# 树莓派实验二

## 实验内容

按键控制呼吸灯的开启及关闭，红绿蓝交替呼吸灯效果

要求：按键按一下，交替呼吸灯开启

按键再按一下，关闭

如此循环

阅读参考书相关介绍，理解dweet\_led.py并整理进实验报告

## 实验代码

from gpiozero import TrafficLights,Button

from time import sleep

from gpiozero import PWMLED

from signal import pause

red\_led = PWMLED(2)

blue\_led = PWMLED(3)

green\_led = PWMLED(4)

button = Button(12,pull\_up=False)

def breath():

sleep(2)

green\_led.off()

blue\_led.pulse()

sleep(2)

bule\_led.off()

red\_led.pulse()

sleep(2)

red\_led.off()

green\_led.pulse()

def stop:

green\_led.off()

red\_led.off()

blue\_led.off()

# 定义按键事件处理函数

def button\_pressed():

print("button is pressed")

global running

if running :

running = False

print("关灯")

else :

running = True

print("开灯")

running = True

flag = True

while True:

if flag:

print("默认开灯")

breath()

flag = False

if running :

print("主函数:breath")

breath()

else :

print("主函数:stop")

stop()

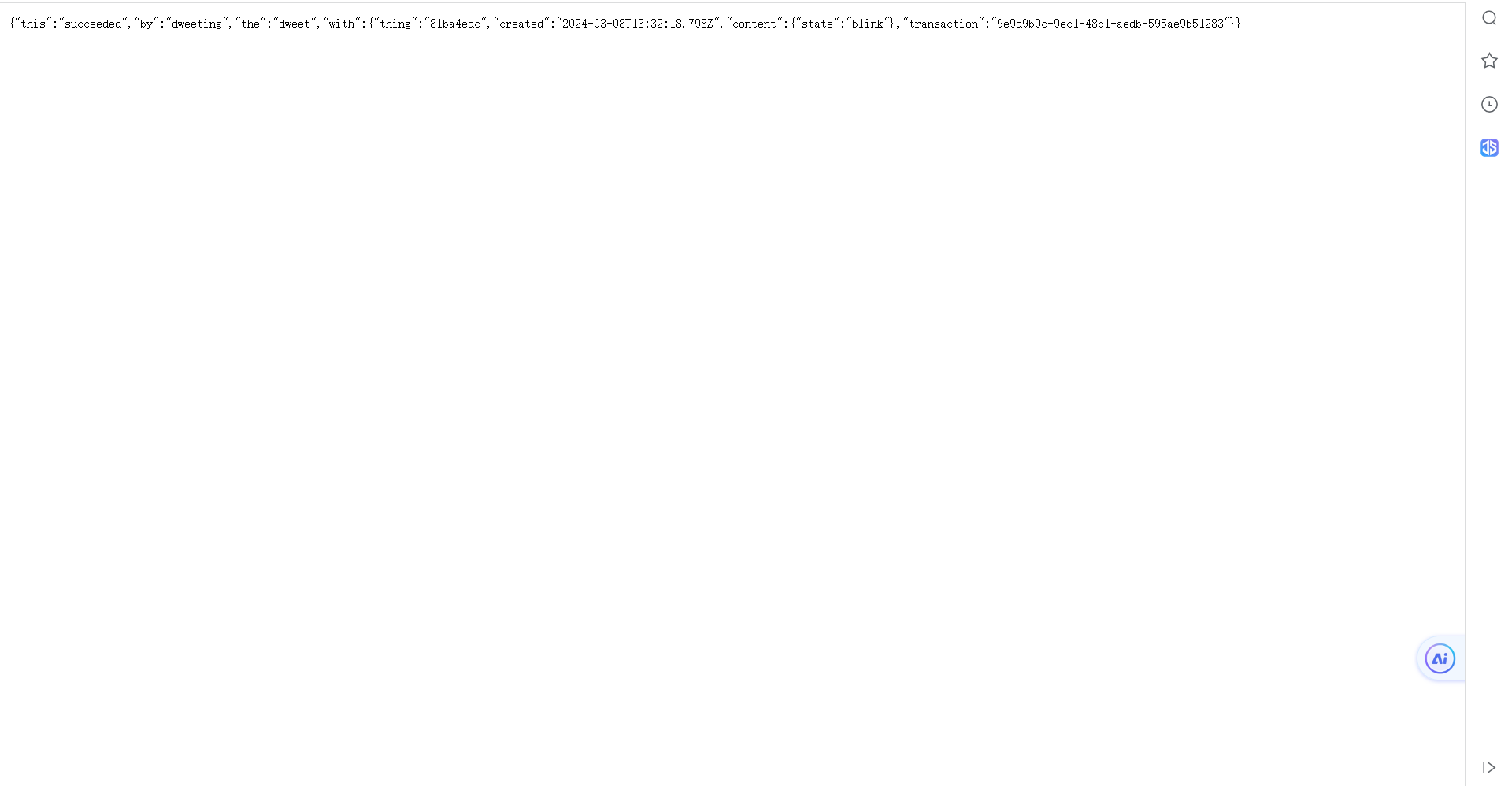
button.when\_pressed = button\_pressed

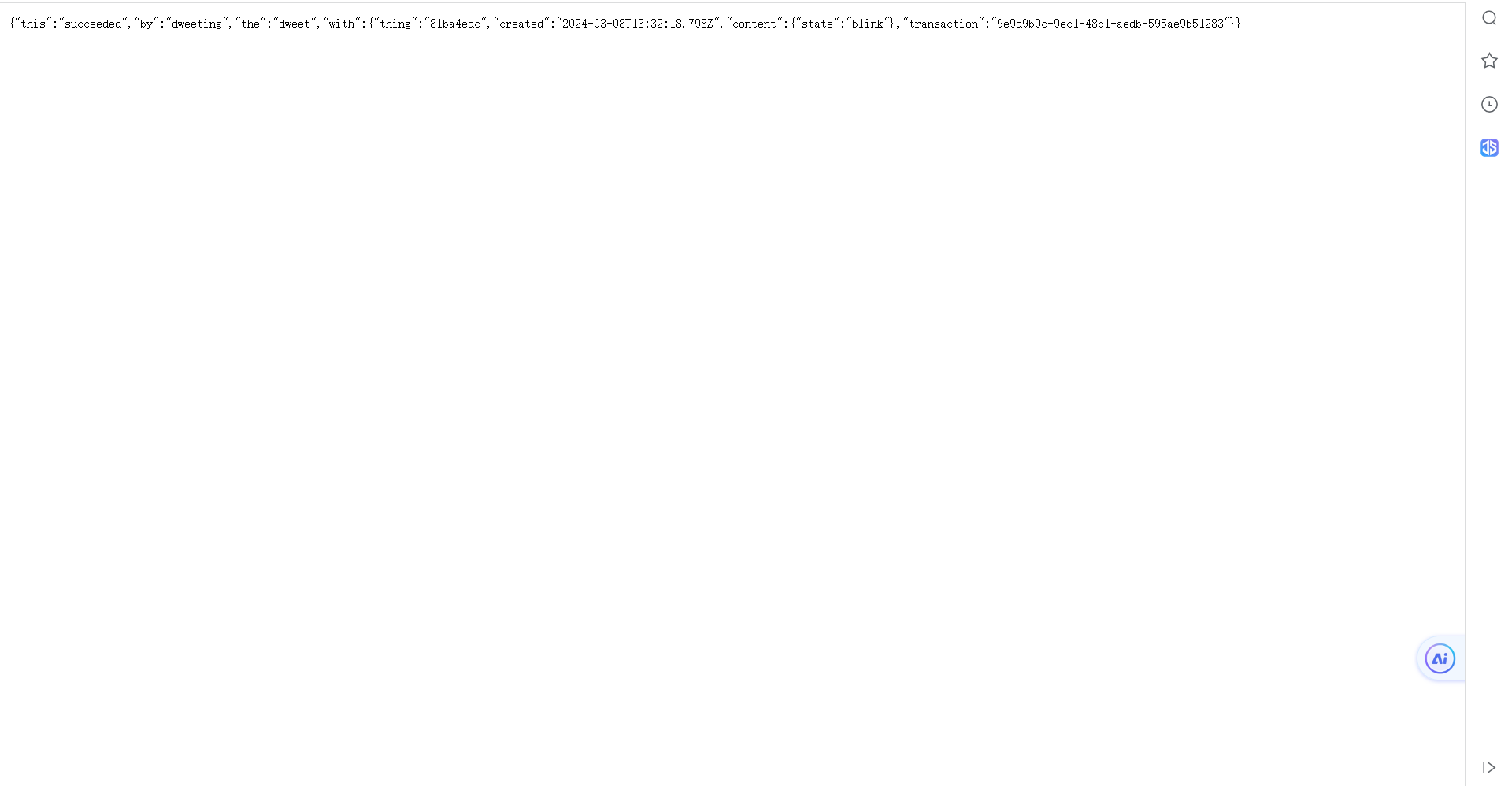
         实验视频见附件

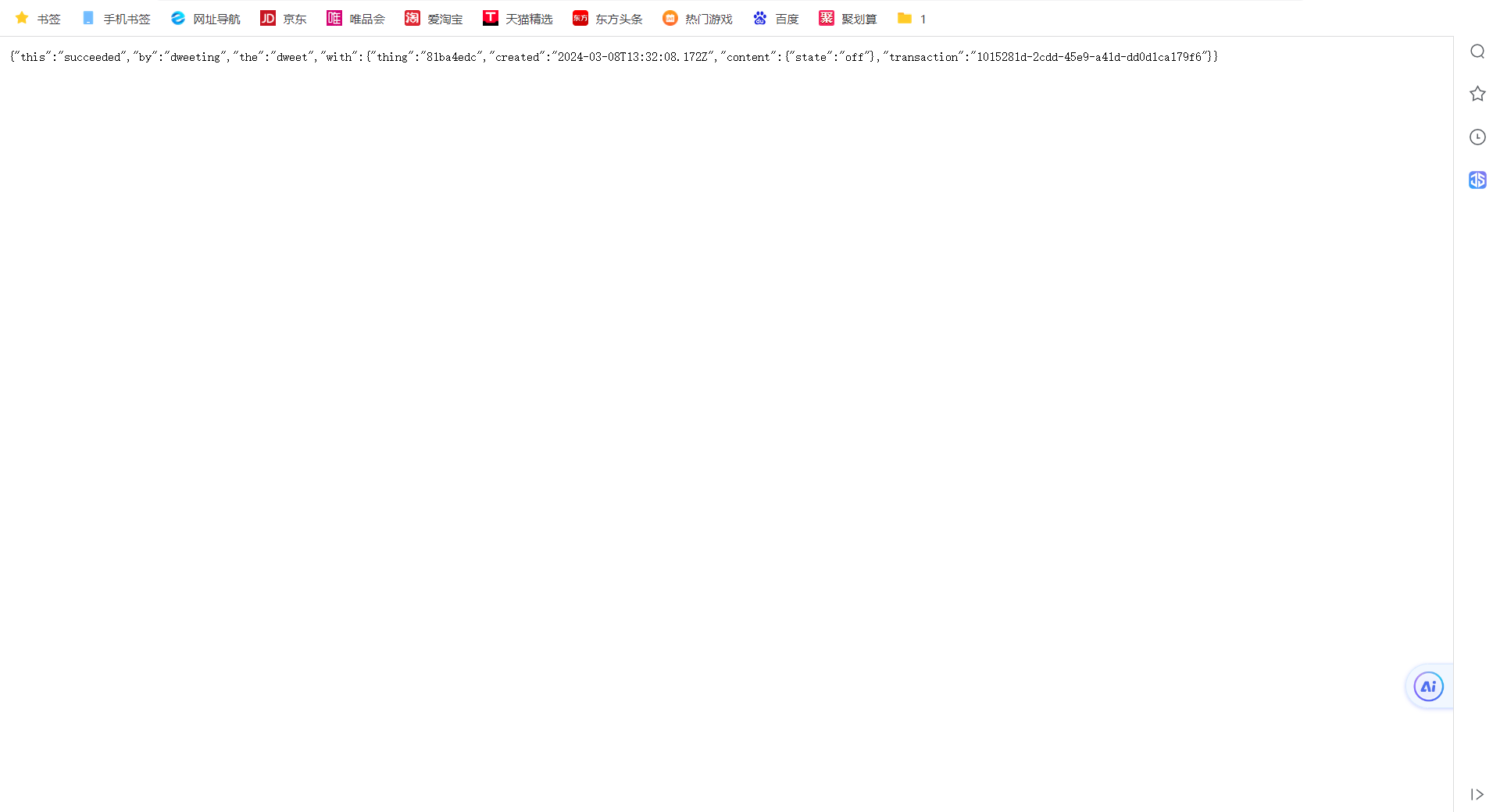
## 理解代码

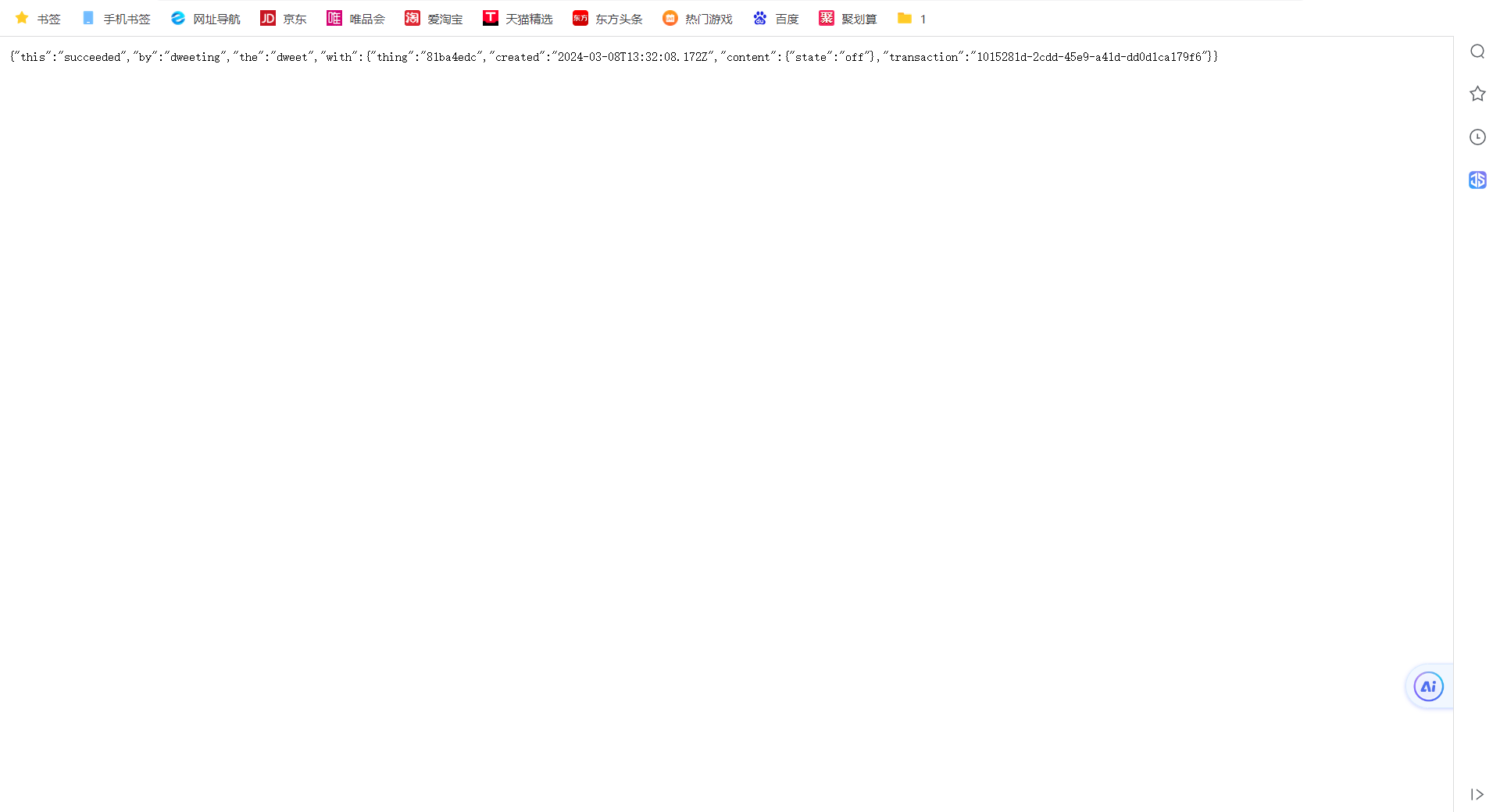
dweet\_led.py

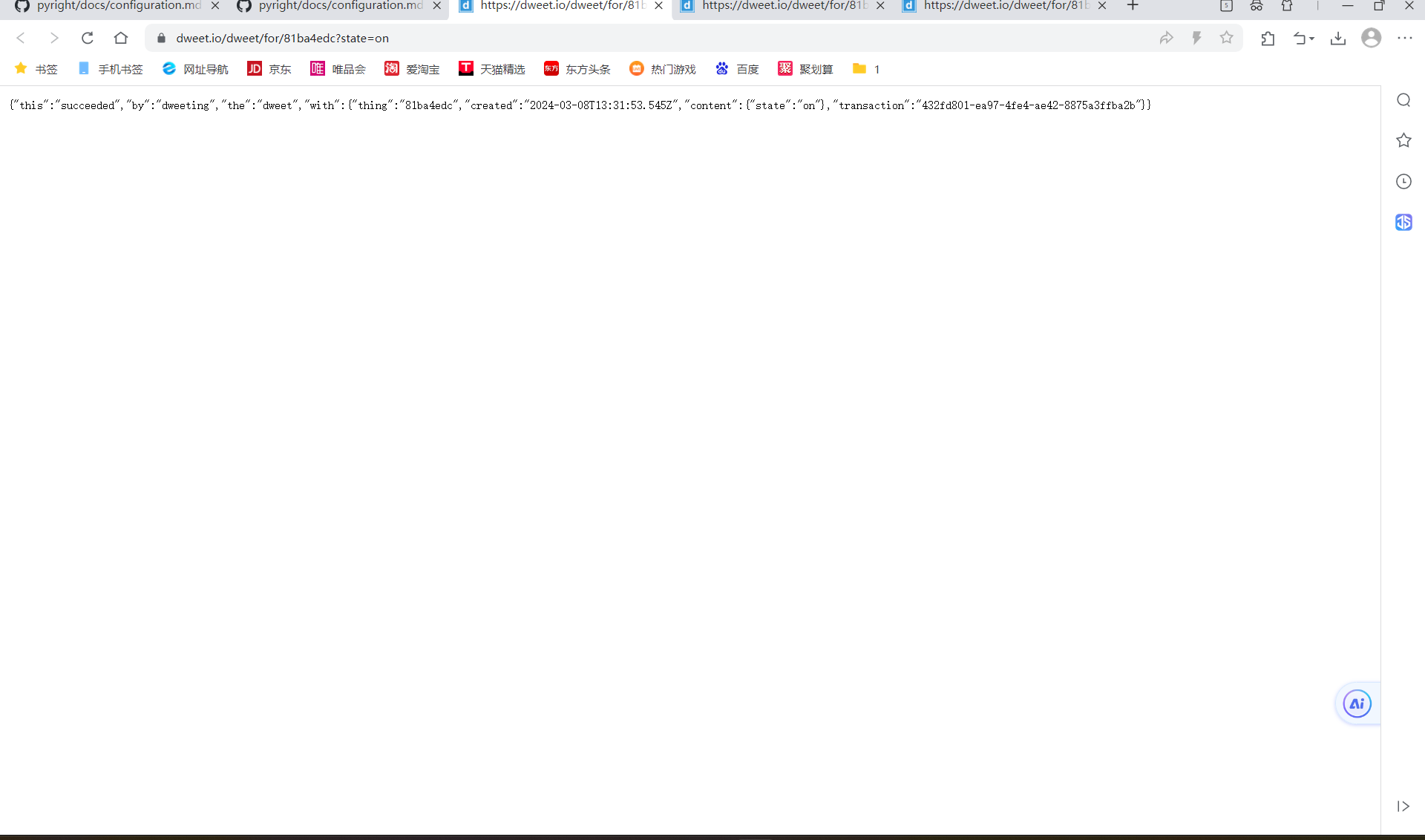
运行结果

