algo_auth

I like an algorithm

```
nc 110.10.147.104 15712
nc 110.10.147.109 15712
```

MIC check(SOLVED) 3.0
KingMaker
20000
aeiou 28.6
algo_auth(SOLVED) 7.0
THE Matrix 90.9
mini converter 21.3
Double? Half! 250.0
archiver 47.6
cg_casino 142.9
Maris_shop 90.9
PyProt3ct 27.8
Butterfree
god-the-reum
Rich Project

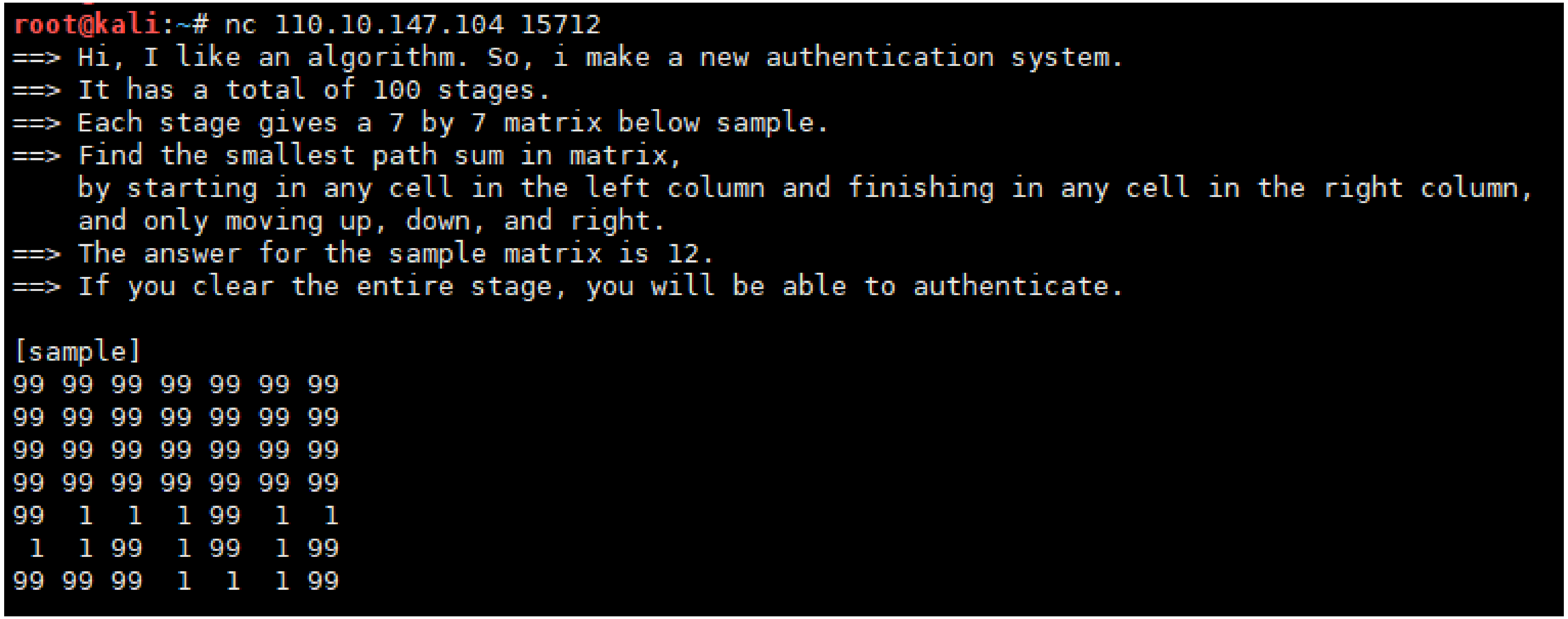
I like an algorithm

codegate 2019 alo_auth writeup

2019/01/28

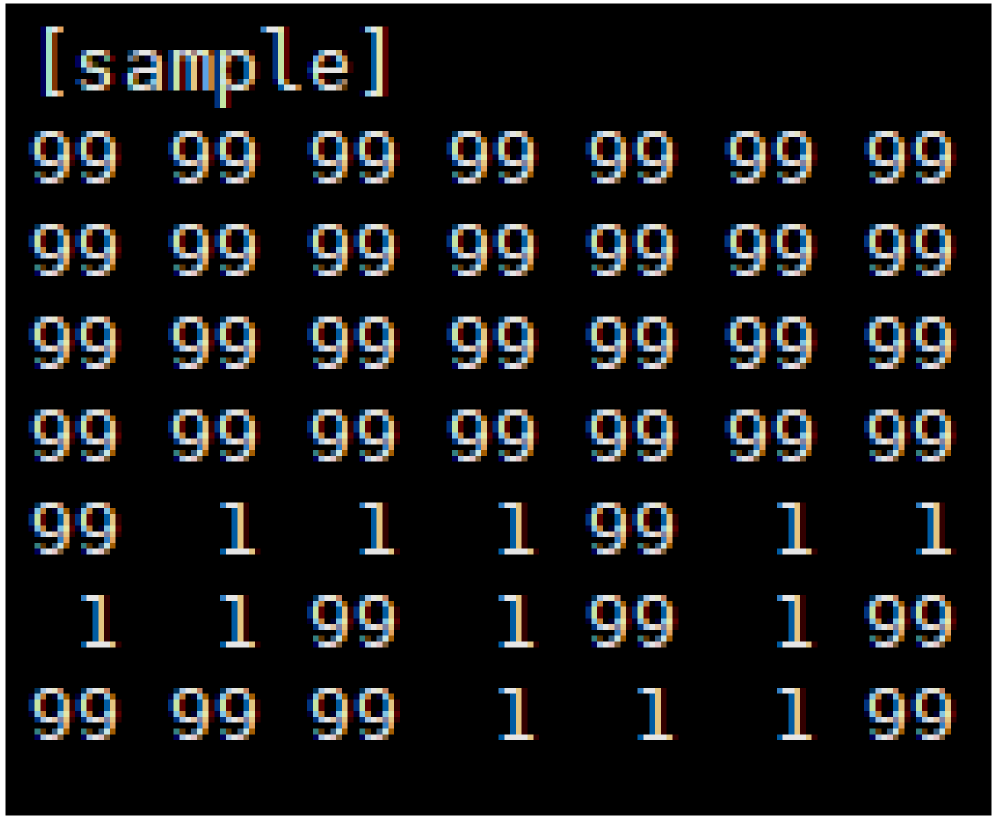
Team SCP

정재훈

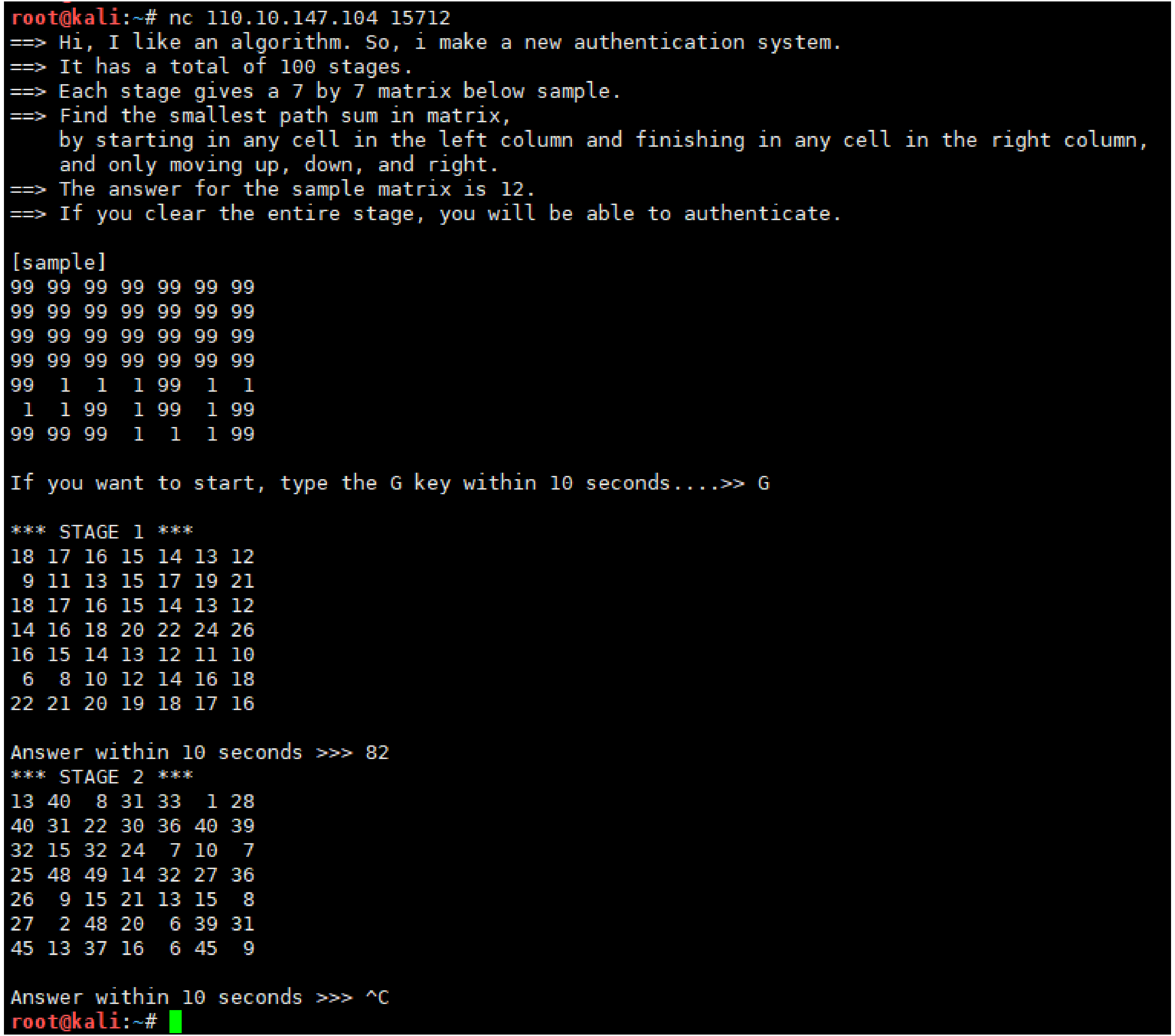


```
root@kali:~# nc 110.10.147.104 15712
==> Hi, I like an algorithm. So, i make a new authentication system.
==> It has a total of 100 stages.
==> Each stage gives a 7 by 7 matrix below sample.
==> Find the smallest path sum in matrix,
