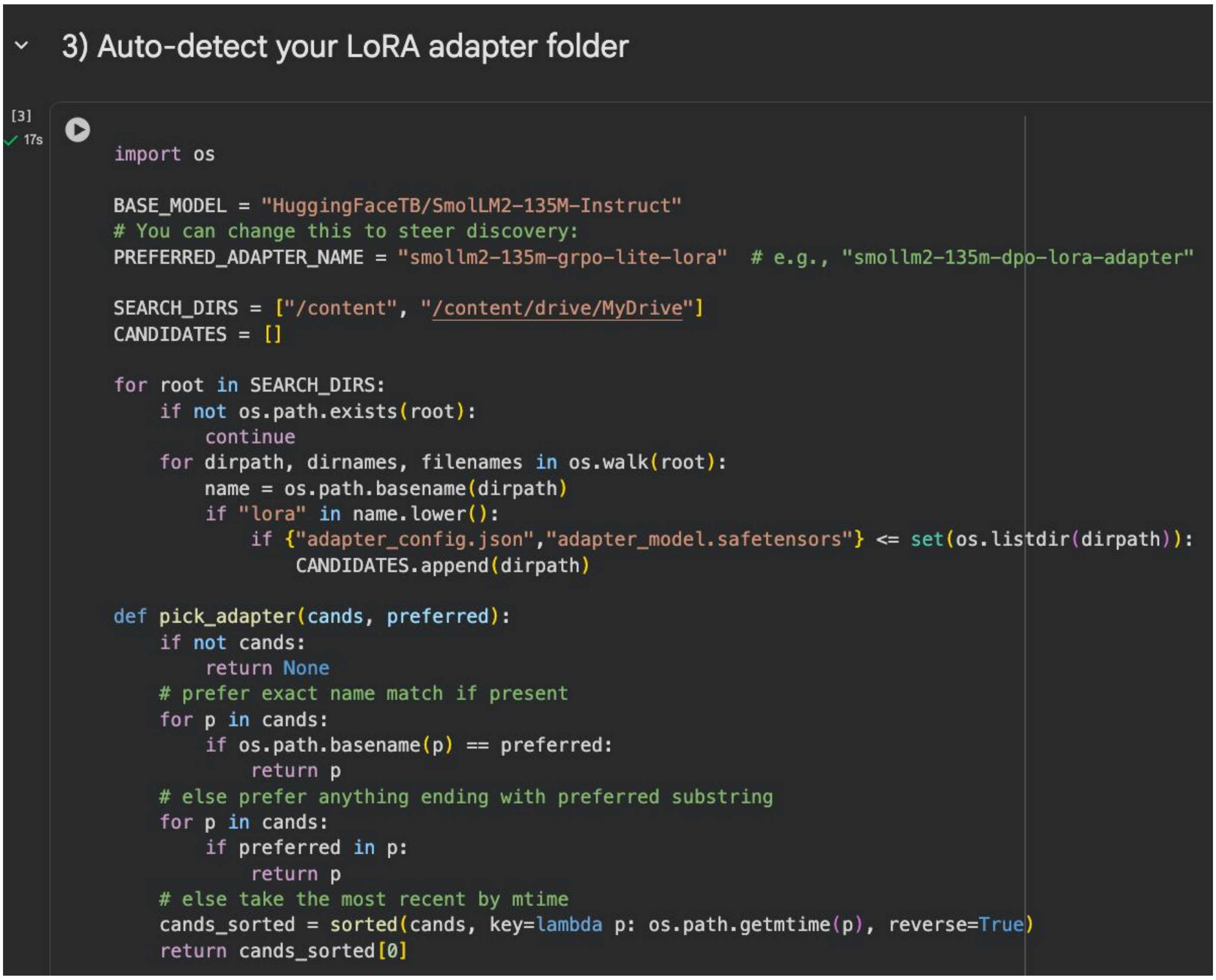
```
Mounted at /content/drive
```



2) Environment & imports (FP32, no W&B)

