LLM model

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk

```
pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.40.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.2)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3->transformers) (2023.6
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3->transformer
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)
```

```
import os

os.environ["HUGGINGFACEHUB_API_TOKEN"] = "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"
```

LangChain

```
from transformers import BartTokenizer
```

5s completed at 10:51 PM

LLM model

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk

LangChain

```
from transformers import BartTokenizer

# Load the BART tokenizer
tokenizer = BartTokenizer.from_pretrained("facebook/bart-large-mnli")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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://huggingface.co/settings/tokens), set it as secret in your Go
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 datasets.
  warnings.warn(
```

```
text = "When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning unit?"

# Tokenize the text
tokenized_input = tokenizer(text, return_tensors="pt")
```

```
from transformers import pipeline

# Load the text generation pipeline with the BART-large model
generator = pipeline("text-generation", model="facebook/bart-large-mnli")

# Define the prompt for generating the optimal indoor temperature setting
prompt = "When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning unit?"
```

5s completed at 10:51 PM

LLM model

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

RAM Disk

```
generator = pipeline("text-generation", model="facebook/bart-large-mnli")

# Define the prompt for generating the optimal indoor temperature setting
prompt = "When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning unit?"

# Generate the recommendation for the indoor temperature setting
recommendation = generator(prompt, max_length=50, num_return_sequences=1)[0]['generated_text']

# Print the recommendation
print("Recommended indoor temperature setting:", recommendation)
```

```
Some weights of BartForCausalLM were not initialized from the model checkpoint at facebook/bart-large-mnli and are newly initialized: ['lm_head.weigl
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples
Recommended indoor temperature setting: When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning u
```

```
from transformers import pipeline

# Load the zero-shot text classification pipeline with the desired model
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

# Define your prompt and potential indoor temperature settings
prompt = "When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning unit?"
labels = ["20°C", "22°C", "24°C", "26°C", "28°C"]  # Indoor temperature settings to choose from

# Perform zero-shot classification to get the recommended indoor temperature setting
result = classifier(prompt, labels)

# Print the recommendation
print("Recommended indoor temperature setting:", result['labels'][0])
```

5s completed at 10:51 PM

LLM model

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[13] # Print the recommendation
     print("Recommended indoor temperature setting:", result['labels'][0])
```

```
Recommended indoor temperature setting: 24°C
```

Prompt Template

```
from transformers import pipeline

# Load the zero-shot text classification pipeline
classifier = pipeline("zero-shot-classification")

# Define your prompt and potential indoor temperature settings
prompt = "When the outdoor temperature is 25°C, what is the best indoor temperature setting for the air conditioning unit?"
labels = ["20°C", "22°C", "24°C", "26°C", "28°C"]  # Indoor temperature settings to choose from

# Perform zero-shot classification to get the recommended indoor temperature setting
result = classifier(prompt, labels)

# Print the recommendation
print("Recommended indoor temperature setting:", result['labels'][0])
```

```
No model was supplied, defaulted to facebook/bart-large-mnli and revision c626438 (https://huggingface.co/facebook/bart-large-mnli).
Using a pipeline without specifying a model name and revision in production is not recommended.
Recommended indoor temperature setting: 24°C
```

+ Code + Text

Output Parsers

5s completed at 10:51 PM

LLM model

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

Output Parsers

```
import spacy

# Load the English tokenizer, tagger, parser, NER, and word vectors
nlp = spacy.load("en_core_web_sm")

# Process the user command
def process_user_command(user_command):
    doc = nlp(user_command)
    device_name = None
    action = None

    for token in doc:
        if token.text.lower() in ["燈", "燈具", "燈光", "照明"]:
            device_name = "燈"
        elif token.text.lower() in ["開啟", "打開", "啟動"]:
            action = "開啟"
        elif token.text.lower() in ["關閉", "關掉", "停止"]:
            action = "關閉"

