lab3：Misc

1. 必做部分 50%

1.1 Challenge 1: songmingti

450×300 的 JPEG 图片，提示"真正的 CTF 选手，就算我什么都不说，也能找出 flag 来"

"感觉后面还有东西呢"

Image 1

JPEG 以 `FF D8 FF`（SOI）开始、以 `FF D9`（EOI）结束。

在整个文件字节流中再搜索一次 `FF D8 FF`

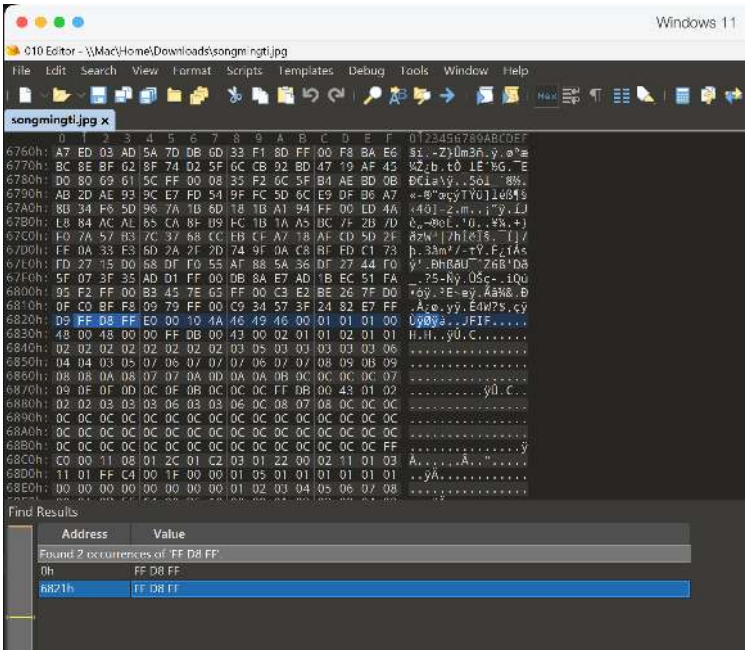

Image 2

存在套娃的内嵌 JPEG，在偏移 `0x6821` 处找到了第二个 JPEG 起始标记

从该处一直截到下一个 `FF D9`，得到一张 450×300 的内层图片。

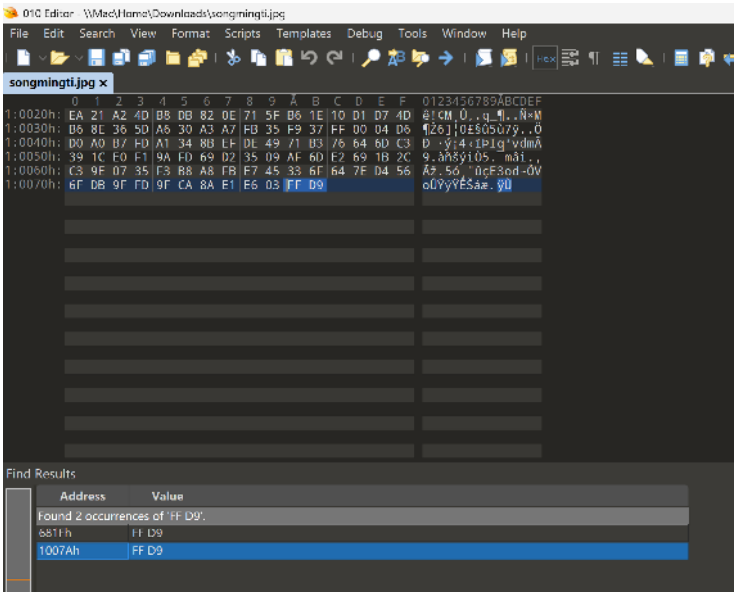

Image 3

```
data = open("1 songmingti.jpg", "rb").read()
start = data.find(b"\xff\xd8\xff", 1)    # 内嵌 JPEG 起始
end   = data.find(b"\xff\xd9", start) + 2
open("1 songmingti_extracted.jpg", "wb").write(data[start:end])
```

Image 4

AAA{the_true_fans_fans_nmb_-1s!}

1.2 Challenge 2: Little "Sister" Bitch

一张 1653×1653 的 PNG，曲名是 t+pazolite - Little "Sister" Bitch。

- 首字母很重要啊！
- 虽然确实也是用那个原理隐写进去的，但是实现上稍微不太一样，不过也能用那个工具做就是了

Image 5

"首字母"：Little **S**ister **B**itch 的每个单词首字母是 **LSB**，因此这是 LSB 隐写题。用脚本把每个像素 R/G/B 三个通道的最低位按行取出，每 8 个 bit 合成一个字节：

```
bits = []
for y in range(h):
    for x in range(w):
        r, g, b = px[x, y]
        bits += [r & 1, g & 1, b & 1]
data = bytearray()
for i in range(0, len(bits) - 7, 8):
    v = 0
    for b in bits[i:i + 8]:
        v = (v << 1) | b
    data.append(v)
```