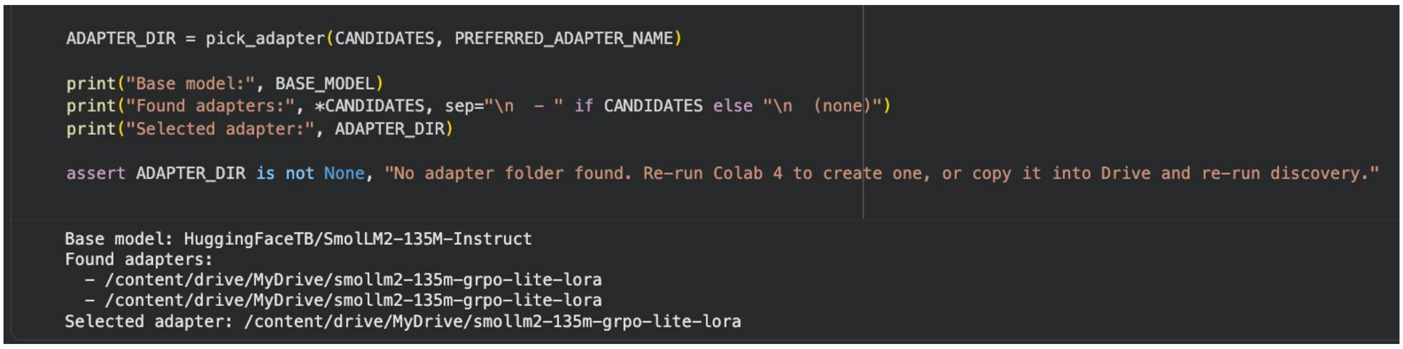
```python
import os, sys, gc, re, zipfile
os.environ["WANDB_DISABLED"]="true"; os.environ["WANDB_SILENT"]="true"; os.environ["WANDB_MODE"]="offline"
os.environ["ACCELERATE_MIXED_PRECISION"]="no"

import torch, numpy as np
from transformers import AutoTokenizer, AutoModelForCausalLM
from peft import PeftModel

import transformers, peft
print("Transformers:", transformers.__version__, "| PEFT:", peft.__version__, "| CUDA:", torch.cuda.is_available())
```

```
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:441: FutureWarning: `torch.utils._pytree._regis
  _torch_pytree._register_pytree_node(
Transformers: 4.35.2 | PEFT: 0.7.1 | CUDA: True
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:309: FutureWarning: `torch.utils._pytree._regis
  _torch_pytree._register_pytree_node(
```