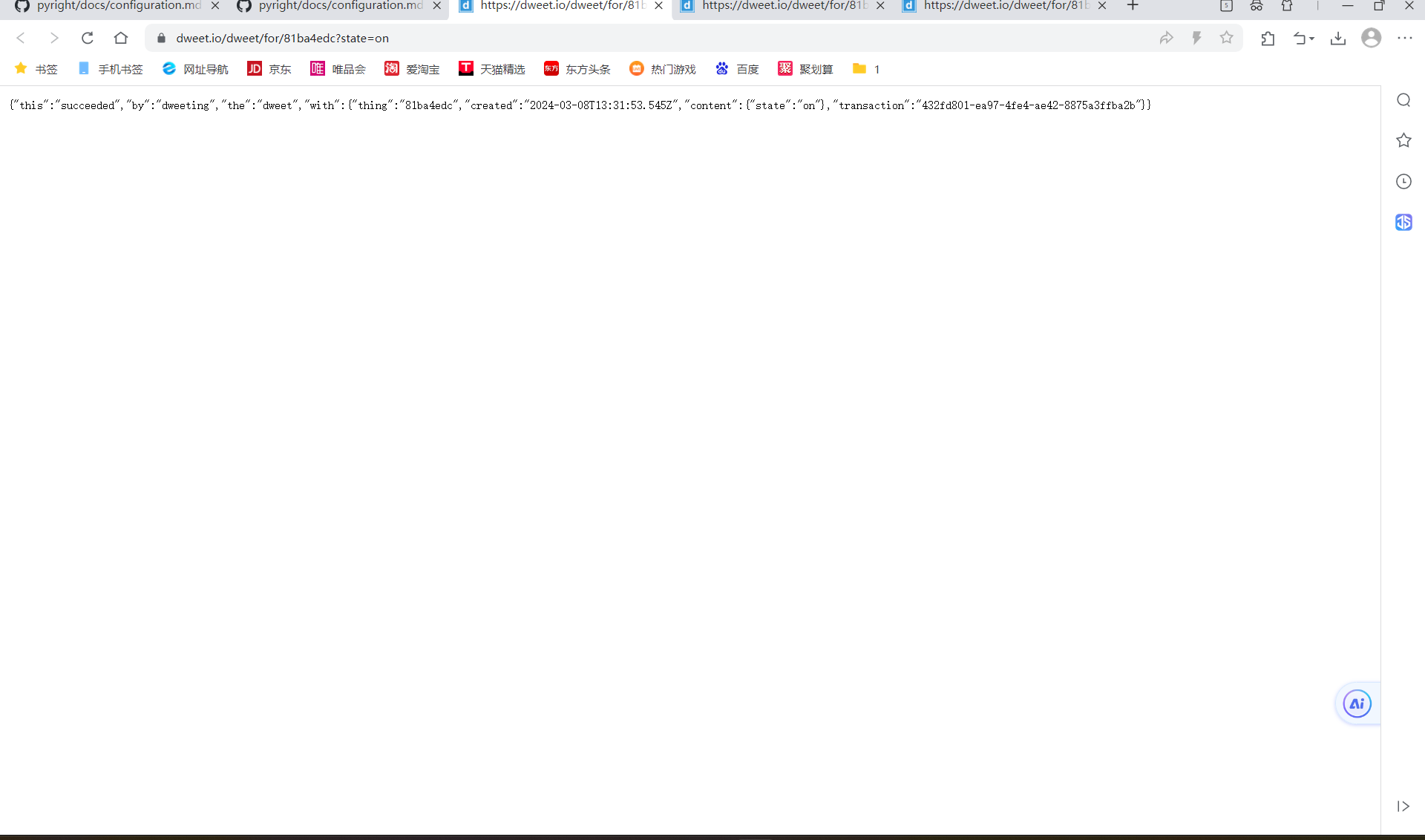








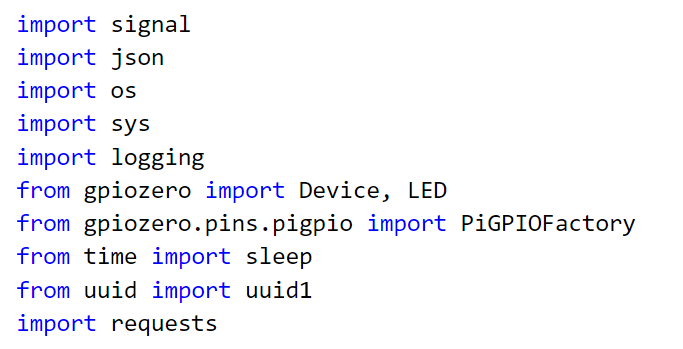




参考dweet.io和相关博客可知

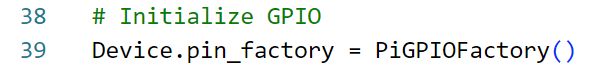
该代码通过dweet.io来远程控制LED灯。

1. 依赖



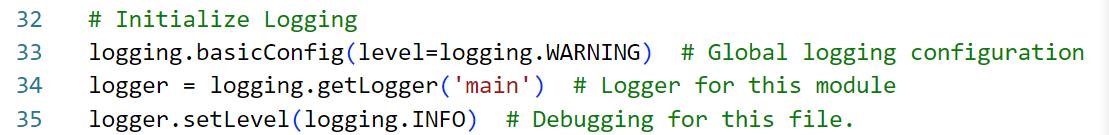
依赖gpiozero来操作GPIO，pigpio作为GPIO的底层驱动，requests用于HTTP通信。

2. GPIO配置：



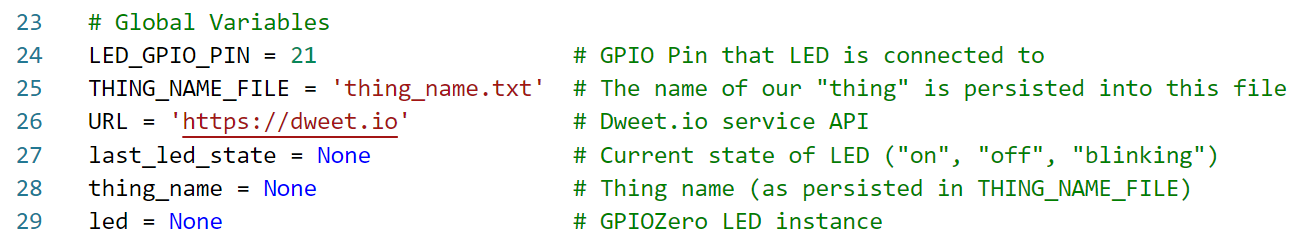
-使用`Device.pin\_factory = PiGPIOFactory()`来初始化GPIO工厂，这是针对Raspberry Pi的配置。

