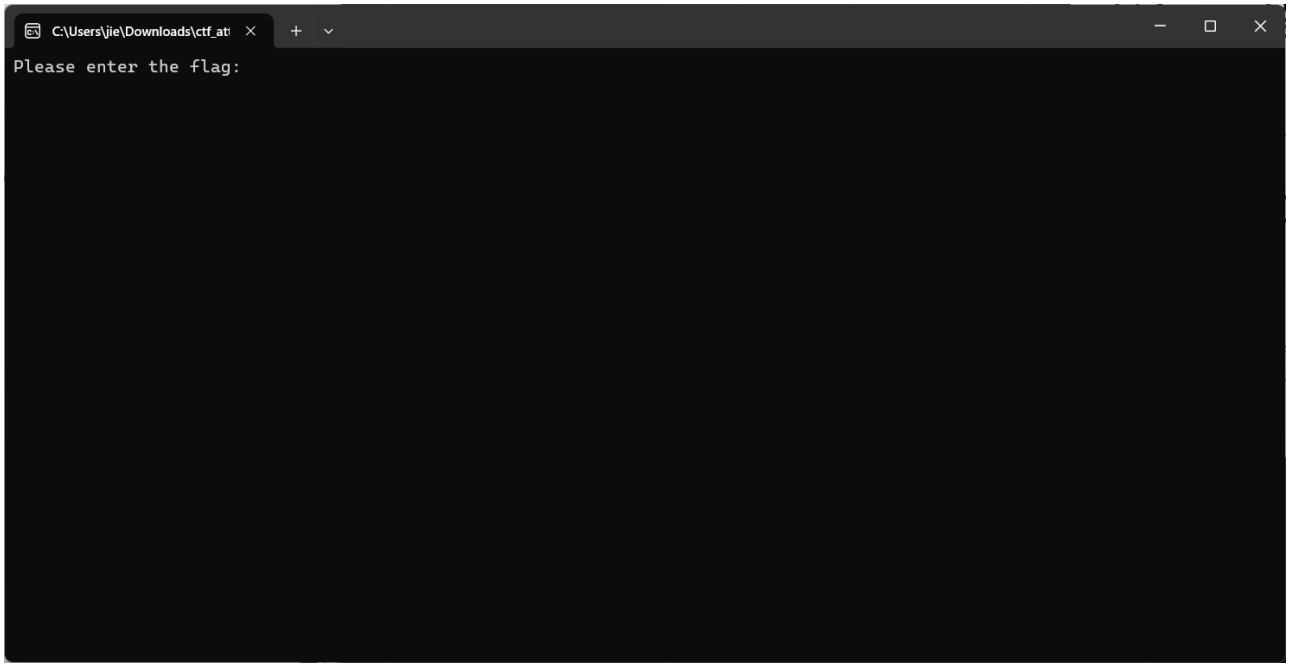
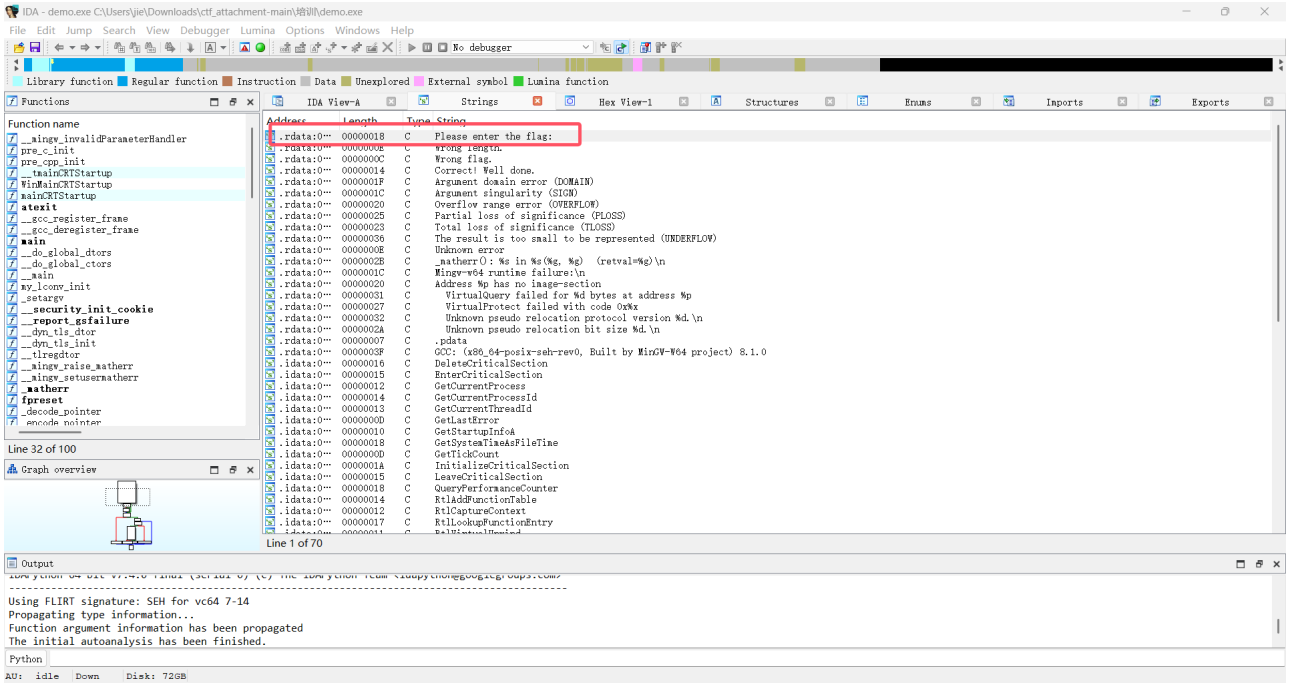
ez_base

