

nc로 서버에 접속하면 인코딩된 리스트 값들이 나온다.

```
def get_flag():
    flag = open("flag.txt", 'r').read()
    flag = flag.strip()
    hex_flag = []
    for c in flag:
        hex_flag.append([str(hex(ord(c)))])

    return hex_flag

def main():
    flag = get_flag()
    print(f"flag = {flag}")
    cypher = scramble(flag)
    print(cypher)

if __name__ == '__main__':
    main()
```

코드를 분석해보면 `get_flag()` 함수로 리스트 형식의 `flag`를 가져와 `scramble` 함수의 인자로 넘겨준다.

```

def scramble(L):
    A = L
    i = 2
    print(f"Len = {len(A)}")
    while (i < len(A)):
        A[i-2] += A.pop(i-1)
        print(A)
        A[i-1].append(A[:i-2])
        # print(A)
        i += 1

```

.scramble 함수에서는 첫 번째 인덱스에 그 다음 인덱스의 값을 가져와 이어붙이는 작업을 리스트 길이만큼 반복 한다.

결국에는 flag의 앞 인덱스부터 차례대로 더미 값을 붙여 보기 어렵게 만드는 작업을 한다.

복호화 과정은 그냥 첫 번째 인덱스부터 헥스 값만 읽어 이어붙이면 된다.

최종 복호화 코드

```

# dequantum.py

def extract_hex(item):
    return [x for x in item if isinstance(x, str) and x.startswith("0x")]

def decode(scrambled):
    flag = ""
    for item in scrambled:
        for hx in extract_hex(item):
            flag += chr(int(hx, 16))

```

```
return flag

if __name__ == "__main__":
    scrambled = [인코딩 데이터]

    flag = decode(scrambled)
    print(flag)
```

picoCTF{python\_is\_weirde67fae57}