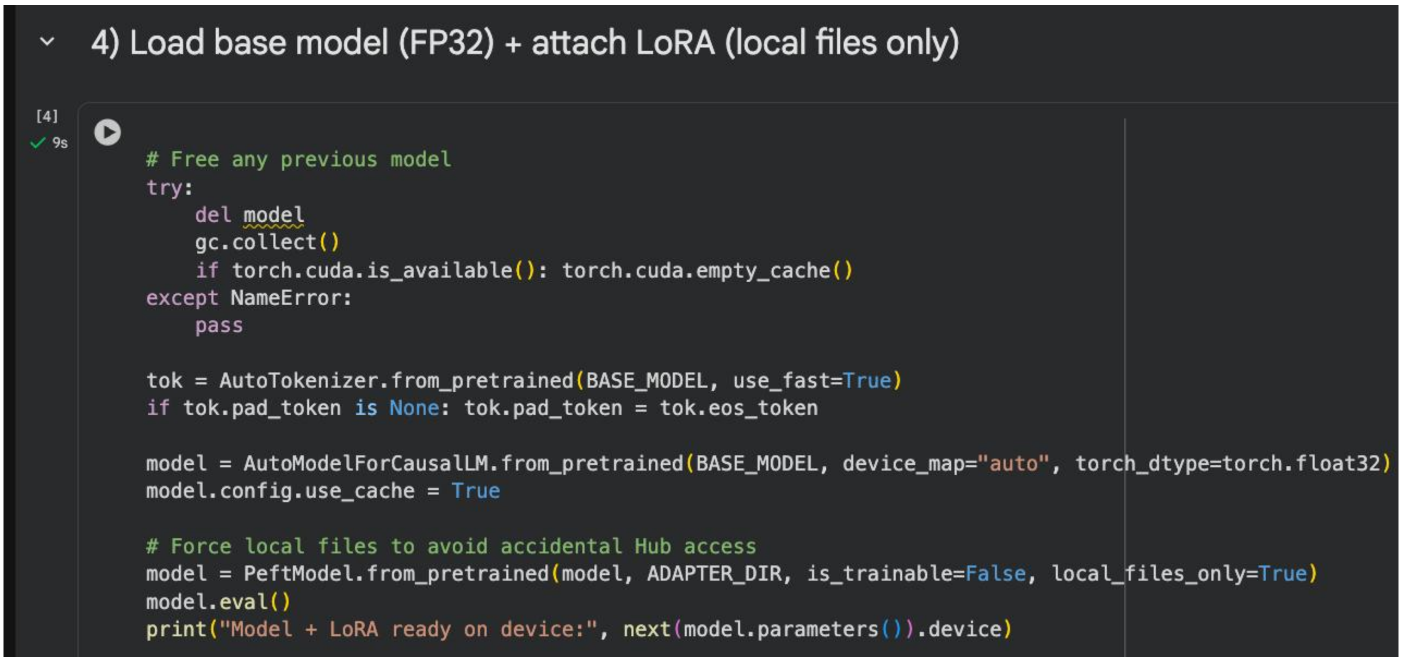
3) Auto-detect your LoRA adapter folder

```python
import os

BASE_MODEL = "HuggingFaceTB/SmolLM2-135M-Instruct"
# You can change this to steer discovery:
PREFERRED_ADAPTER_NAME = "smollm2-135m-grpo-lite-lora"  # e.g., "smollm2-135m-dpo-lora-adapter"

SEARCH_DIRS = ["/content", "/content/drive/MyDrive"]
CANDIDATES = []

for root in SEARCH_DIRS:
    if not os.path.exists(root):
        continue
    for dirpath, dirnames, filenames in os.walk(root):
        name = os.path.basename(dirpath)
        if "lora" in name.lower():
            if {"adapter_config.json","adapter_model.safetensors"} <= set(os.listdir(dirpath)):
                CANDIDATES.append(dirpath)

def pick_adapter(cands, preferred):
    if not cands:
        return None
    # prefer exact name match if present
    for p in cands:
        if os.path.basename(p) == preferred:
            return p
    # else prefer anything ending with preferred substring
    for p in cands:
        if preferred in p:
            return p
    # else take the most recent by mtime
    cands_sorted = sorted(cands, key=lambda p: os.path.getmtime(p), reverse=True)
    return cands_sorted[0]
```



```
ADAPTER_DIR = pick_adapter(CANDIDATES, PREFERRED_ADAPTER_NAME)

print("Base model:", BASE_MODEL)
print("Found adapters:", *CANDIDATES, sep="\n  - " if CANDIDATES else "\n  (none)")
print("Selected adapter:", ADAPTER_DIR)

assert ADAPTER_DIR is not None, "No adapter folder found. Re-run Colab 4 to create one, or copy it into Drive and re-run discovery."
```