---

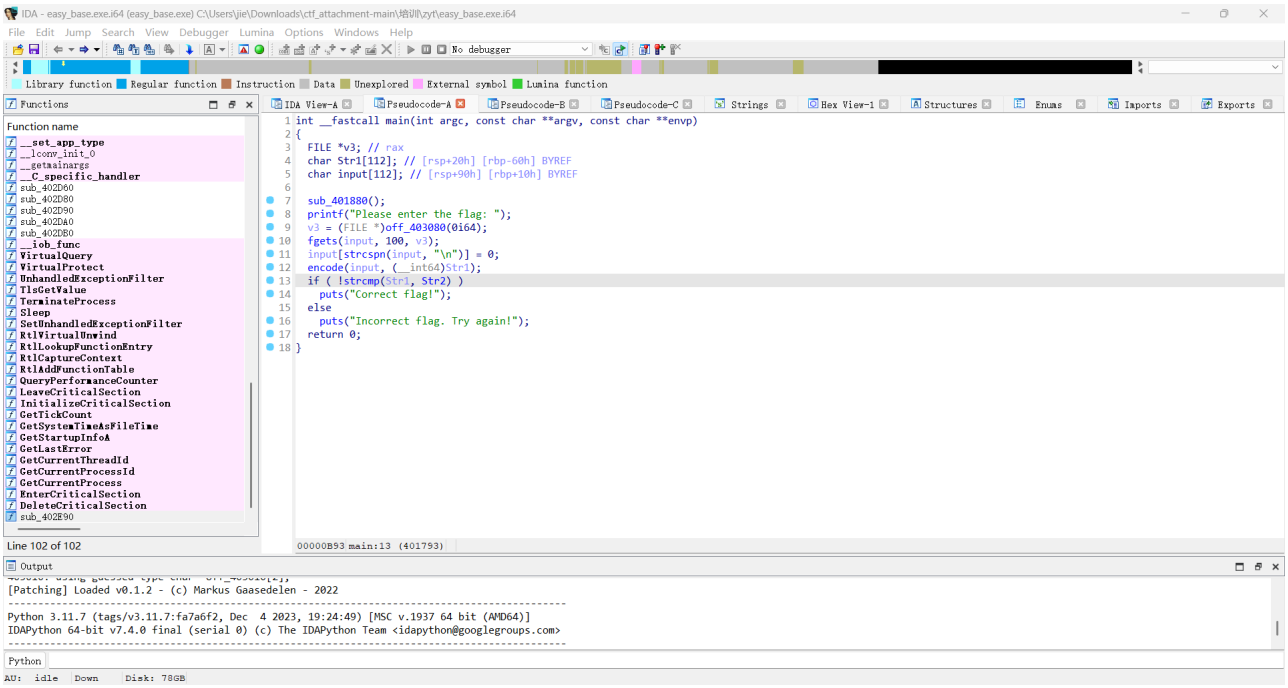
1. 首先打开该exe文件，发现要求输入flag



