import requests

import base64

import hmac

import time

from urllib.parse import quote

import matplotlib.pyplot as plt

from datetime import datetime

import sys

# 设置中文字体支持

plt.rcParams["font.family"] = ["SimHei", "WenQuanYi Micro Hei", "Heiti TC"]

plt.rcParams["axes.unicode\_minus"] = False # 解决负号显示问题

# -------------------------- Token 生成函数 --------------------------

def generate\_one\_net\_token(user\_id, access\_key):

version = '2022-05-01'

res = 'userid/%s' % user\_id

et = str(int(time.time()) + 3600) # Token 过期时间，这里设为 1 小时后

method = 'sha1'

# 解码 access\_key

key = base64.b64decode(access\_key)

# 拼接签名原文

org = f"{et}\n{method}\n{res}\n{version}"

# 计算 HMAC 签名

sign\_b = hmac.new(key=key, msg=org.encode(), digestmod=method)

sign = base64.b64encode(sign\_b.digest()).decode()

# URL 编码（按需处理特殊字符）

sign = quote(sign, safe='')

res = quote(res, safe='')

# 拼接成最终的 Token（对应 authorization）

token = f"version={version}&res={res}&et={et}&method={method}&sign={sign}"

return token

# -------------------------- 数据解析与分析函数 --------------------------

def parse\_device\_data(data):

"""解析设备属性数据（同时提取温度和湿度）"""

if not isinstance(data, dict):

return None

# 获取响应中的 data 字段（是一个列表）

properties\_list = data.get('data', [])

if not isinstance(properties\_list, list):

return None

properties = {}

# 遍历属性列表，提取标识符和值

for item in properties\_list:

identifier = item.get('identifier')

value = item.get('value')

# 尝试将数值字符串转换为 float 类型

if isinstance(value, str):

try:

value = float(value)

except ValueError:

pass # 无法转换时保持原字符串

if identifier and value is not None:

properties[identifier] = value

return properties

def analyze\_environment\_data(properties):

"""分析环境数据（温度和湿度）"""

if not properties:

return "没有可分析的属性数据"

analysis\_results = []

# 分析温度数据（修改为匹配实际设备的属性名：temp）

temperature = properties.get('temp')

if temperature is not None:

temp\_status = "正常"

if isinstance(temperature, (int, float)):

if temperature > 30:

temp\_status = "偏高"

elif temperature < 10:

temp\_status = "偏低"

analysis\_results.append(f"温度: {temperature}°C ({temp\_status})")

# 分析湿度数据（修改为匹配实际设备的属性名：humidity）

humidity = properties.get('humidity')

if humidity is not None:

hum\_status = "正常"

if isinstance(humidity, (int, float)):

if humidity > 80:

hum\_status = "偏高"

elif humidity < 30:

hum\_status = "偏低"

analysis\_results.append(f"湿度: {humidity}% ({hum\_status})")

return "\n".join(analysis\_results) if analysis\_results else "未找到温度和湿度数据"

# -------------------------- 数据记录与可视化函数 --------------------------

def record\_data(time\_stamp, properties, data\_records):

"""记录每次采集的温湿度数据"""

# 提取温度和湿度数据（修改为匹配实际设备的属性名）

temperature = properties.get('temp')

humidity = properties.get('humidity')

if temperature is not None or humidity is not None:

data\_records.append((time\_stamp, temperature, humidity))

return data\_records

def visualize\_live\_data(data\_records):

"""可视化实时采集的温湿度数据，并生成表格"""

if not data\_records:

print("没有温湿度的实时数据可供可视化")

return

# 生成温湿度数据表格（使用普通文本格式）

print("\n温湿度数据记录表：")

print("=" \* 60)

print(f"{'采集时间':<20} {'温度':<10} {'湿度':<10}")

print("-" \* 60)

for timestamp, temp, hum in data\_records:

temp\_str = f"{temp}°C" if temp is not None else "-"

hum\_str = f"{hum}%" if hum is not None else "-"

print(f"{timestamp:<20} {temp\_str:<10} {hum\_str:<10}")

print("=" \* 60)

# 绘制温湿度趋势图

if any(record[1] is not None for record in data\_records) or any(record[2] is not None for record in data\_records):

timestamps = [record[0] for record in data\_records]

temps = [record[1] for record in data\_records]

hums = [record[2] for record in data\_records]

# 1. 绘制分离的子图

plt.figure(figsize=(14, 7))

# 绘制温度曲线

if any(t is not None for t in temps):

plt.subplot(1, 2, 1)

plt.plot(timestamps, [t for t in temps if t is not None], 'o-', color='#1f77b4', linewidth=2, label='温度')

plt.title('温度趋势图', fontsize=14)

plt.xlabel('采集时间', fontsize=12)

plt.ylabel('温度 (°C)', fontsize=12)

plt.grid(True, linestyle='--', alpha=0.7)