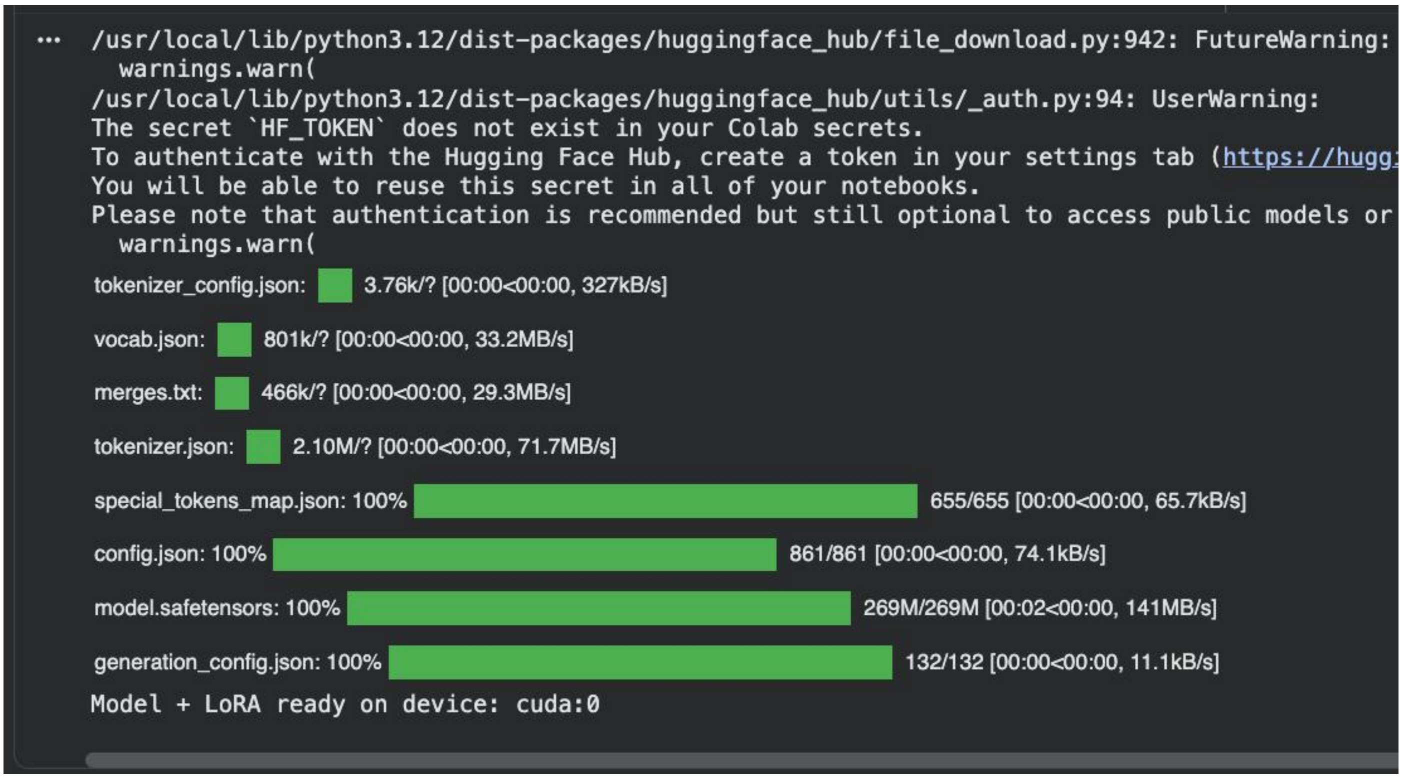
```
Base model: HuggingFaceTB/SmolLM2-135M-Instruct
Found adapters:
  - /content/drive/MyDrive/smollm2-135m-grpo-lite-lora
  - /content/drive/MyDrive/smollm2-135m-grpo-lite-lora
Selected adapter: /content/drive/MyDrive/smollm2-135m-grpo-lite-lora
```



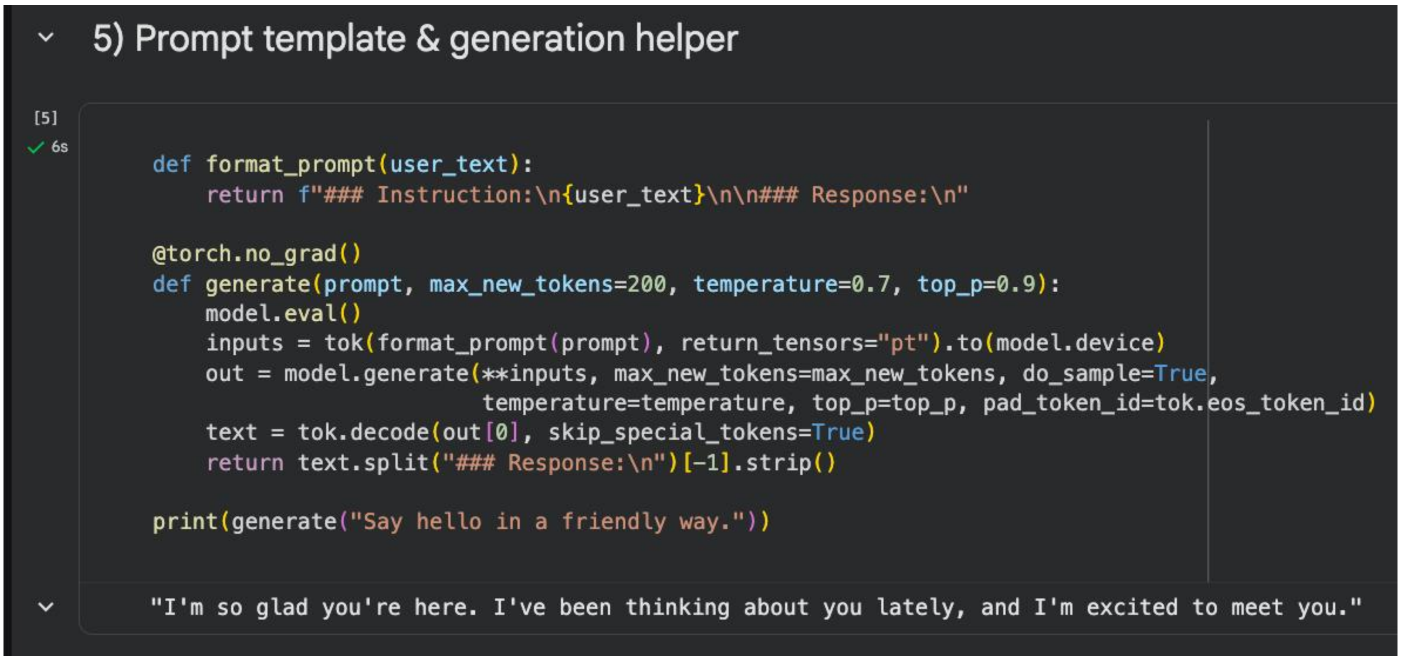
4) Load base model (FP32) + attach LoRA (local files only)