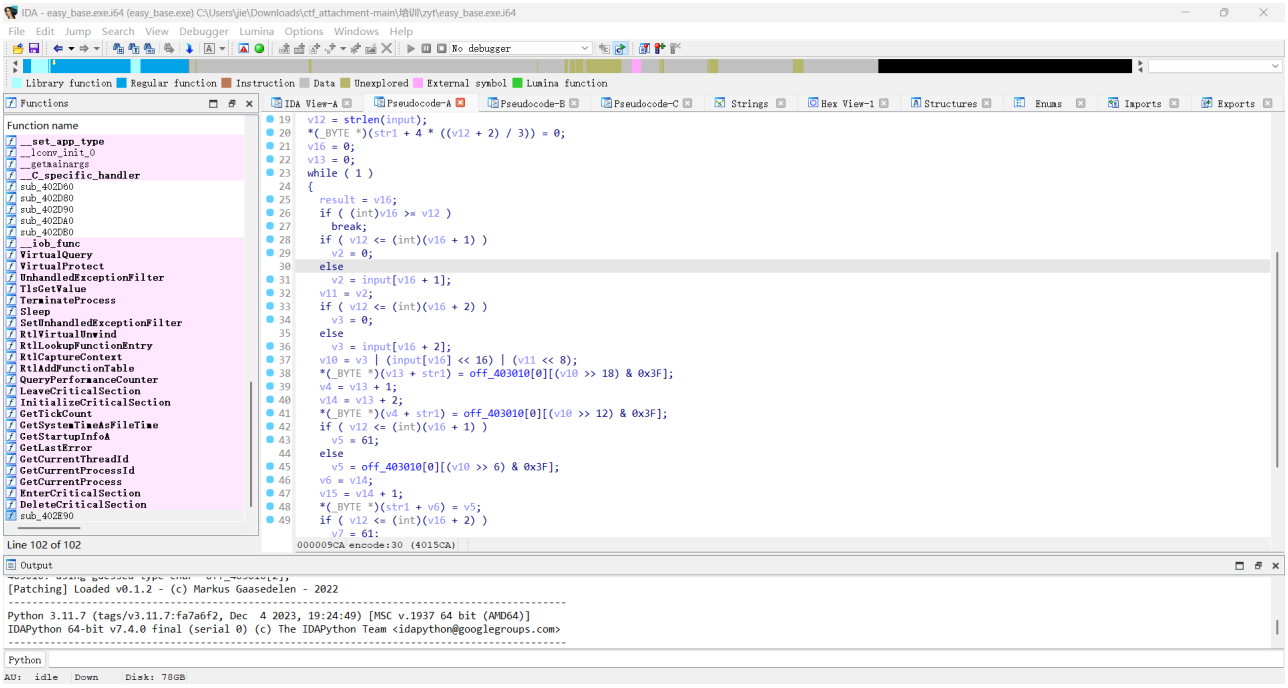
2. 使用IDA对其进行逆向分析，按下`Shift + F12`进入`String`界面，对相应字符串进行搜索