提取出的字节流以 `RIFF` 开头，它是一个 WAV 音频文件

偏移	字节（小端）	含义	值
0x00	52 49 46 46	RIFF 魔数	—
0x04	42 9e 0f 00	RIFF 块长度	0x0f9e42 = 1023554
…	…	…	…
0x28	1e 9e 0f 00	音频数据长度	0x0f9e1e = 1023518

Table 1

似乎是死亡华尔兹之类的变种...? 听起来有高频的、很不自然的杂音，大概是因为音频隐写

2 challenge_spectrogram.py 3, U

lab3_misc > 2 challenge_spectrogram.py > ...

```python
root = Path(__file__).parent
wav = root / "2 challenge.wav"
assets = root / "lab3_misc.assets"

w = wave.open(str(wav), "rb")
n = w.getnframes()
rate = w.getframerate()
samp = np.frombuffer(w.readframes(n), dtype=np.int16).astype(np.float64)
print(f"samples={n} rate={rate} duration={n / rate:.2f}s")

# 波形图
plt.figure(figsize=(14, 3))
plt.plot(np.arange(len(samp)) / rate, samp, linewidth=0.2)
plt.title("waveform")
plt.xlabel("time s")
plt.tight_layout()
plt.savefig(assets / "2_waveform.png", dpi=80)
plt.close()

# 频谱图 (STFT)
plt.figure(figsize=(14, 5))
plt.specgram(samp, Fs=rate, NFFT=2048, noverlap=1024, cmap="gray")
plt.title("spectrogram")
plt.xlabel("time s")
plt.ylabel("Hz")
plt.tight_layout()
plt.savefig(assets / "2_spectrogram.png", dpi=80)
plt.close()
print("saved 2_waveform.png & 2_spectrogram.png")
```

问题 3 输出 调试控制台 终端 端口

```
(.venv) BillFeng@BillFengMacBook lab3_misc % python3 '2 challenge_spectrogram.py'
samples=511759 rate=44100 duration=11.60s
saved 2_waveform.png & 2_spectrogram.png
(.venv) BillFeng@BillFengMacBook lab3_misc %
```

zsh

zsh lab3_...