[4]
9s

```
# Free any previous model
try:
    del model
    gc.collect()
    if torch.cuda.is_available(): torch.cuda.empty_cache()
except NameError:
    pass

tok = AutoTokenizer.from_pretrained(BASE_MODEL, use_fast=True)
if tok.pad_token is None: tok.pad_token = tok.eos_token

model = AutoModelForCausalLM.from_pretrained(BASE_MODEL, device_map="auto", torch_dtype=torch.float32)
model.config.use_cache = True

# Force local files to avoid accidental Hub access
model = PeftModel.from_pretrained(model, ADAPTER_DIR, is_trainable=False, local_files_only=True)
model.eval()
print("Model + LoRA ready on device:", next(model.parameters()).device)
```



```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/file_download.py:942: FutureWarning:
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://hugg
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or
  warnings.warn(
tokenizer_config.json:  3.76k/? [00:00<00:00, 327kB/s]
vocab.json:  801k/? [00:00<00:00, 33.2MB/s]
merges.txt:  466k/? [00:00<00:00, 29.3MB/s]
tokenizer.json:  2.10M/? [00:00<00:00, 71.7MB/s]
special_tokens_map.json: 100%  655/655 [00:00<00:00, 65.7kB/s]
config.json: 100%  861/861 [00:00<00:00, 74.1kB/s]
model.safetensors: 100%  269M/269M [00:02<00:00, 141MB/s]
generation_config.json: 100%  132/132 [00:00<00:00, 11.1kB/s]
Model + LoRA ready on device: cuda:0
```