    by starting in any cell in the left column and finishing in any cell in the right column,
    and only moving up, down, and right.
==> The answer for the sample matrix is 12.
==> If you clear the entire stage, you will be able to authenticate.

[sample]
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99  1  1  1 99  1  1
 1  1 99  1 99  1 99
99 99 99  1  1  1 99
```

최소 비용 경로



```
[sample]
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99  1  1  1 99  1  1
 1  1 99  1 99  1 99
99 99 99  1  1  1 99
```



```
root@kali:~# nc 110.10.147.104 15712
==> Hi, I like an algorithm. So, i make a new authentication system.
==> It has a total of 100 stages.
==> Each stage gives a 7 by 7 matrix below sample.
==> Find the smallest path sum in matrix,
    by starting in any cell in the left column and finishing in any cell in the right column,
    and only moving up, down, and right.
==> The answer for the sample matrix is 12.
==> If you clear the entire stage, you will be able to authenticate.

[sample]
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99 99 99 99 99 99 99
99  1  1  1 99  1  1
 1  1 99  1 99  1 99
99 99 99  1  1  1 99

If you want to start, type the G key within 10 seconds....>> G

*** STAGE 1 ***
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>> 82
*** STAGE 2 ***
13 40  8 31 33  1 28
40 31 22 30 36 40 39
32 15 32 24  7 10  7
25 48 49 14 32 27 36
26  9 15 21 13 15  8
27  2 48 20  6 39 31
45 13 37 16  6 45  9

Answer within 10 seconds >>> ^C
root@kali:~#
```