    return {"device_name": device_name, "action": action}

# Example user command
user_command = "請把客廳的燈開啟"
json_output = process_user_command(user_command)
print(json_output)
```

```
{'device_name': None, 'action': None}
```

5s completed at 10:55 PM

LLM model

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

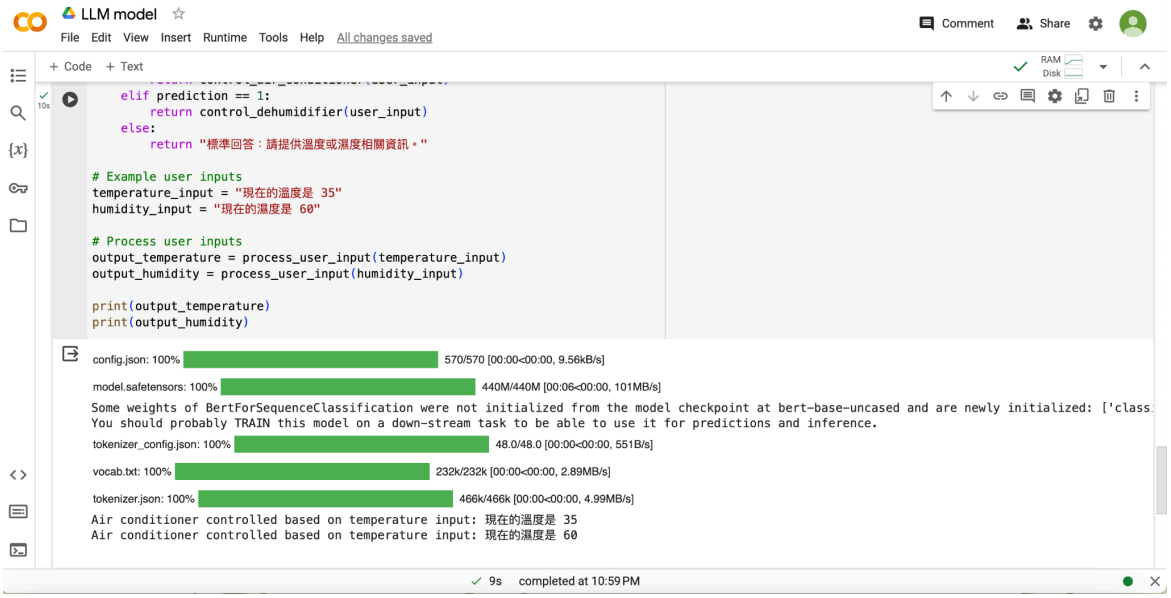
+ Code + Text

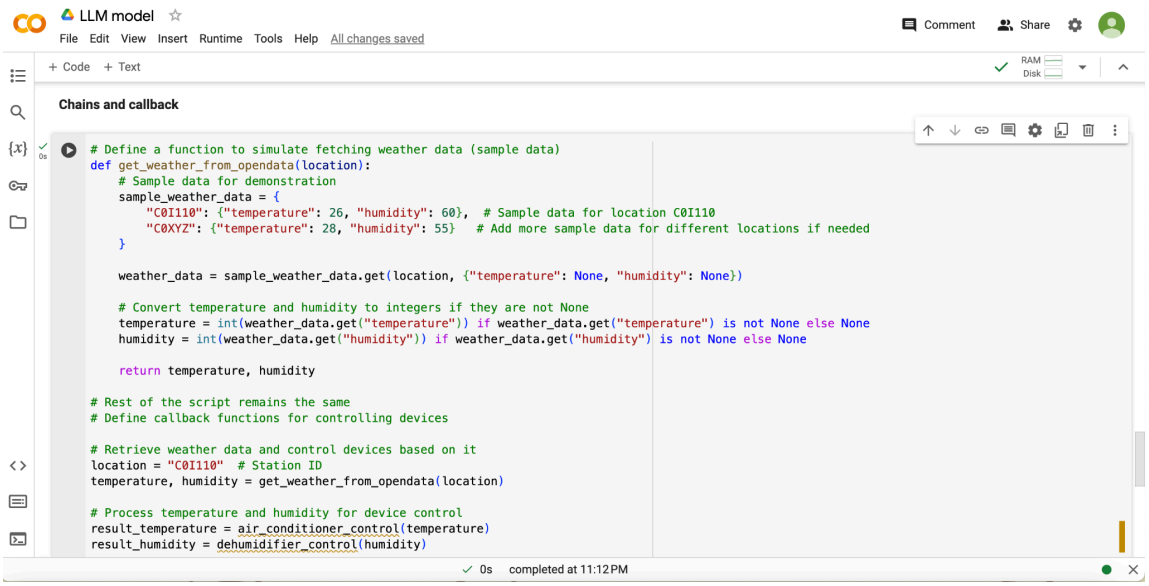
RAM Disk

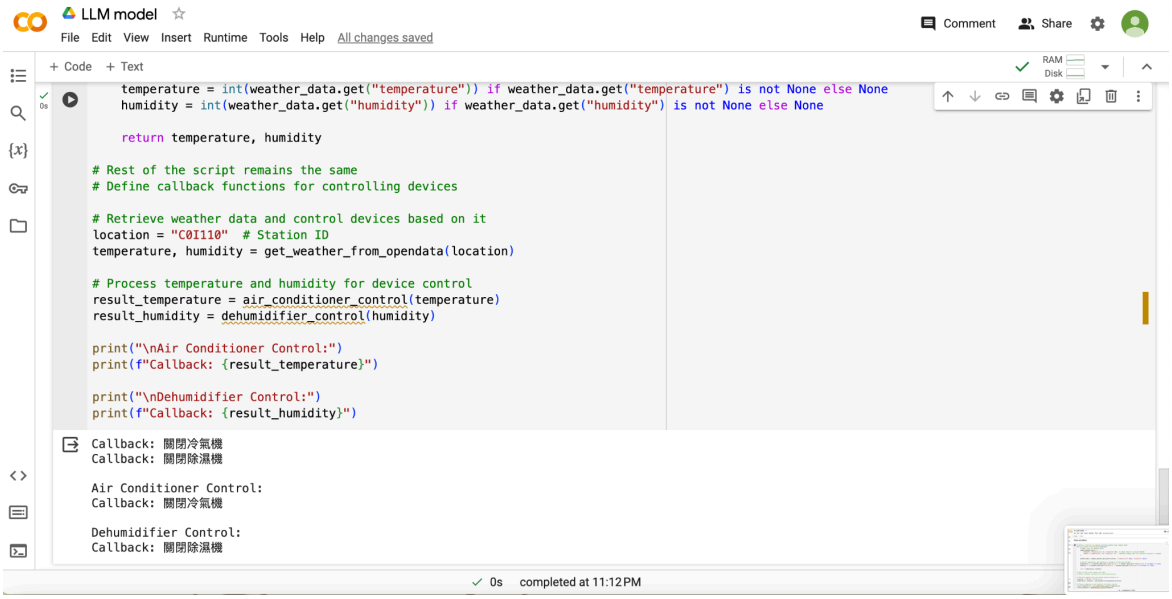
Chains

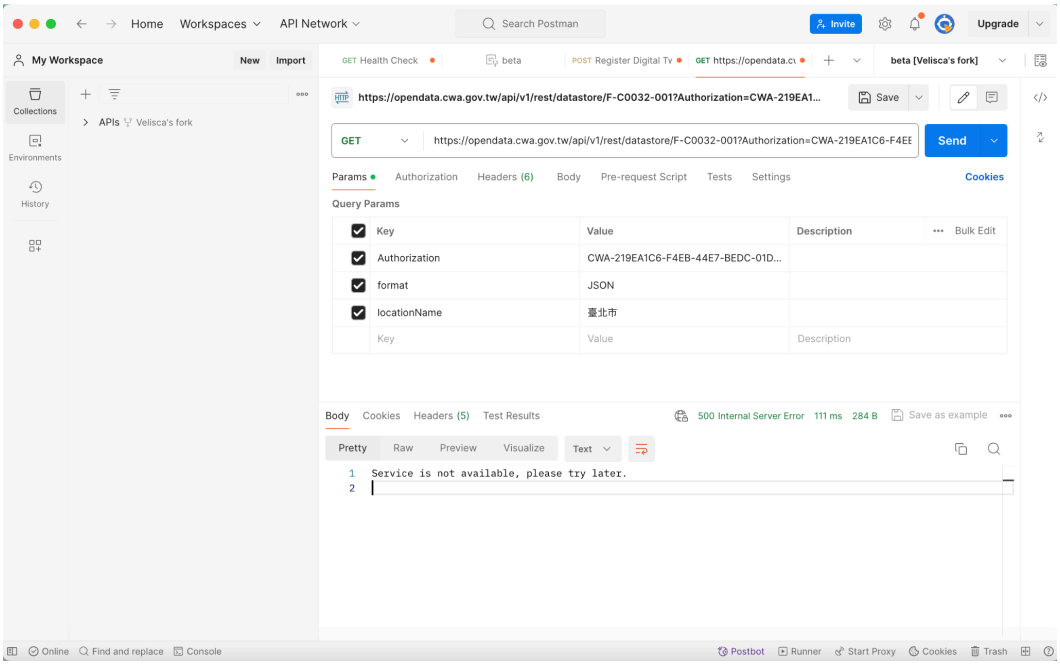
```
from transformers import BertTokenizer, BertForSequenceClassification
import torch