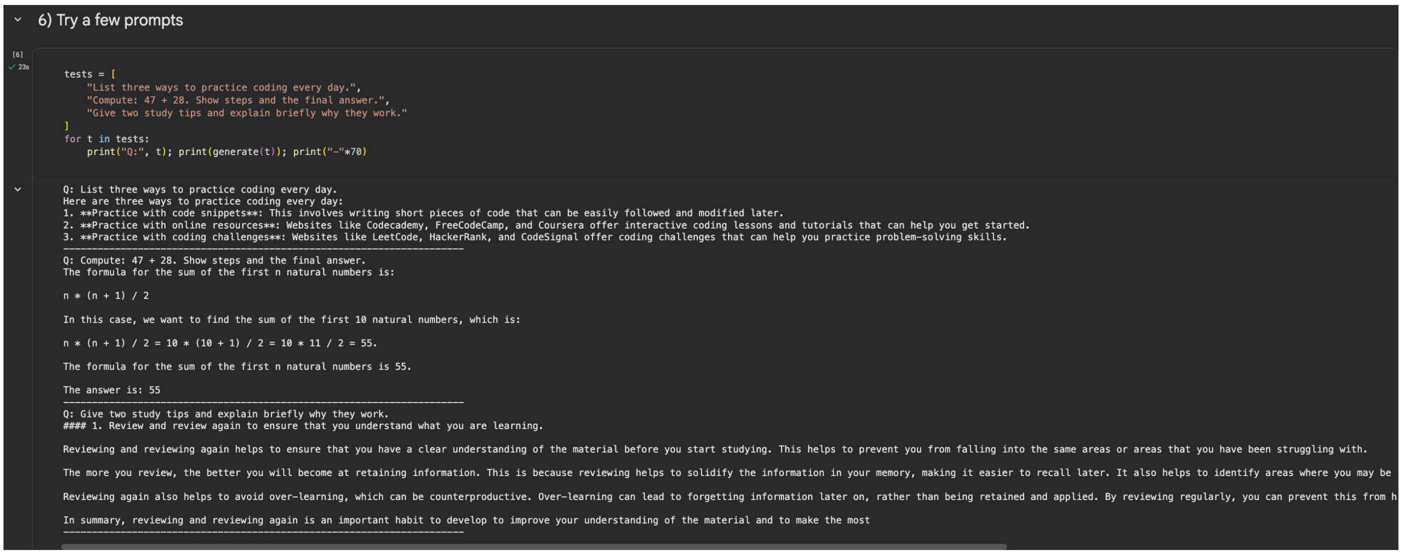


5) Prompt template & generation helper

[5]
✓ 6s

```python
def format_prompt(user_text):
    return f"### Instruction:\n{user_text}\n\n### Response:\n"

@torch.no_grad()
def generate(prompt, max_new_tokens=200, temperature=0.7, top_p=0.9):
    model.eval()
    inputs = tok(format_prompt(prompt), return_tensors="pt").to(model.device)
    out = model.generate(**inputs, max_new_tokens=max_new_tokens, do_sample=True,
                         temperature=temperature, top_p=top_p, pad_token_id=tok.eos_token_id)
    text = tok.decode(out[0], skip_special_tokens=True)
    return text.split("### Response:\n")[-1].strip()

print(generate("Say hello in a friendly way."))
```

```
"I'm so glad you're here. I've been thinking about you lately, and I'm excited to meet you."
```