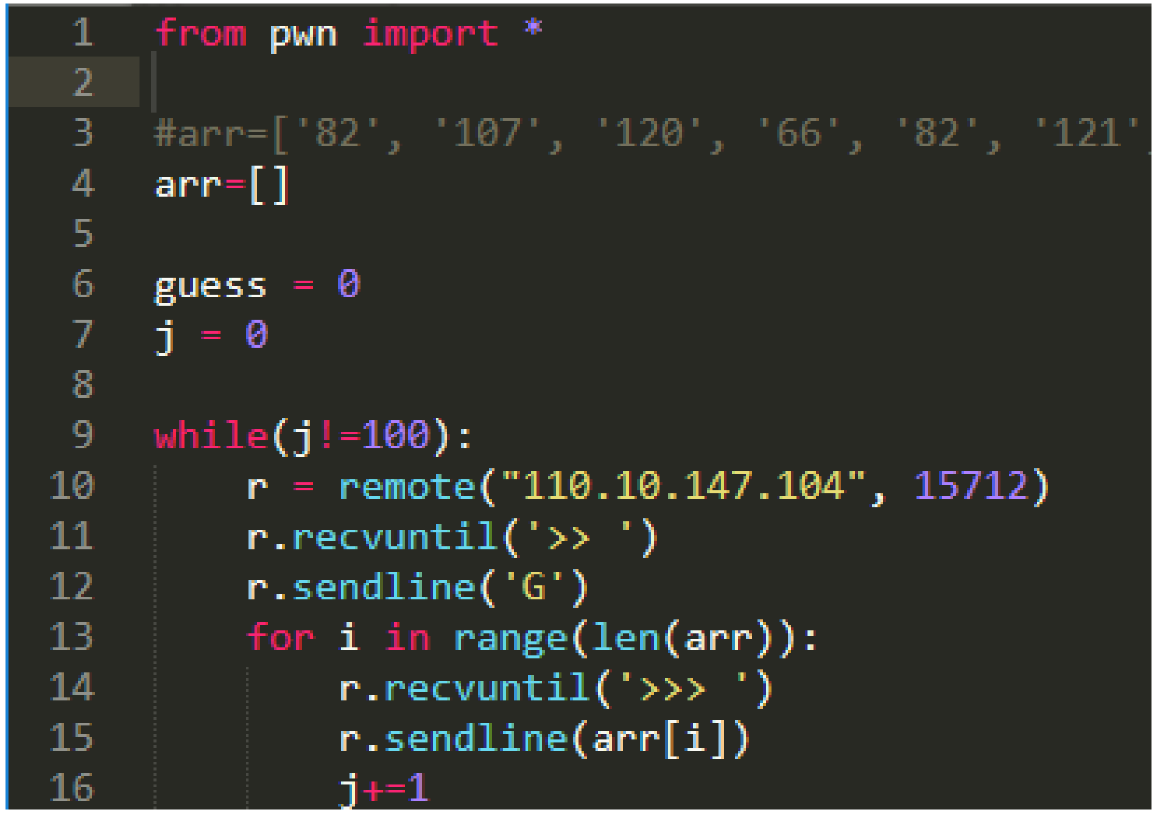
서순

→ ‘G’를 입력해 시작
→ 최적의 경로를 찾아 입력
→ 다음 스테이지

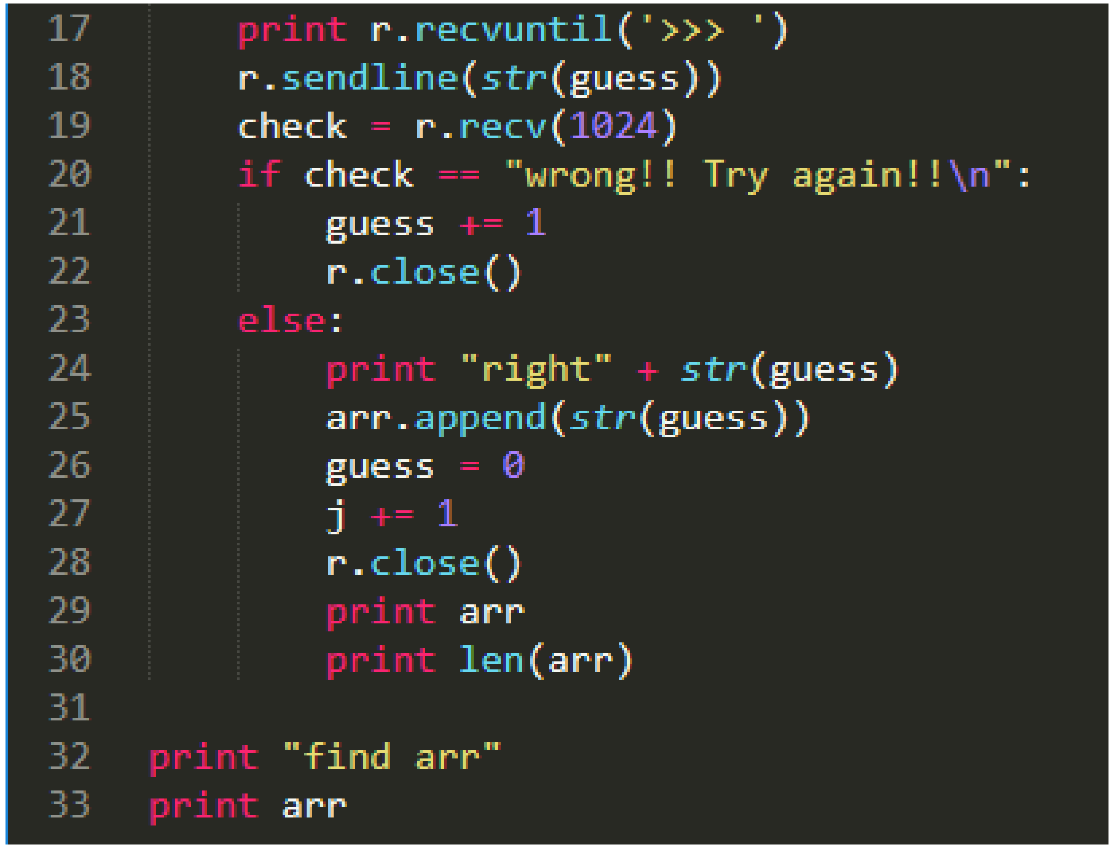
→총 100스테이지까지 있음

BruteForce Code

ex.py



```python
from pwn import *

#arr=['82', '107', '120', '66', '82', '121'
arr=[]

guess = 0
j = 0

while(j!=100):
    r = remote("110.10.147.104", 15712)
    r.recvuntil('>> ')
    r.sendline('G')
    for i in range(len(arr)):
        r.recvuntil('>>> ')
        r.sendline(arr[i])