Image 6

画出波形图和频谱图

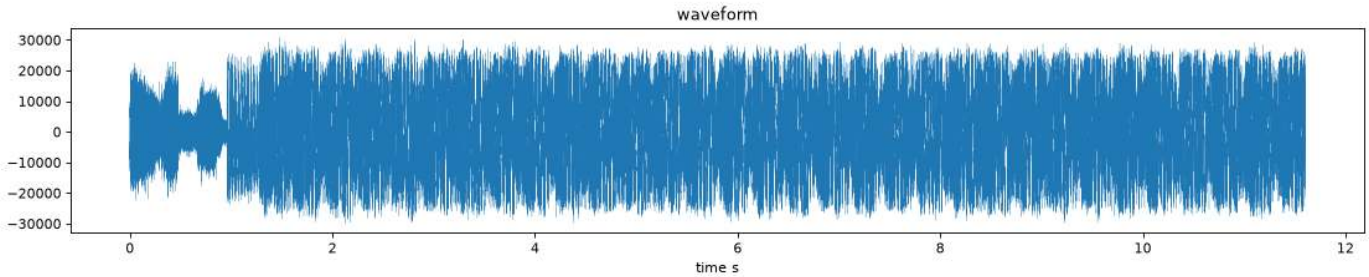

Image 7

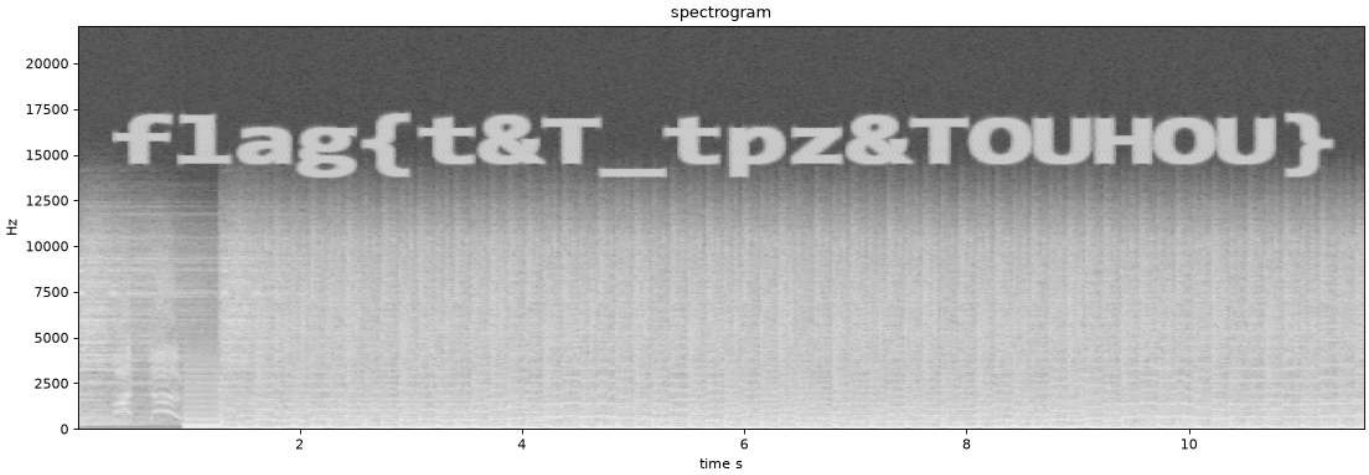

Image 8

flag藏在了频谱图`flag{t&T_tpz&TOUHOU}`

1.3 Challenge 3: 似乎是什么景区

原图：

Image 9

国内景区，采用百度识图

临沂服务区

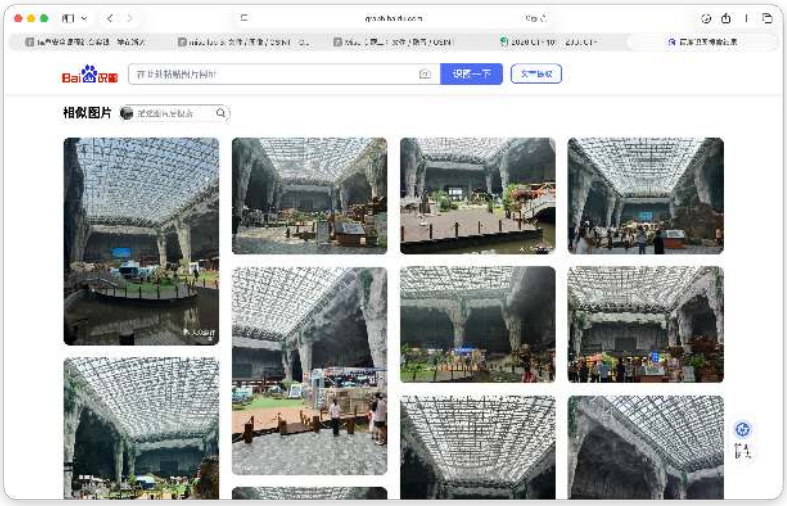

Image 10

Image 11

蓝色牌子是 **520精品集合店**，可以通过后边两个柱子的相对位置、店铺左下角右下角两个花盆的定位点和店铺左侧的柜子判定。

Image 12

Image 13

不过找了一圈发现只有这两个蓝牌子，并且有同时入镜的照片，也不必分析了。

Image 14

看不太清楚，迭代继续搜索

Image 15

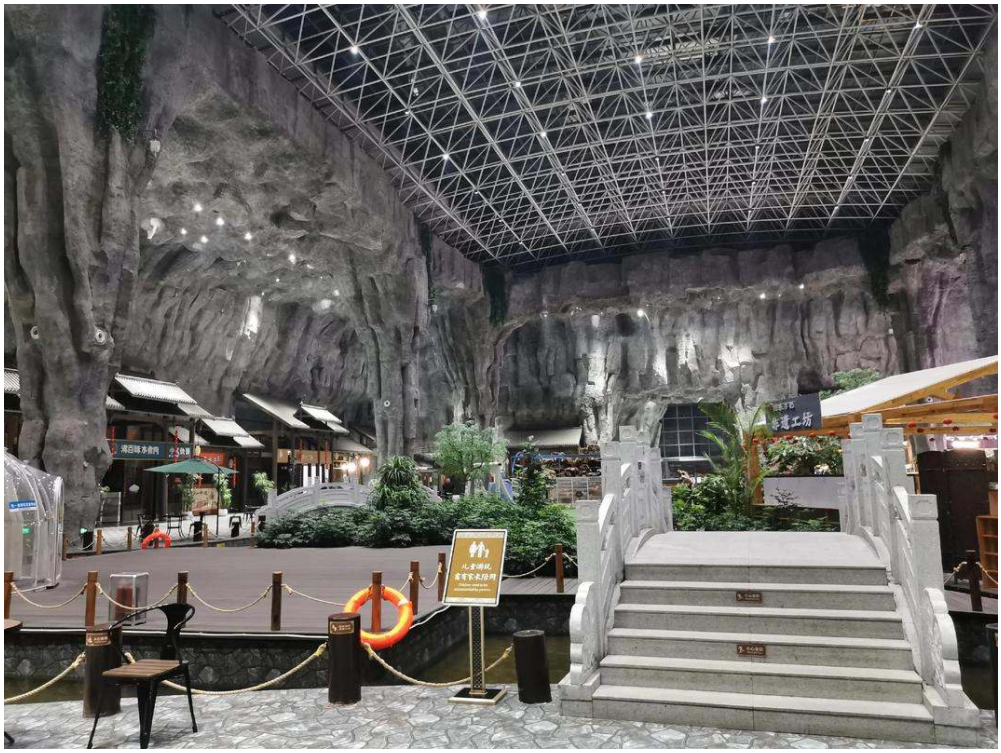

Image 16

儿童游玩需有家长陪同

这个真搜了有一段时间，太多超低清图片了。

flag{520精品集合店_儿童游玩需有家长陪同}

2. 选做部分 最高 65%

2.1 Challenge A: Palette Stego

题目给了一张调色板模式的 PNG（800×369，8-bit，颜色类型 3）。调色板模式的图片里，PLTE chunk 存放的是颜色表（256 个 RGB 颜色），而 IDAT 里每个像素并不存 RGB，只存一个 1 字节的"索引"，指向 PLTE 中的某个颜色。

