## Simple Math

Ask ChatGPT to translate python bytecode given (sorry I haven't learned how to read it, and it just worked so) The code translates to this:

```
def conv(s, l):
        for i in range(0, len(s), l):
            yield s[i : i + l]

    flag = open("flag.txt").read()
    assert len(flag) % 5 == 0

    N = [
        412881107802,
        397653008560,
        378475773842,
        412107467700,
        410815948500,
        424198405792,
        379554633200,
        404975010927,
        419449858501,
        383875726561,
    ]

    NR = list(reversed(N))
    flags = []

    for i, j, k in zip(conv(flag, 5), N, NR):
        x = int.from_bytes(i.encode(), "big")
        y = ((x + j) * 1337) ^ k
        y -= 871366131
        flags.append(y)

    print(flags)
```

Now that we have the source code, reversing it is as simple as reversing all the operations.

```
outputs = [
    927365724618649,
    855544946535839,
    1075456339888851,
    1051300489856216,
    854566738228717,
    862564607600557,
    1107196607637040,
    835104762026329,
    1108826984434051,
    843310935687105,
]

N = [
    412881107802,
    397653008560,
    378475773842,
```

```
    412107467700,
    410815948500,
    424198405792,
    379554633200,
    404975010927,
    419449858501,
    383875726561,
]

NR = list(reversed(N))

flag = b""
for output, j, k in zip(outputs, N, NR):
    yy = output + 871366131
    y = yy ^ k
    x = y // 1337 - j
    flag += x.to_bytes(5, "big")

print(flag)
```

This results in `b'ARA6{8yT3_c0d3_W1Th_51MPl3_m4th_15_345Y____R19ht?}'`

## What Shark?

Read the scap file with wireshark, then filter all HTTP request and responses. This will show a log of communication, first registering, then logging in, pinging twice, then updating what seems like a profile picture. The key is that since HOST is local (192.168.xxx.xxx), it leads to the thought that maybe the sent image might be what's needed. Inspect the data of that last request, and match the beginning and the end with the beginning and the end of a typical hexdump of a png file. Saving that part of data to a standalone png file results and opening it shows a white paper containing the flag.

## Readable

This gives a png file that has malformed metadata. First compare it to a typical example png file.

```
xxd chall.png | head -n 5
00000000: 0000 011b 0000 01e4 0806 0000 000e e98b  ................
00000010: ed00 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
00000020: 0004 6741 4d41 0000 b18f 0bfc 6105 0000  ..gAMA......a...
00000030: 0009 7048 5973 0000 0ec1 0000 0ec1 01b8  ..pHYs..........
00000040: 916b ed00 00ff a549 4441 5478 5eec fdeb  .k.....IDATx^...

xxd chall.png | tail -n 5
00025ea0: 4633 1bdf a090 ecb3 669d 2f64 3f91 d937  F3......f./d?..7
00025eb0: fd9a 1e92 7504 e8ef f277 6101 7d35 faeb  ....u....wa.}5..
00025ec0: 24a0 dff7 f5e8 b359 4372 b6b7 ff1f 28cb  $......YCr....(.
00025ed0: 79a5 314e c367 0000 0000 4945 4e44 ae42  y.1N.g....IEND.B
00025ee0: 6082

xxd example.png | head -n 5
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
00000010: 0000 0352 0000 0236 0802 0000 00d5 562e  ...R...6......V.
00000020: a800 000a 4369 4343 5049 4343 2070 726f  ....CiCCPICC pro
00000030: 6669 6c65 0000 78da 9d53 7758 93f7 163e  file..x..SwX...>
```

```
00000040: dff7 650f 5642 d8f0 b197 6c81 0022 23ac  ..e.VB....l.."#.

xxd example.png | tail -n 5
0007d210: 6653 0bea 1bcd 61ab 3db5 0309 ce6c 5a18  fS....a.=....lZ.
0007d220: 667d f97e 2df3 6c5a b27a 5242 e23c a748  f}.~-.lZ.zRB.<.H
0007d230: 8a20 d066 8944 baa6 4008 8494 48cf fc07  . .f.D..@...H...
0007d240: 3804 1fc5 711e 6797 0000 0000 4945 4e44  8...q.g.....IEND
0007d250: ae42 6082
```

It can be seen that although the end section is intact, the beginning is malformed. The fact that IDAT section (data section) exists further hints that only the headers are needed to be repaired. Referring to the png specification, it can be seen that the first several bytes of the png doesn't have a complete segment. This is because the 5th to 8th bits (or 4th to 7th) of a segment must be latin alphabets, which doesn't appear here. By guessing (Again, thanks to GPT), that headless segment is very likely to be the IHDT segment. Adding the png signature along with IHDT length and signature metadata to the beginning of the file: `8950 4e47 0d0a 1a0a 0000 000d 4948 4452` The file becomes a proper viewable png.

```
xxd out.png | head -n 5
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
00000010: 0000 011b 0000 01e4 0806 0000 000e e98b  ................
00000020: ed00 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
00000030: 0004 6741 4d41 0000 b18f 0bfc 6105 0000  ..gAMA......a...
00000040: 0009 7048 5973 0000 0ec1 0000 0ec1 01b8  ..pHYs..........
```

The flag can be seen in the png image containing a smiling person, showing `ARA6{PnG_5I9n4tur3_1$_3A5y_R1gh7???}`

## IDK

Pertama lihat bahwa nilai setiap list yang dibulatkan ke prima terdekat dalam flag akan dikalikan dengan selisih 2^(2*158). kita dapat mengubah nilai c menjadi bentuk biner terlebih dahulu. disana kita akan melihat gap 0 yang cukup banyak,

(Maaf masalah formatting)

0b1000001010100100100000100110110011111011011100110110000101111001011000010101111101110100011001010111001001110101

dengan mengambil membagi nilai biner yang dibagi oleh 000000... yang banyak (menjadi 8 buah) dan mengubahnya dari biner ke bytes maka akan didapatkan flag Dimana akan di sesuaikan menjadi lebih proper

akan mendapatkan: `ARA6{saya_terus_terang_ga_tahu_ini_tiba_tiba_terus_terang_saya_tidak_diberi_tahu_saya_t`