        j+=1
```

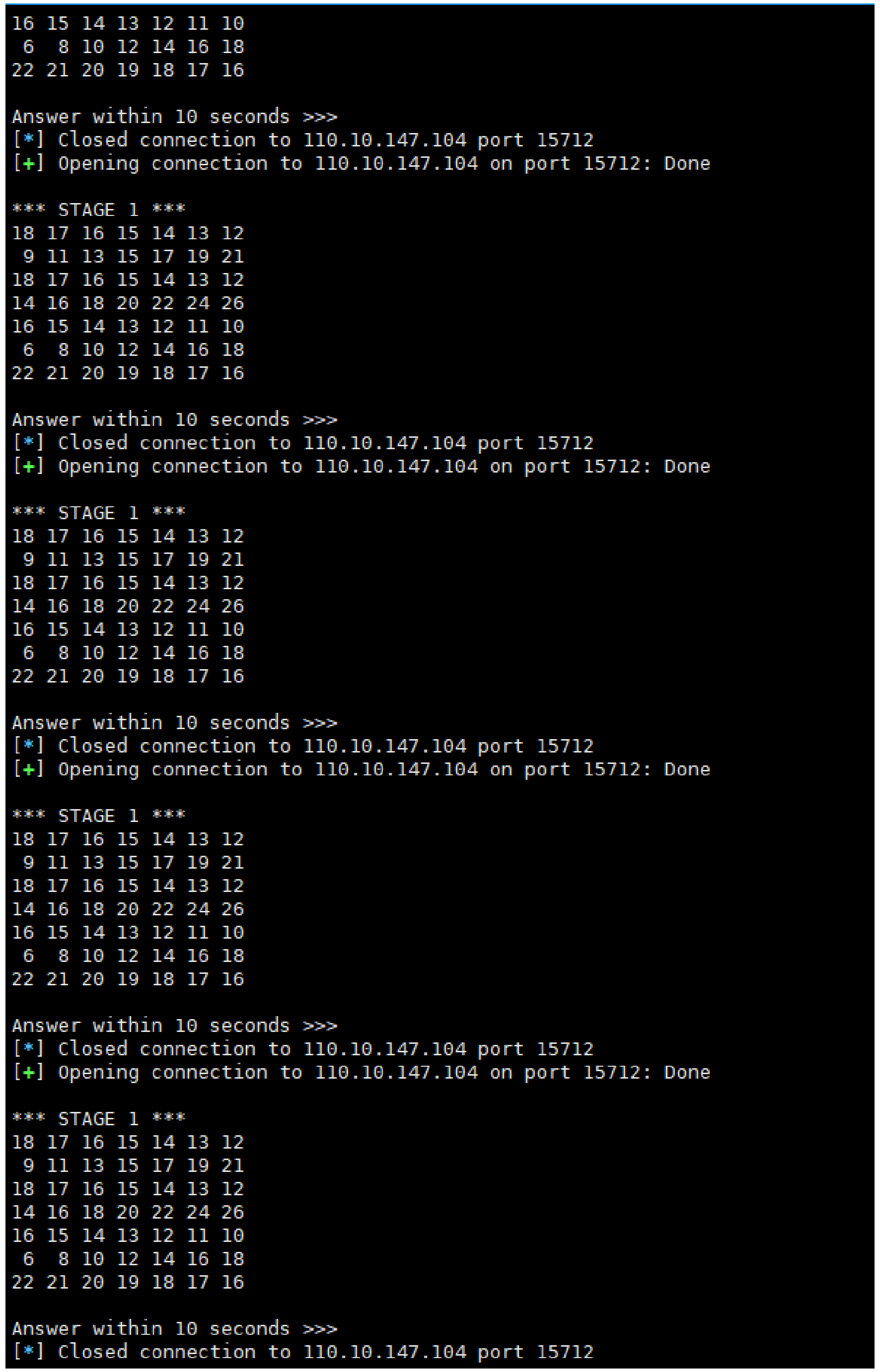


```python
    print r.recvuntil('>>> ')
    r.sendline(str(guess))
    check = r.recv(1024)
    if check == "wrong!! Try again!!\n":
        guess += 1
        r.close()
    else:
        print "right" + str(guess)
        arr.append(str(guess))
        guess = 0
        j += 1
        r.close()
        print arr
        print len(arr)

print "find arr"
print arr
```