这道题用的是 **EZStego** 隐写。它针对调色板类图像。先把调色板按亮度排序，然后把要藏的数据比特写进每个像素索引的 LSB。因为调色板按亮度排好序后，相邻索引对应的颜色亮度很接近，所以即便索引被改成相邻值（±1），画面也几乎看不出变化，隐写很隐蔽。

解码，把每个像素的索引取出来，读它的最低位，再把 8 个 bit 拼成一个字节即可。

具体做法是先按 PNG 规范解析出各个 chunk——IHDR（图像头，记录宽、高、位深、颜色类型）、PLTE（调色板，256 个 RGB 颜色）、IDAT（压缩后的像素数据）、IEND（文件结束标记）。然后对索引逐个取 LSB 拼字节即可。

```
# 1) 解析 PNG chunk (IHDR/PLTE/IDAT/IEND)
pos = 8
while pos < len(data):
    ln = struct.unpack(">I", data[pos:pos+4])[0]
    typ = data[pos+4:pos+8].decode()
    chunks[typ] = chunks.get(typ, b"") + data[pos+8:pos+8+ln]
    pos += 12 + ln

# 2) 解压 IDAT, 每行开头跳过 1 字节 filter
raw = zlib.decompress(chunks["IDAT"])
pixels = b"".join(raw[i*(w+1)+1:(i+1)*(w+1)] for i in range(h))

# 3) 取索引 LSB
bits = [p & 1 for p in pixels]
flag = bytes(int("".join(map(str, bits[i:i+8])), 2)
             for i in range(0, len(bits), 8))
```