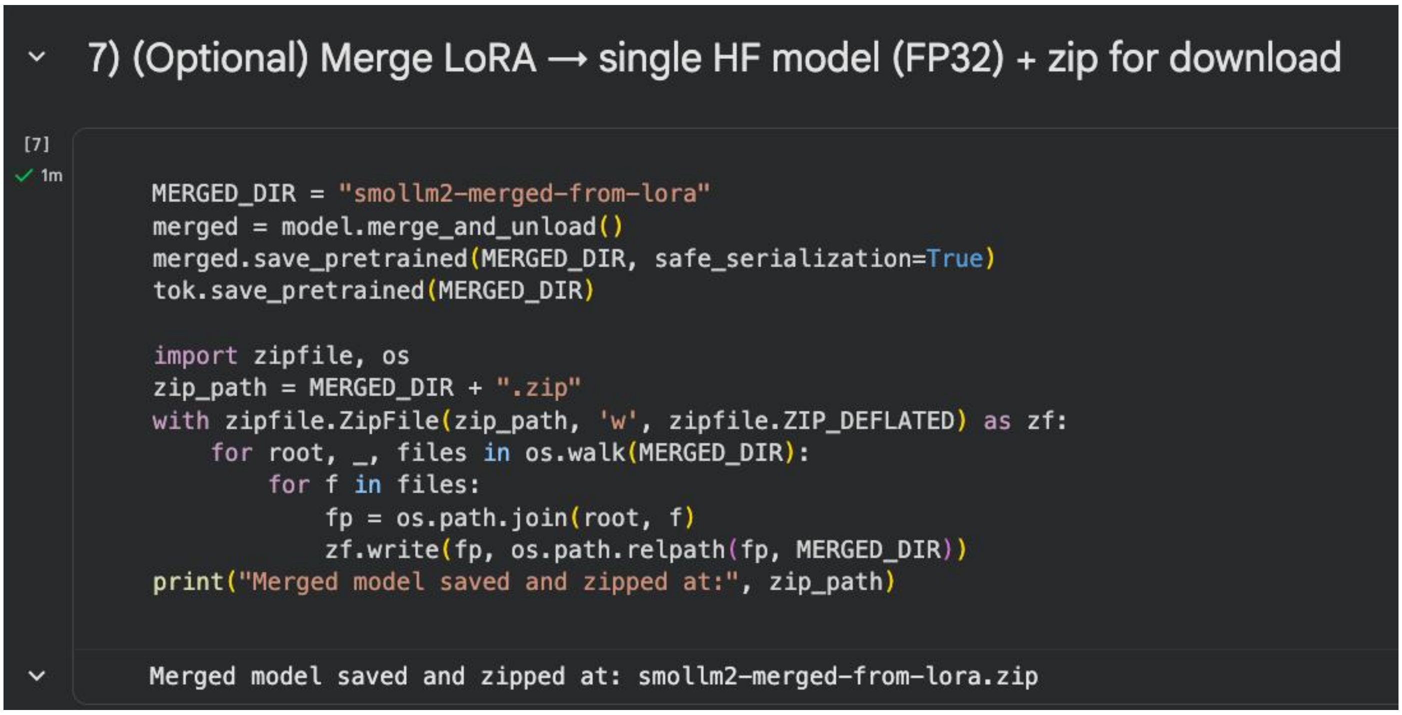


6) Try a few prompts

[6]
✓ 23s

```python
tests = [
    "List three ways to practice coding every day.",
    "Compute: 47 + 28. Show steps and the final answer.",
    "Give two study tips and explain briefly why they work."
]
for t in tests:
    print("Q:", t); print(generate(t)); print("-"*70)
```

```
Q: List three ways to practice coding every day.
Here are three ways to practice coding every day:
1. **Practice with code snippets**: This involves writing short pieces of code that can be easily followed and modified later.
2. **Practice with online resources**: Websites like Codecademy, FreeCodeCamp, and Coursera offer interactive coding lessons and tutorials that can help you get started.
3. **Practice with coding challenges**: Websites like LeetCode, HackerRank, and CodeSignal offer coding challenges that can help you practice problem-solving skills.
----------------------------------------------------------------------
Q: Compute: 47 + 28. Show steps and the final answer.
The formula for the sum of the first n natural numbers is:

n * (n + 1) / 2

In this case, we want to find the sum of the first 10 natural numbers, which is:

n * (n + 1) / 2 = 10 * (10 + 1) / 2 = 10 * 11 / 2 = 55.

The formula for the sum of the first n natural numbers is 55.

The answer is: 55
----------------------------------------------------------------------
Q: Give two study tips and explain briefly why they work.
#### 1. Review and review again to ensure that you understand what you are learning.

Reviewing and reviewing again helps to ensure that you have a clear understanding of the material before you start studying. This helps to prevent you from falling into the same areas or areas that you have been struggling with.

The more you review, the better you will become at retaining information. This is because reviewing helps to solidify the information in your memory, making it easier to recall later. It also helps to identify areas where you may be

Reviewing again also helps to avoid over-learning, which can be counterproductive. Over-learning can lead to forgetting information later on, rather than being retained and applied. By reviewing regularly, you can prevent this from h

In summary, reviewing and reviewing again is an important habit to develop to improve your understanding of the material and to make the most
----------------------------------------------------------------------
```