# Load the pre-trained BERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Define functions to control the air conditioner and dehumidifier
def control_air_conditioner(input_text):
    return f"Air conditioner controlled based on temperature input: {input_text}"

def control_dehumidifier(input_text):
    return f"Dehumidifier controlled based on humidity input: {input_text}"

# Process user input and route to the appropriate execution chain
def process_user_input(user_input):
    inputs = tokenizer(user_input, return_tensors="pt", padding=True, truncation=True)
    outputs = model(**inputs)
    prediction = torch.argmax(outputs.logits).item()

    if prediction == 0:
        return control_air_conditioner(user_input)
    elif prediction == 1:
        return control_dehumidifier(user_input)
    else:
        return "標準回答：請提供溫度或濕度相關資訊。"
```

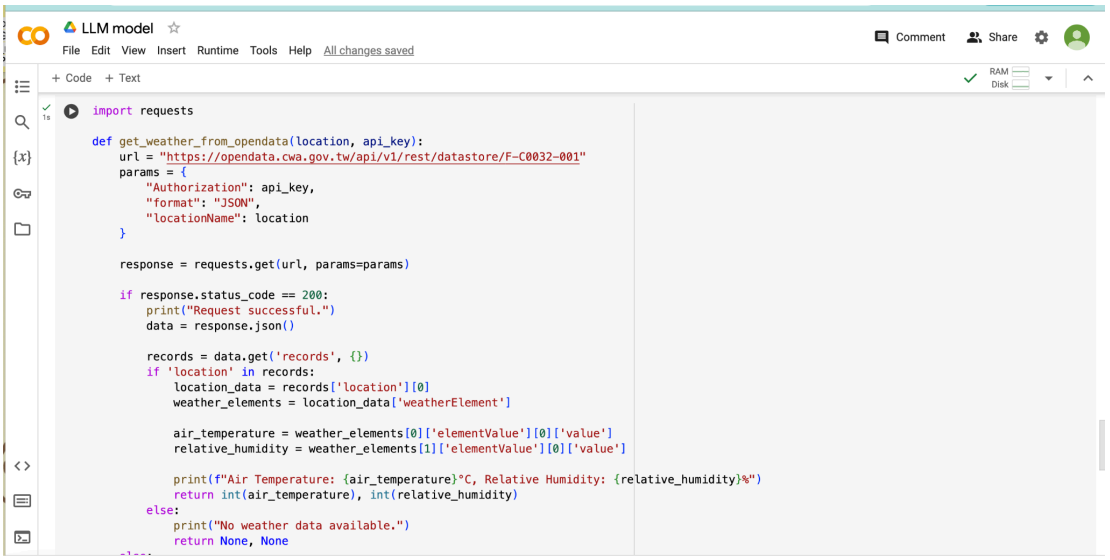
9s completed at 10:59 PM

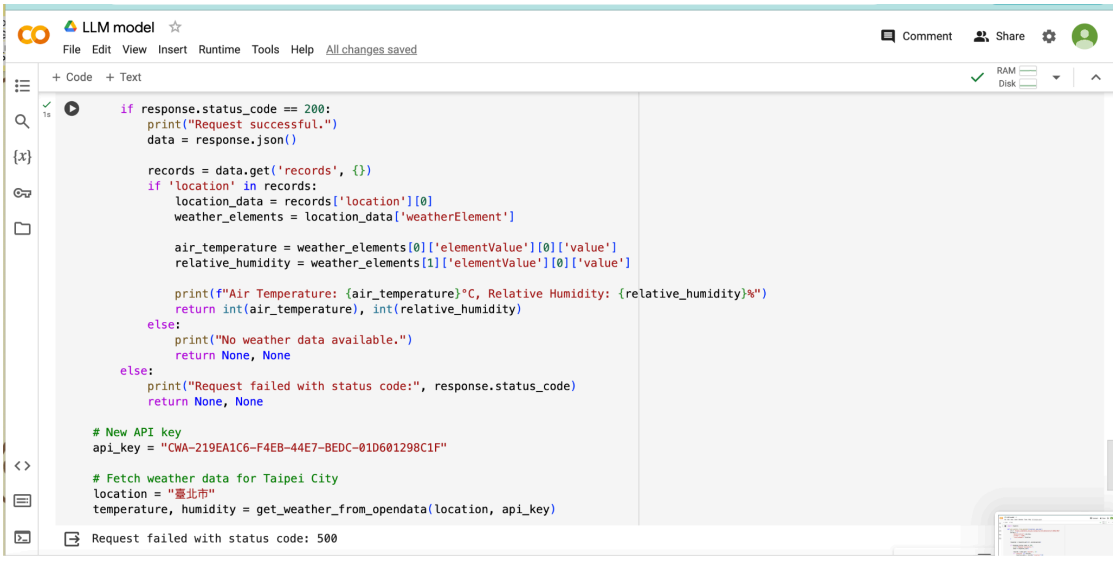

LLM model
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
elif prediction == 1:
return control_dehumidifier(user_input)
else:
return "標準回答：請提供溫度或濕度相關資訊。"
# Example user inputs
temperature_input = "現在的溫度是 35"
humidity_input = "現在的濕度是 60"
# Process user inputs
output_temperature = process_user_input(temperature_input)
output_humidity = process_user_input(humidity_input)
print(output_temperature)
print(output_humidity)
config.json: 100% 570/570 [00:00<00:00, 9.56kB/s]
model.safetensors: 100% 440M/440M [00:06<00:00, 101MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['class
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 551B/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 2.89MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 4.99MB/s]
Air conditioner controlled based on temperature input: 現在的溫度是 35
Air conditioner controlled based on temperature input: 現在的濕度是 60
9s completed at 10:59 PM


LLM model
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Chains and callback
# Define a function to simulate fetching weather data (sample data)
def get_weather_from_opendata(location):
# Sample data for demonstration
sample_weather_data = {
"C0I110": {"temperature": 26, "humidity": 60}, # Sample data for location C0I110