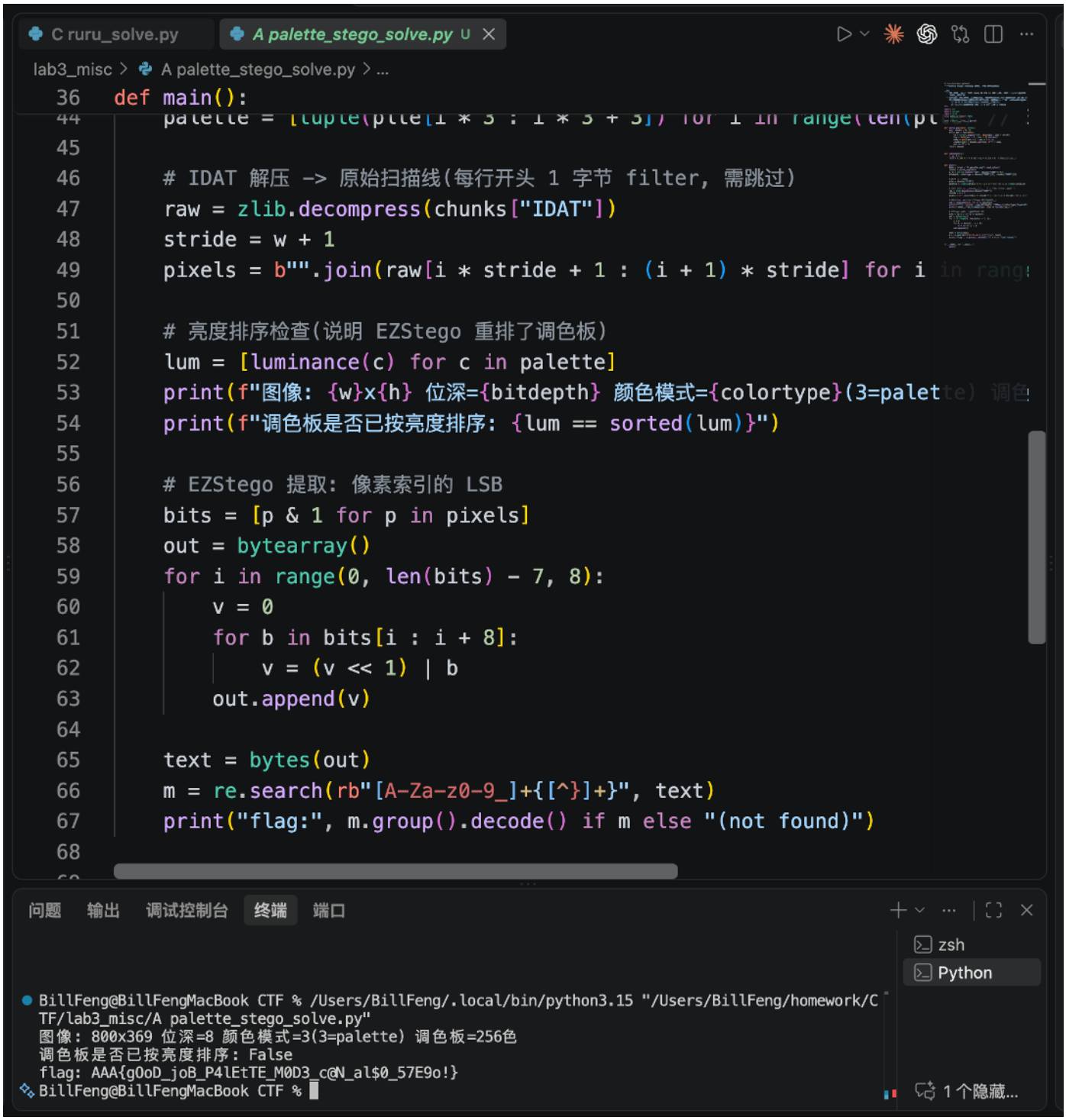

Image 17

2.2 Challenge B: Time & Power

SJTUCTF 2025 的一道题。附件 `B data.npz` 里有 `input`（1053 个字符）、`input_id`（组号 0-26）、`power`（1053 条 × 100 点功耗轨迹）。

这是一道侧信道能量分析的题。设备对密码是逐位检查的：输入的字符和某一位密码对上了，设备才会继续执行这一位的处理，功耗轨迹上就多出一个"谷"；对不上就直接跳过。

> 为什么会引起功耗变化?
> 你可以把功耗想象成时间轴上的曲线：设备处理一个字符时，先做共同的操作（所有字符都一样），只有匹配成功时才会多做一段操作。这段额外操作可能是一个比较耗电的步骤（寄存器大量翻转），也可能是进入下一阶段的空闲等待——取决于设备实现。

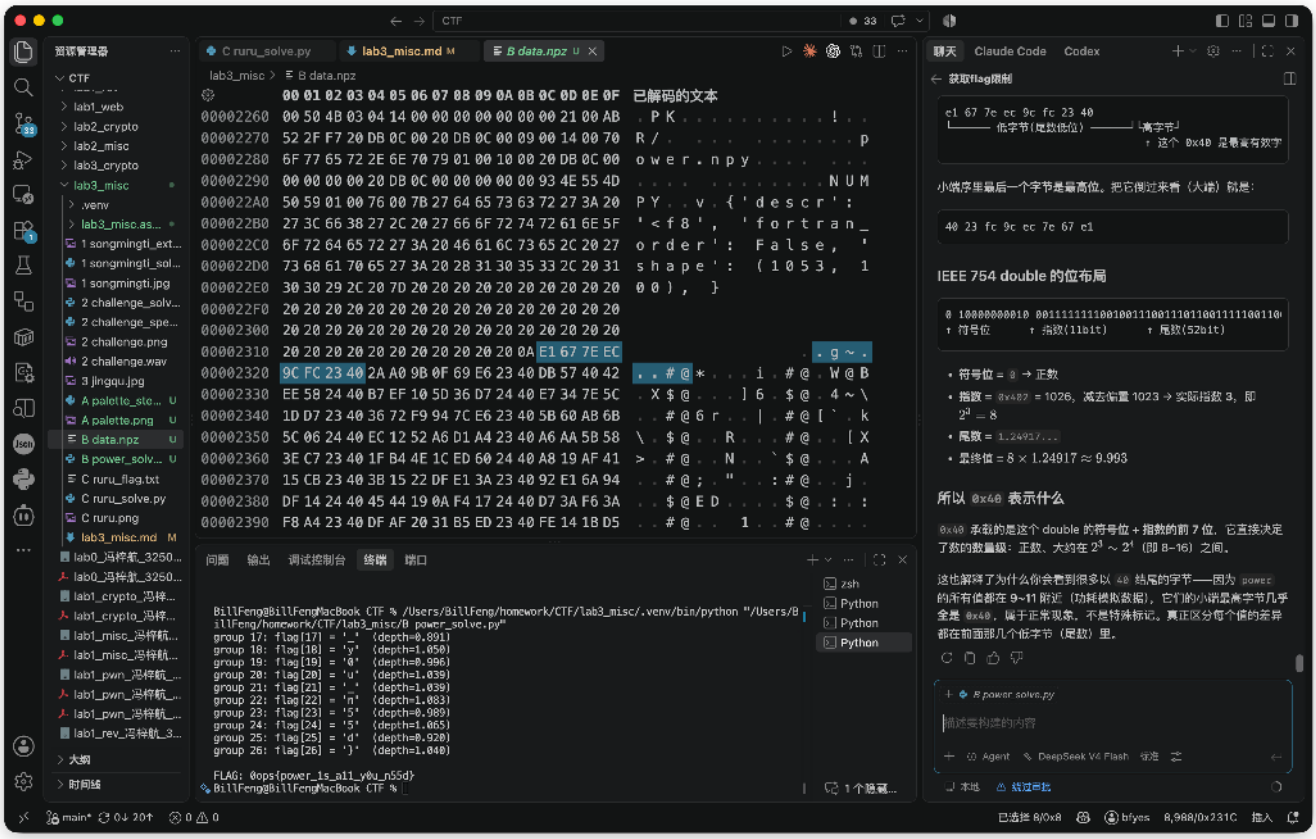

Image 18

因此把 27 组（`input_id` 0-26）看成 flag 的 27 位，每组里遍历完整字母表（39 种字符 `a-z 0-9 _ { }`）各测一条功耗轨迹，那么正确的那个字符，它的轨迹在某个采样点明显是"谷"，其余字符在那个点是"峰"。