从而找到main()函数



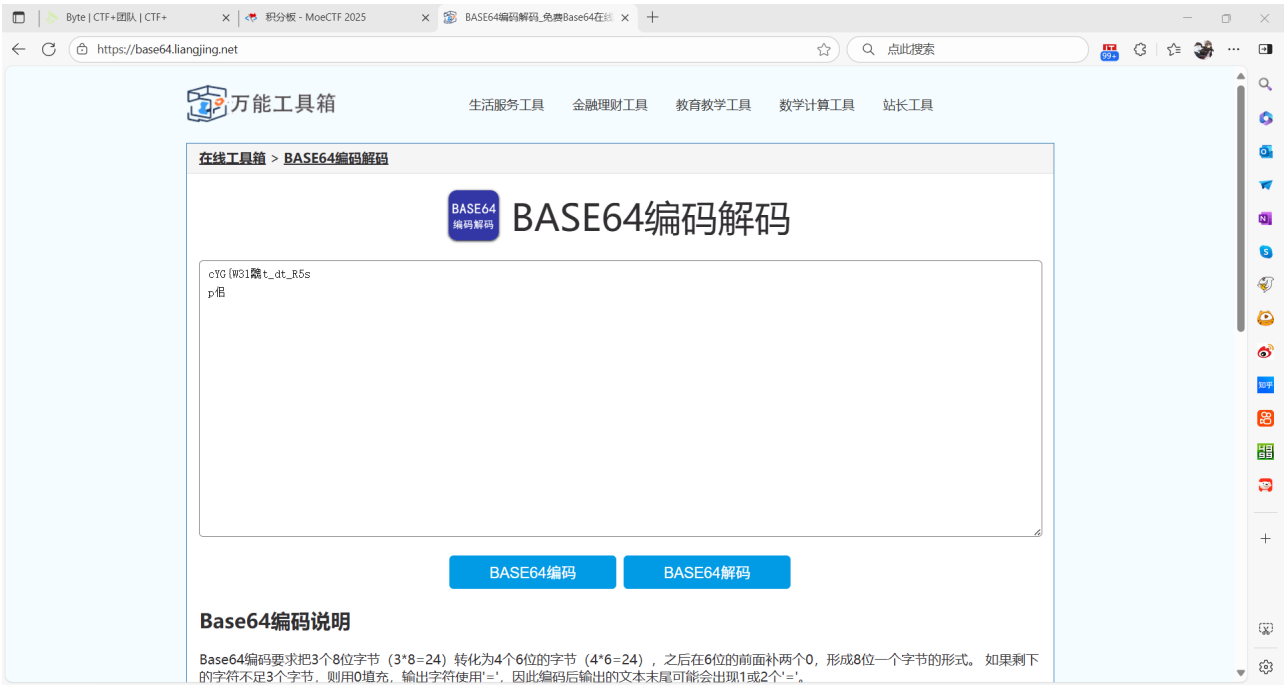
3. 对encode()函数进行分析，发现其结构符合base64编码加密过程



4. 又经过逆向分析的str2经过解码即为所求flag，经过搜索得str2值为

Y1lHe1czbKQwbXRfZHRfUjVzDXC0fU==

在网上利用在线base64解码工具进行解码得到的为乱码



5. 重新进行逆向分析发现该加密算法所遵循的编码表并非传统的base64编码，而是经过“魔改”，该代码所遵循的编码表为EFGHIJKLMNOPQRSTUVWXYZABCDabcdefghijklmnopqrstuvwxyz0123456789+/
6. 于是利用python写出脚本对其进行解码

```
def custom_base64_decode(encoded_str):
    custom_table =
"EFGHIJKLMNOPQRSTUVWXYZABCDabcdefghijklmnopqrstuvwxyz0123456789+/"

    char_to_index = {char: idx for idx, char in enumerate(custom_table)}

    padding = encoded_str.count('=')
    encoded_str = encoded_str.replace('=', '')

    bytes_data = []
    for i in range(0, len(encoded_str), 4):
        group = encoded_str[i:i+4]
        indices = [char_to_index[char] for char in group]
        while len(indices) < 4:
            indices.append(0)

        value = (indices[0] << 18) | (indices[1] << 12) | (indices[2] << 6) |
indices[3]

        byte1 = (value >> 16) & 0xFF
        byte2 = (value >> 8) & 0xFF
        byte3 = value & 0xFF

        bytes_data.append(byte1)
        if i + 2 < len(encoded_str):
            bytes_data.append(byte2)
        if i + 3 < len(encoded_str):
            bytes_data.append(byte3)
```

```
    return bytes(bytes_data).decode('utf-8', errors='replace')

if __name__ == "__main__":
    encoded_input = input("请输入使用自定义Base64编码的字符串：")
    decoded_result = custom_base64_decode(encoded_input)
    print(f"解密结果：{decoded_result}")
```

求得flag为SYC{W3lc0m3_T3_B4se64},输入程序后发现答案正确