"C0XYZ": {"temperature": 28, "humidity": 55} # Add more sample data for different locations if needed
}
weather_data = sample_weather_data.get(location, {"temperature": None, "humidity": None})
# Convert temperature and humidity to integers if they are not None
temperature = int(weather_data.get("temperature")) if weather_data.get("temperature") is not None else None
humidity = int(weather_data.get("humidity")) if weather_data.get("humidity") is not None else None
return temperature, humidity
# Rest of the script remains the same
# Define callback functions for controlling devices
# Retrieve weather data and control devices based on it
location = "C0I110" # Station ID
temperature, humidity = get_weather_from_opendata(location)
# Process temperature and humidity for device control
result_temperature = air_conditioner_control(temperature)
result_humidity = dehumidifier_control(humidity)
0s completed at 11:12 PM


LLM model
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
temperature = int(weather_data.get("temperature")) if weather_data.get("temperature") is not None else None
humidity = int(weather_data.get("humidity")) if weather_data.get("humidity") is not None else None
return temperature, humidity
# Rest of the script remains the same
# Define callback functions for controlling devices
# Retrieve weather data and control devices based on it
location = "C0I110" # Station ID
temperature, humidity = get_weather_from_opendata(location)
# Process temperature and humidity for device control
result_temperature = air_conditioner_control(temperature)
result_humidity = dehumidifier_control(humidity)
print("\nAir Conditioner Control:")
print(f"Callback: {result_temperature}")
print("\nDehumidifier Control:")
print(f"Callback: {result_humidity}")
Callback: 關閉冷氣機
Callback: 關閉除濕機
Air Conditioner Control:
Callback: 關閉冷氣機
Dehumidifier Control:
Callback: 關閉除濕機
0s completed at 11:12 PM


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
APIs
Velisca's fork
GET Health Check
beta
POST Register Digital Tv
GET https://opendata.cv
beta [Velisca's fork]
https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-C0032-001?Authorization=CWA-219EA1...
Save
GET
https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-C0032-001?Authorization=CWA-219EA1C6-F4EE