所以对每组，只要算出每个候选字符相对其余 38 个字符均值的最大"谷深度"，谷深度最大的就是这一位的 flag 字符：

```
CHARSET = "abcdefghijklmnopqrstuvwxyz0123456789_{}"
flag = ""
for g in range(27):
    M = power[g*39:(g+1)*39]          # 39 个候选字符的功耗轨迹
    depth = np.zeros(39)
    for c in range(39):
        others = (M.sum(axis=0) - M[c]) / 38
        depth[c] = np.max(others - M[c])    # 该字符的最大谷深度
    flag += CHARSET[np.argmax(depth)]
```

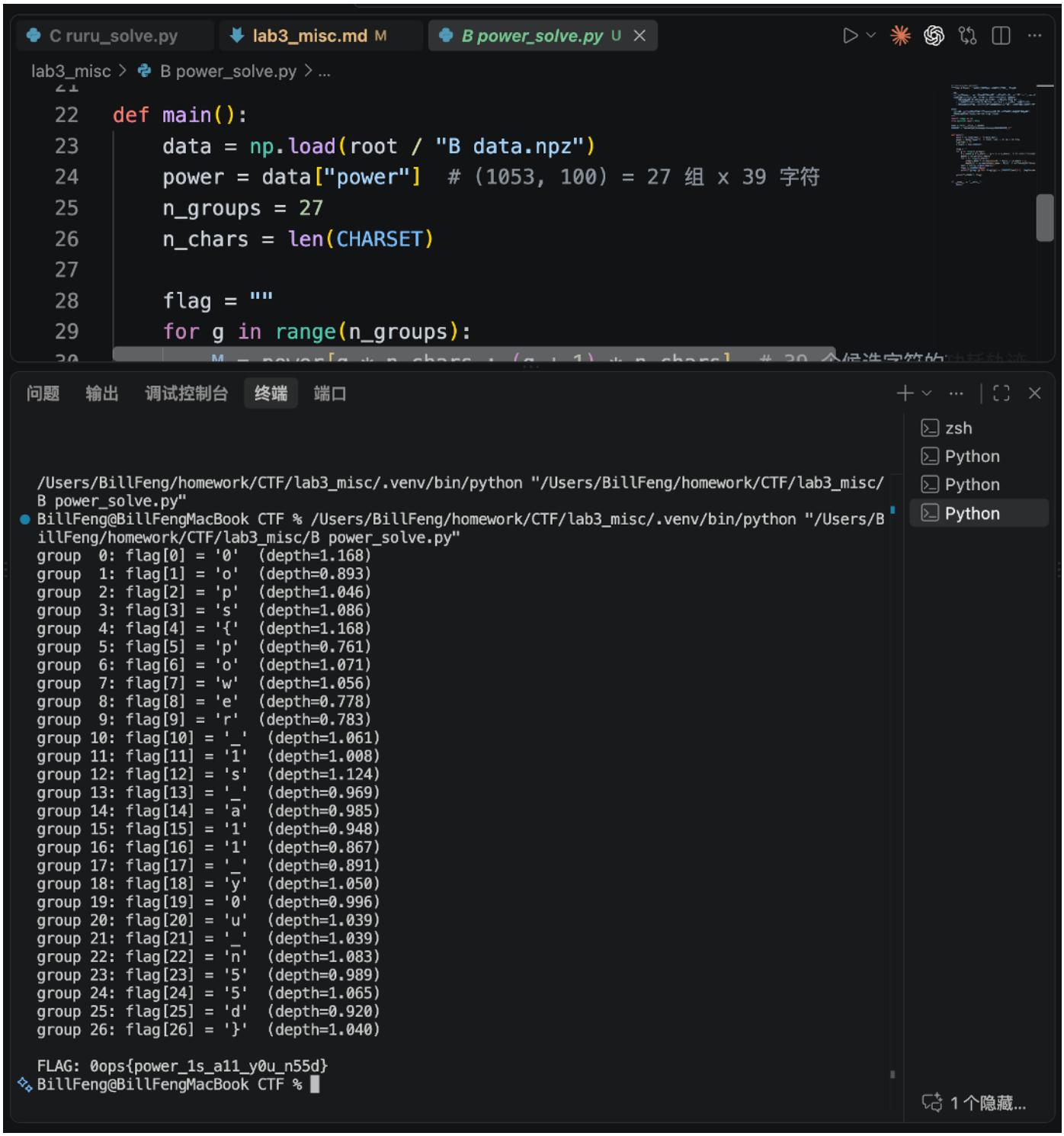

Image 19

`0ops{power_1s_a11_y0u_n55d}`

2.3 Challenge C: RURU

`C ruru.png` 本身是一张调色板 PNG，但在它的字节流里按 zip 签名 `PK\x03\x04`（local header）和 `PK\x05\x06`（EOCD）能切出一个内嵌的 zip。