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

# 绘制湿度曲线

if any(h is not None for h in hums):

plt.subplot(1, 2, 2)

plt.plot(timestamps, [h for h in hums if h is not None], 's-', color='#ff7f0e', linewidth=2, label='湿度')

plt.title('湿度趋势图', fontsize=14)

plt.xlabel('采集时间', fontsize=12)

plt.ylabel('湿度 (%)', fontsize=12)

plt.grid(True, linestyle='--', alpha=0.7)

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.savefig('温湿度趋势图\_分图.png', dpi=300, bbox\_inches='tight')

print("\n温湿度趋势图（分图）已保存为: 温湿度趋势图\_分图.png")

# 2. 新增：绘制在同一图表中的组合图

if any(t is not None for t in temps) and any(h is not None for h in hums):

plt.figure(figsize=(14, 7))

# 创建主坐标轴（温度）

ax1 = plt.subplot(111)

line1 = ax1.plot(timestamps, [t for t in temps if t is not None], 'o-', color='#1f77b4', linewidth=2, label='温度')

ax1.set\_ylabel('温度 (°C)', color='#1f77b4', fontsize=12)

ax1.tick\_params(axis='y', labelcolor='#1f77b4')

# 创建次坐标轴（湿度）

ax2 = ax1.twinx()

line2 = ax2.plot(timestamps, [h for h in hums if h is not None], 's-', color='#ff7f0e', linewidth=2, label='湿度')

ax2.set\_ylabel('湿度 (%)', color='#ff7f0e', fontsize=12)

ax2.tick\_params(axis='y', labelcolor='#ff7f0e')

# 设置标题和x轴

plt.title('温湿度趋势组合图', fontsize=14)

ax1.set\_xlabel('采集时间', fontsize=12)

plt.xticks(rotation=45, ha='right')

# 添加图例

lines = line1 + line2

labels = [l.get\_label() for l in lines]

plt.legend(lines, labels, loc='upper left')

plt.grid(True, linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.savefig('温湿度趋势图\_组合.png', dpi=300, bbox\_inches='tight')

print("温湿度趋势图（组合）已保存为: 温湿度趋势图\_组合.png")

plt.show()

else:

print("由于缺少温度或湿度数据，无法生成组合图")

else:

print("没有足够的数据绘制趋势图")

# -------------------------- 主程序 --------------------------

if \_\_name\_\_ == "\_\_main\_\_":

# 1. 替换为你的实际参数

product\_id = "817X0XSh39"

device\_name = "dht11"

user\_id = "451383"

access\_key = "k33iniqM8EtNVzF+yXm+8EPaQspHKBb1aOiDXpMhgmVxxW7Ja6zg+4doplk9WH2s"

# 2. 配置采集参数

INTERVAL = 30 # 数据采集间隔（秒）

MAX\_SAMPLES = 10 # 最大采集次数

data\_records = [] # 存储采集的数据

print(f"开始按每 {INTERVAL} 秒一次的频率采集设备数据，总共采集 {MAX\_SAMPLES} 次...")

for i in range(MAX\_SAMPLES):

try:

# 生成新的 Token（每次请求都生成，确保不过期）

authorization = generate\_one\_net\_token(user\_id, access\_key)

# 构建 API 请求 - 查询设备属性

url = f"https://iot-api.heclouds.com/thingmodel/query-device-property?product\_id={product\_id}&device\_name={device\_name}"

headers = {

"authorization": authorization # 把生成的凭证放到请求头

}

# 发送请求

current\_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

print(f"\n第 {i+1}/{MAX\_SAMPLES} 次采集 @ {current\_time}")

response = requests.get(url, headers=headers)

if response.status\_code == 200:

data = response.json()

# 解析数据

properties = parse\_device\_data(data)

if properties:

print("设备属性数据：")

for prop\_name, prop\_value in properties.items():

print(f" {prop\_name}: {prop\_value}")

# 分析环境数据

print("\n环境数据分析结果:")

print(analyze\_environment\_data(properties))

# 记录数据

data\_records = record\_data(current\_time, properties, data\_records)

else:

print("无法解析设备属性数据")

else:

print(f"请求失败，状态码：{response.status\_code}，响应：{response.text}")

except requests.exceptions.RequestException as e:

print(f"请求异常：{e}")

except Exception as e:

print(f"发生未知错误：{e}")

# 如果不是最后一次采集，则等待指定间隔时间

if i < MAX\_SAMPLES - 1:

print(f"\n等待 {INTERVAL} 秒后进行下一次采集...")

time.sleep(INTERVAL)

# 采集完成后，可视化数据

if data\_records:

visualize\_live\_data(data\_records)

print("\n数据采集完成！")

print(f"共采集了 {len(data\_records)} 个数据点")

else:

print("\n没有采集到有效数据")