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Authorization
CWA-219EA1C6-F4EB-44E7-BEDC-01D...
format
JSON
locationName
臺北市
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
500 Internal Server Error
111 ms
284 B
Save as example
Pretty
Raw
Preview
Visualize
Text
1 Service is not available, please try later.
2
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


LLM model
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
import requests
def get_weather_from_opendata(location, api_key):
url = "https://opendata.cwa.gov.tw/api/v1/rest/datastore/F-C0032-001"
params = {
"Authorization": api_key,
"format": "JSON",
"locationName": location
}
response = requests.get(url, params=params)
if response.status_code == 200:
print("Request successful.")
data = response.json()
records = data.get('records', {})
if 'location' in records:
location_data = records['location'][0]
weather_elements = location_data['weatherElement']
air_temperature = weather_elements[0]['elementValue'][0]['value']
relative_humidity = weather_elements[1]['elementValue'][0]['value']
print(f"Air Temperature: {air_temperature}°C, Relative Humidity: {relative_humidity}%")
return int(air_temperature), int(relative_humidity)
else:
print("No weather data available.")
return None, None


LLM model
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
if response.status_code == 200:
print("Request successful.")
data = response.json()
records = data.get('records', {})
if 'location' in records:
location_data = records['location'][0]
weather_elements = location_data['weatherElement']
air_temperature = weather_elements[0]['elementValue'][0]['value']
relative_humidity = weather_elements[1]['elementValue'][0]['value']
print(f"Air Temperature: {air_temperature}°C, Relative Humidity: {relative_humidity}%")
return int(air_temperature), int(relative_humidity)
else:
print("No weather data available.")
return None, None
else:
print("Request failed with status code:", response.status_code)
return None, None
# New API key
api_key = "CWA-219EA1C6-F4EB-44E7-BEDC-01D601298C1F"
# Fetch weather data for Taipei City
location = "臺北市"
temperature, humidity = get_weather_from_opendata(location, api_key)
Request failed with status code: 500