2.3.1 千层面

内嵌 zip 解开后得到 `999_0061c023.zip`，嵌套伪加密。

```
999_0061c023.zip -> 998_00bb42a0.zip -> 997_7e1b2bb5.zip -> ... -> 001_d7f1ca16.zip
   -> The_password_is_an_8_character_hex_string.zip -> ruru.txt
```

忽略 CRC，逐层剥开

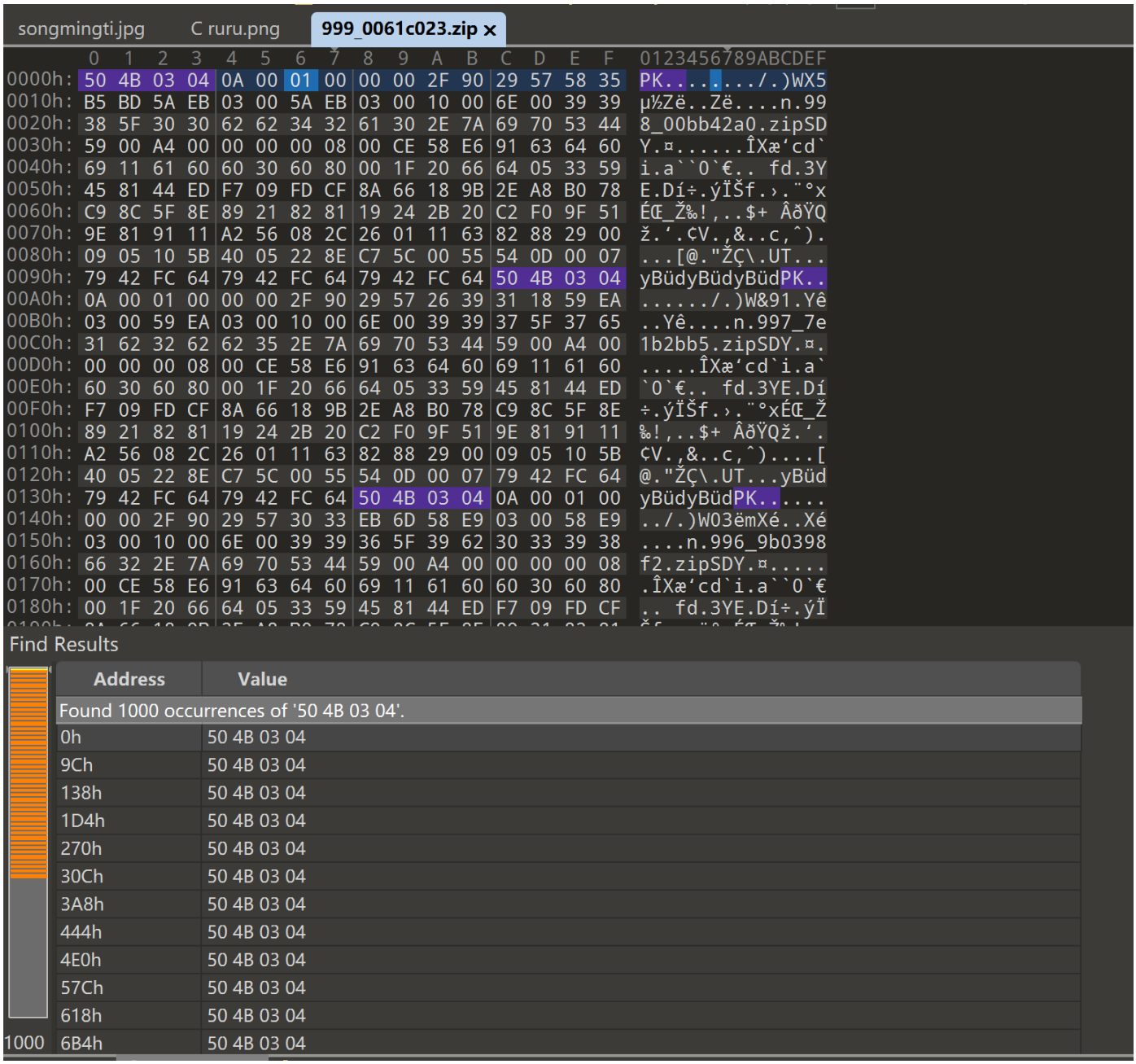

Image 20

```python
def unpeel(data):
    p = data.find(b"PK\x03\x04")
    method = struct.unpack("<H", data[p + 8 : p + 10])[0]
    csize  = struct.unpack("<I", data[p + 18 : p + 22])[0]
    nlen   = struct.unpack("<H", data[p + 26 : p + 28])[0]
    elen   = struct.unpack("<H", data[p + 28 : p + 30])[0]
    name   = data[p + 30 : p + 30 + nlen]
    raw    = data[p + 30 + nlen + elen : p + 30 + nlen + elen + csize]
    return name, raw if method == 0 else zlib.decompress(raw, -zlib.MAX_WBITS)
```