3. 日志设置：



使用`logging`模块，有全局的日志设置和针对当前文件的调试级别设置。

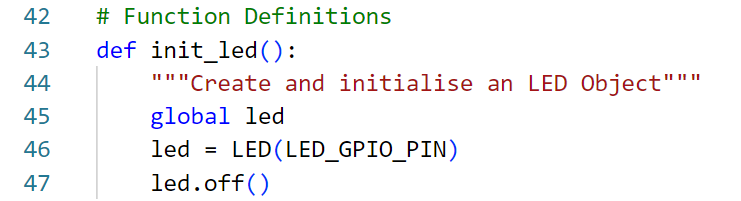
4. 全局变量：



LED连接的GPIO引脚号`LED\_GPIO\_PIN`(21)，LED的初始状态为`off`，`thing\_name`变量用于存储“thing”的名称,`thing\_name.txt`用于记录“thing”名称。

5. 函数定义：

- `init\_led()`：初始化LED对象。



- `resolve\_thing\_name(thing\_file)`：从文件中读取或创建一个新的“thing”名称。

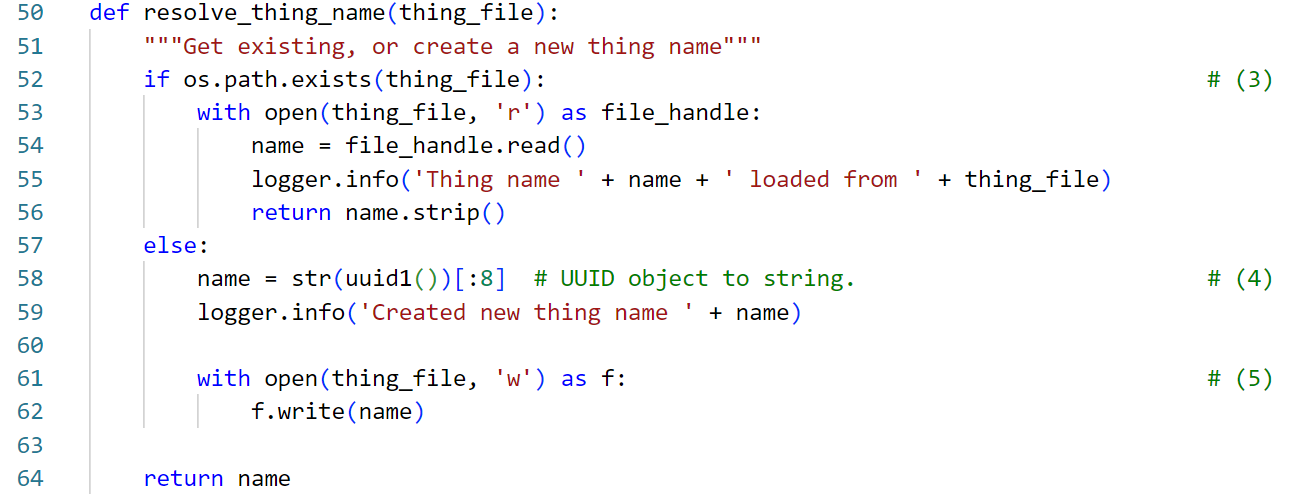
如果不从文件中读取thing\_name,就生成一个uuid1对象

ps:

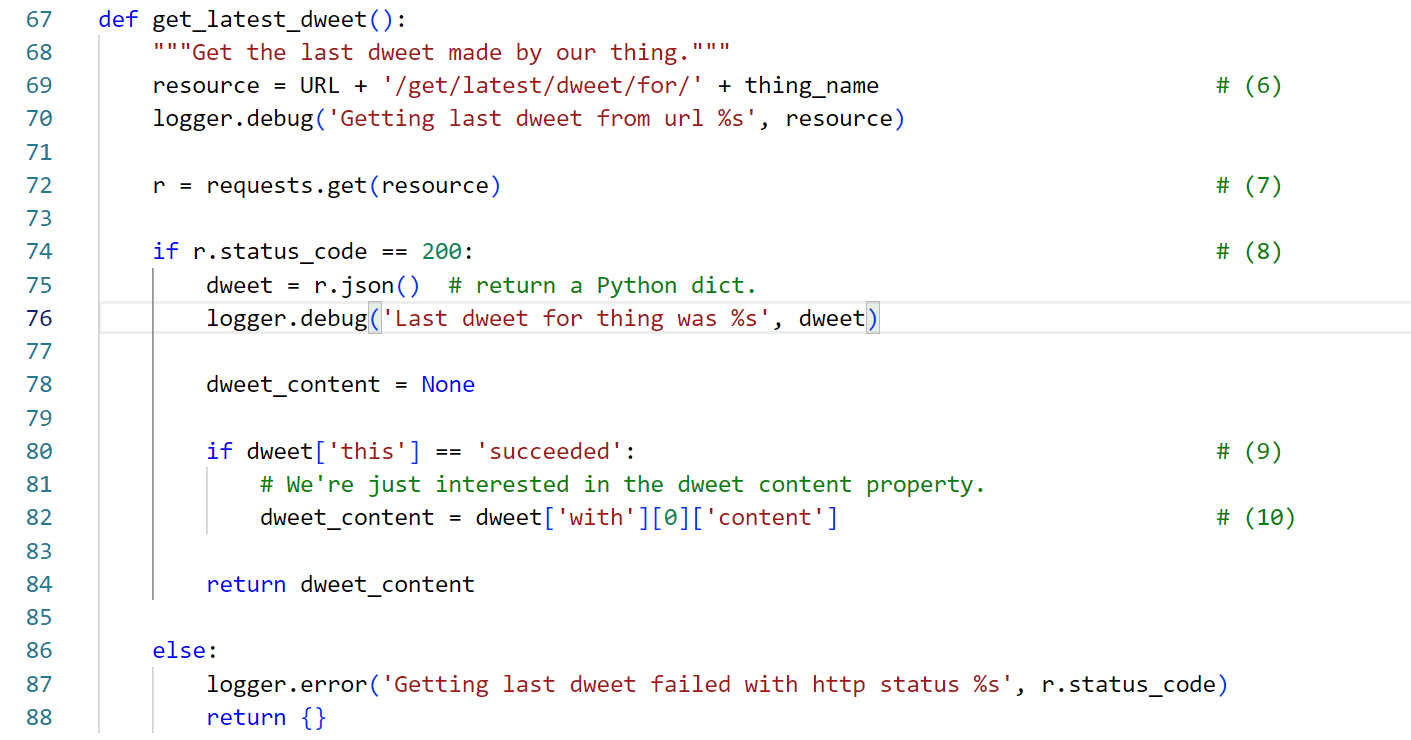
UUID（Universally Unique Identifier，通用唯一识别码）是一种用于标识信息的技术，它旨在创建一个唯一的标识符，这个标识符在整个宇宙中几乎是不可能重复的。UUID有多种格式，包括UUID1、UUID3、UUID4、UUID5和UUID6，但在Python中通常使用的是UUID1和UUID4。

UUID1是基于时间和节点ID（MAC地址）的UUID。它通常包含一个时间戳和一个节点ID，因此它可以被认为具有一定的可追溯性。这意味着UUID1生成的标识符可能与特定的设备和时间相关联。

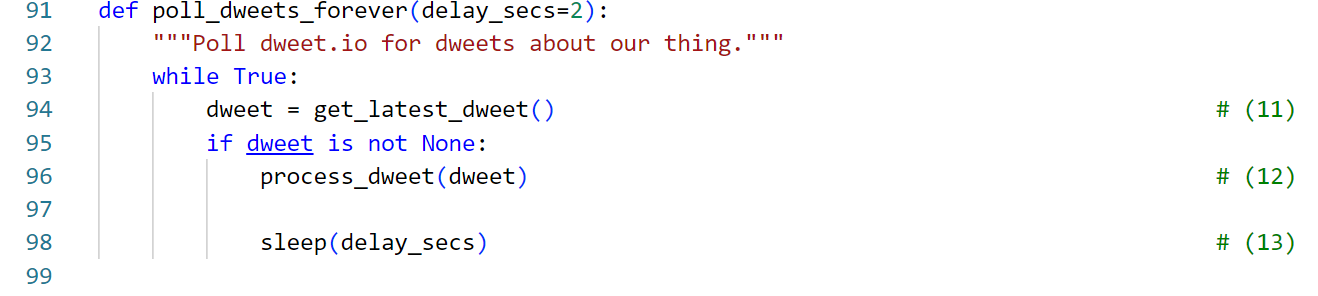
在许多应用场景中，UUID被用来唯一标识数据库记录、网络设备、软件配置等。由于它们的唯一性和不可预测性，UUID在需要确保标识符全球唯一性的场合是非常有用的。