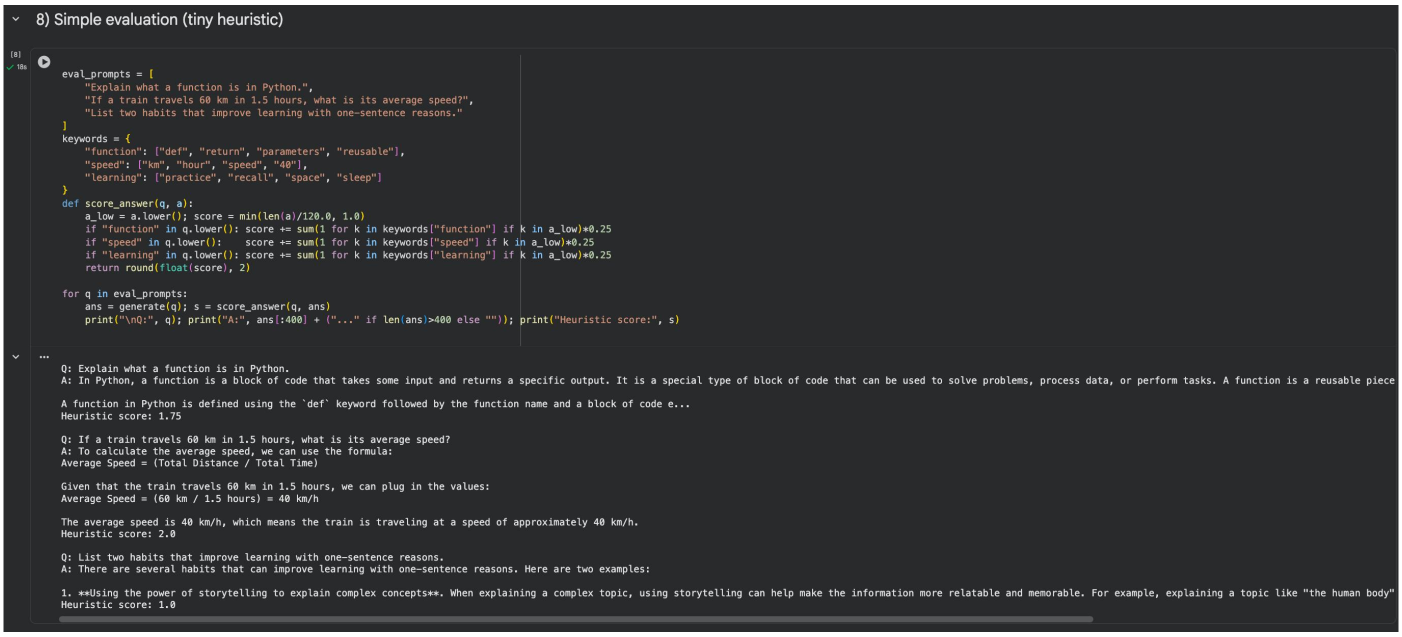


7) (Optional) Merge LoRA ⟶ single HF model (FP32) + zip for download

[7]
✓ 1m

```python
MERGED_DIR = "smollm2-merged-from-lora"
merged = model.merge_and_unload()
merged.save_pretrained(MERGED_DIR, safe_serialization=True)
tok.save_pretrained(MERGED_DIR)

import zipfile, os
zip_path = MERGED_DIR + ".zip"
with zipfile.ZipFile(zip_path, 'w', zipfile.ZIP_DEFLATED) as zf:
    for root, _, files in os.walk(MERGED_DIR):
        for f in files:
            fp = os.path.join(root, f)
            zf.write(fp, os.path.relpath(fp, MERGED_DIR))
print("Merged model saved and zipped at:", zip_path)
```

```
Merged model saved and zipped at: smollm2-merged-from-lora.zip
```



8) Simple evaluation (tiny heuristic)

[8]
18s

```python
eval_prompts = [
    "Explain what a function is in Python.",
    "If a train travels 60 km in 1.5 hours, what is its average speed?",
    "List two habits that improve learning with one-sentence reasons."
]
keywords = {
    "function": ["def", "return", "parameters", "reusable"],
    "speed": ["km", "hour", "speed", "40"],
    "learning": ["practice", "recall", "space", "sleep"]
}
def score_answer(q, a):
    a_low = a.lower(); score = min(len(a)/120.0, 1.0)
    if "function" in q.lower(): score += sum(1 for k in keywords["function"] if k in a_low)*0.25
    if "speed" in q.lower():    score += sum(1 for k in keywords["speed"] if k in a_low)*0.25
    if "learning" in q.lower(): score += sum(1 for k in keywords["learning"] if k in a_low)*0.25
    return round(float(score), 2)

for q in eval_prompts:
    ans = generate(q); s = score_answer(q, ans)
    print("\nQ:", q); print("A:", ans[:400] + ("..." if len(ans)>400 else "")); print("Heuristic score:", s)
```

```
Q: Explain what a function is in Python.
A: In Python, a function is a block of code that takes some input and returns a specific output. It is a special type of block of code that can be used to solve problems, process data, or perform tasks. A function is a reusable piece

A function in Python is defined using the `def` keyword followed by the function name and a block of code e...
Heuristic score: 1.75

Q: If a train travels 60 km in 1.5 hours, what is its average speed?
A: To calculate the average speed, we can use the formula:
Average Speed = (Total Distance / Total Time)

Given that the train travels 60 km in 1.5 hours, we can plug in the values:
Average Speed = (60 km / 1.5 hours) = 40 km/h

The average speed is 40 km/h, which means the train is traveling at a speed of approximately 40 km/h.
Heuristic score: 2.0

Q: List two habits that improve learning with one-sentence reasons.
A: There are several habits that can improve learning with one-sentence reasons. Here are two examples:

1. **Using the power of storytelling to explain complex concepts**. When explaining a complex topic, using storytelling can help make the information more relatable and memorable. For example, explaining a topic like "the human body"
Heuristic score: 1.0
```