到最里层

```
L1000  The_password_is_an_8_character_hex_string.zip  314 bytes
L1001  ruru.txt                                        57 bytes
```

2.3.2 爆破

`The_password...zip` 里的 `ruru.txt` 是真加密，文件名提示密码是 8 字符 hex 字符串。

用 john 的 mask 模式爆破

```
zip2john xxx.zip > hash.txt
john --format=PKZIP --mask='?h?h?h?h?h?h?h?h' hash.txt
```

john 爆破得到密码 `fa1ab417`，用它解密 `ruru.txt`

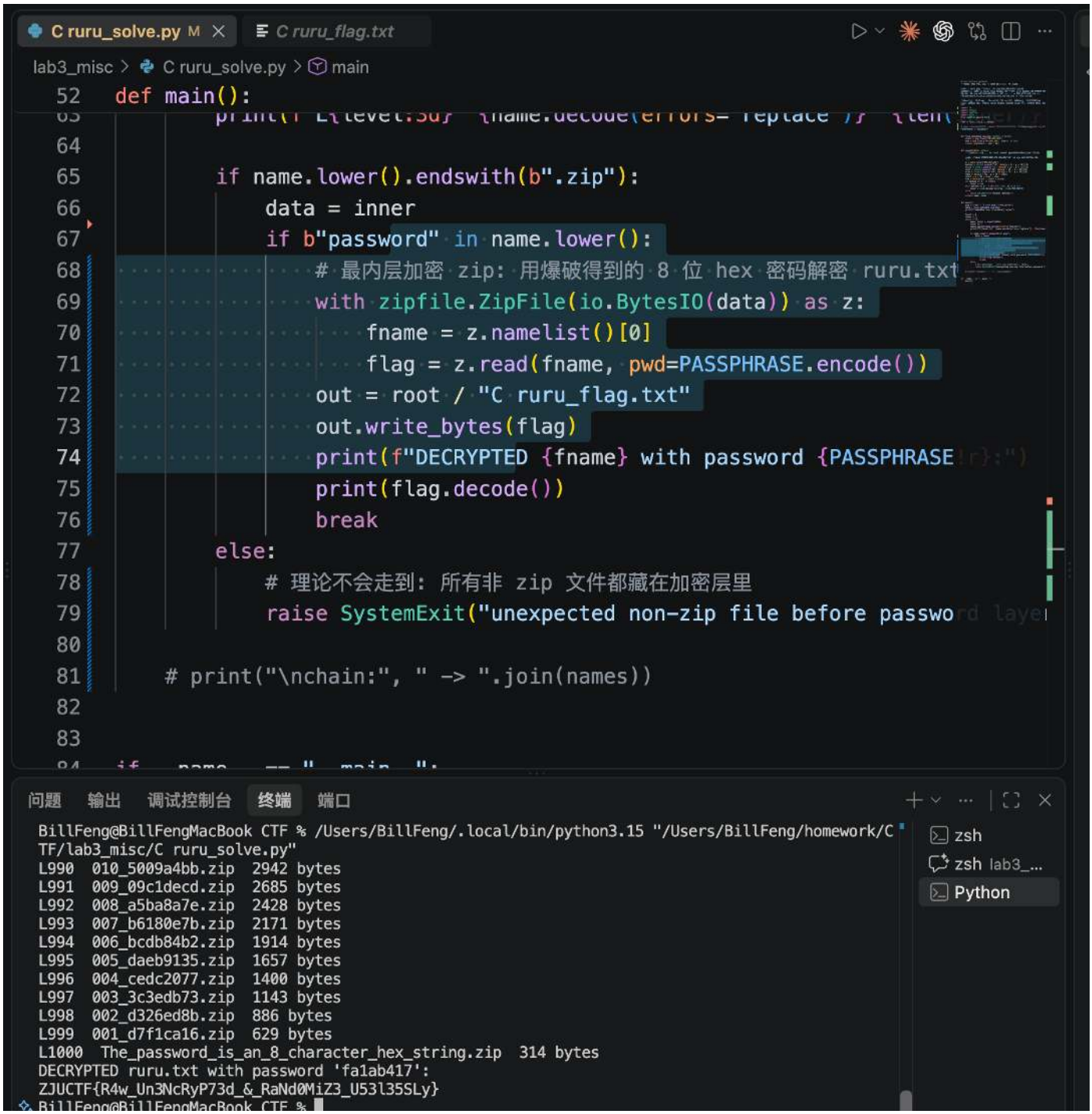

Image 21