- `get\_latest\_dweet()`：从dweet.io获取关于这个“thing”的dweet\_content。



- `poll\_dweets\_forever(delay\_secs=2)`：不断地轮询（延时两秒）dweet.io服务来检查关于led的新dweet。



- `stream\_dweets\_forever()`：无限期地监听来自dweet.io关于thing的流式dweet。

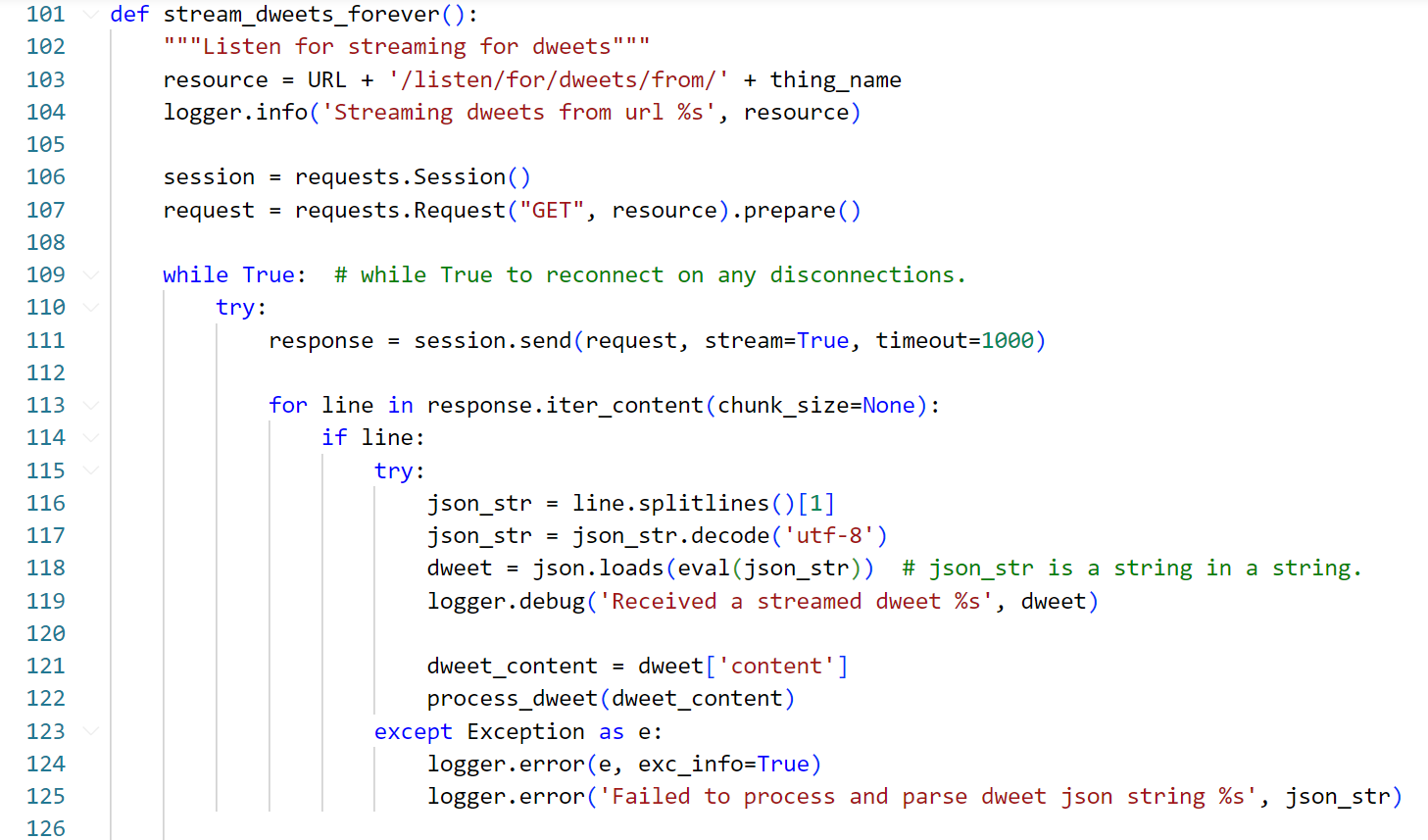
ps:

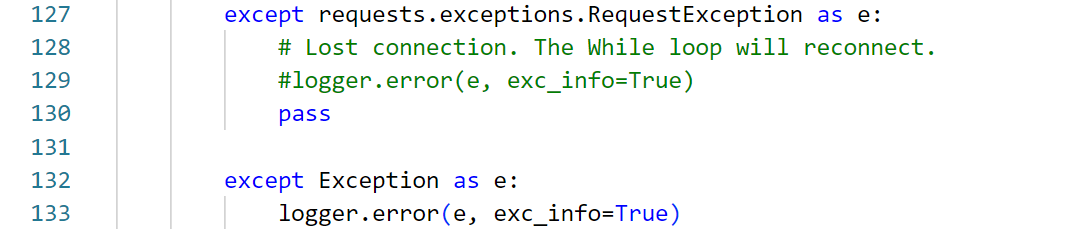
流式dweet的特点包括：

1. 实时性：数据可以立即传输和接收，适用于需要快速响应的应用。

2. 连续性：数据可以持续地传输，而不需要每次都重新建立连接。

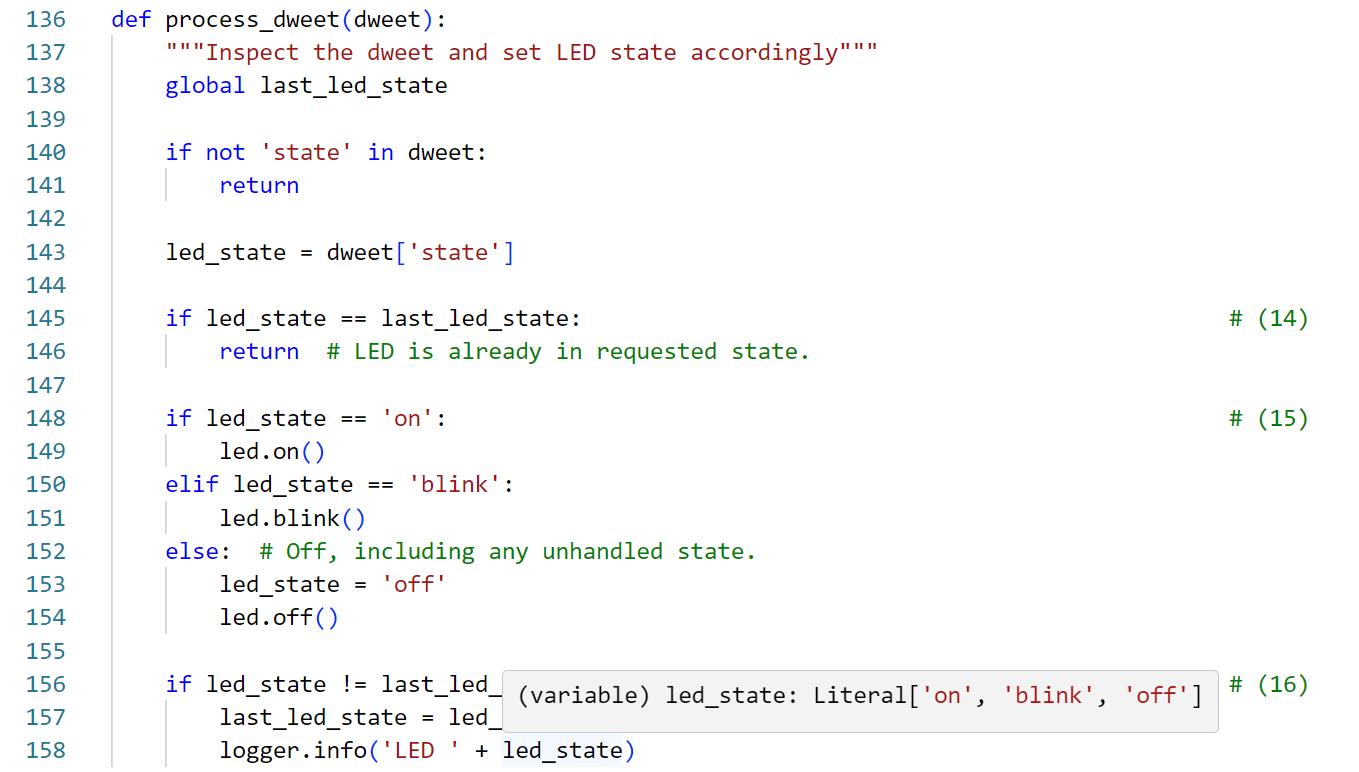
3. 轻量级：因为数据是逐个发送的，所以可以在保持较低的网络负载的同时传输大量数据。 4. 可靠性：通常会有机制来确保数据的完整性和顺序性。



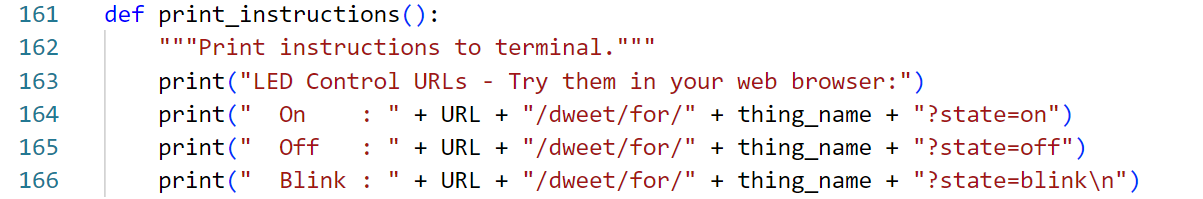


- `process\_dweet(dweet)`：处理接收到的dweet内容，设置LED的状态。

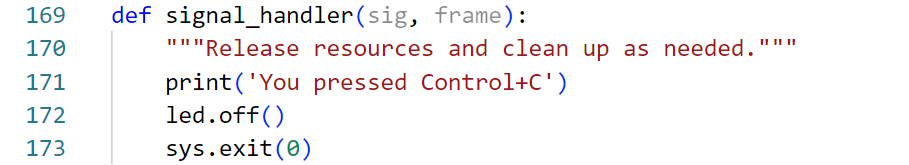
led\_state为”on”,”blink”,”off”分别对应触发led.on(),led.blink(),led.off()，实现了dweet的远程控制



- `print\_instructions()`：打印如何控制LED的说明。

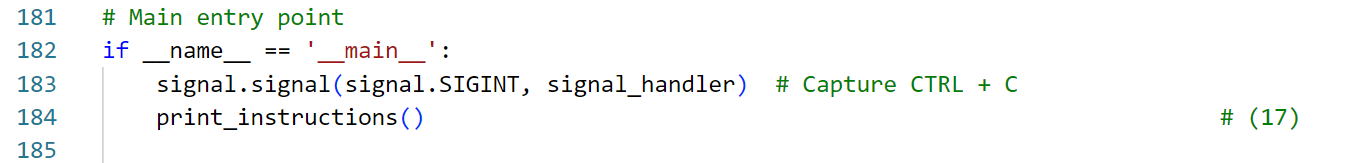


- `signal\_handler(sig, frame)`：处理退出信号，如CTRL+C，正确关闭LED。

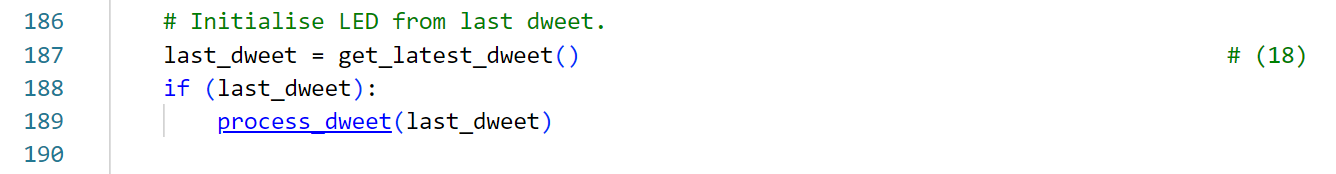


6.主程序：

- 主程序首先设置了一个信号处理器来处理退出信号(CTRL+C)。当信号处理器被触发时，LED会被关闭，程序退出。



- 接着初始化LED并根据最新的dweet调用process\_dweet()设置LED状态。



- 然后打印控制LED的说明并进入一个无限循环等待dweet的到来。 循环中可以选择实时流式dweet或定期轮询dweet（流式：stream\_dweets\_forever(),轮询：poll\_dweets\_forever()）。