0부터 +1 해가면서 넣음

스테이지가 넘어가면
해당 숫자 저장 후 다시 0
부터

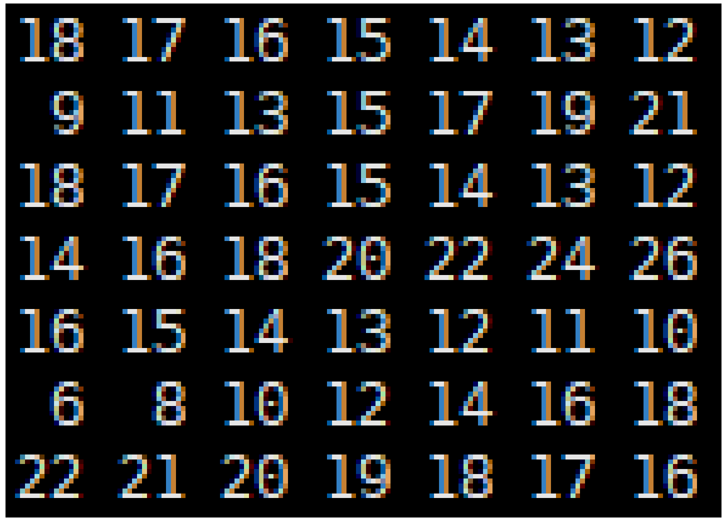
하지만......
브포만 하면 시간이 너무
오래걸림



```
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>>
[*] Closed connection to 110.10.147.104 port 15712
[+] Opening connection to 110.10.147.104 on port 15712: Done

*** STAGE 1 ***
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>>
[*] Closed connection to 110.10.147.104 port 15712
[+] Opening connection to 110.10.147.104 on port 15712: Done

*** STAGE 1 ***
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>>
[*] Closed connection to 110.10.147.104 port 15712
[+] Opening connection to 110.10.147.104 on port 15712: Done

*** STAGE 1 ***
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>>
[*] Closed connection to 110.10.147.104 port 15712
[+] Opening connection to 110.10.147.104 on port 15712: Done

*** STAGE 1 ***
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16

Answer within 10 seconds >>>
[*] Closed connection to 110.10.147.104 port 15712
```

Logic

input



```
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16
```

$\longrightarrow$

```
18+17+16+15+14+13+12  = 105
 9+11+13+15+17+19+21  = 105
18+17+16+15+14+13+12  = 105
14+16+18+20+22+24+26  = 140
16+15+14+13+12+11+10  = 91
 6+ 8+10+12+14+16+18  = 84
22+21+20+19+18+17+16  = 133
```

$\longrightarrow$ $\boxed{84}$

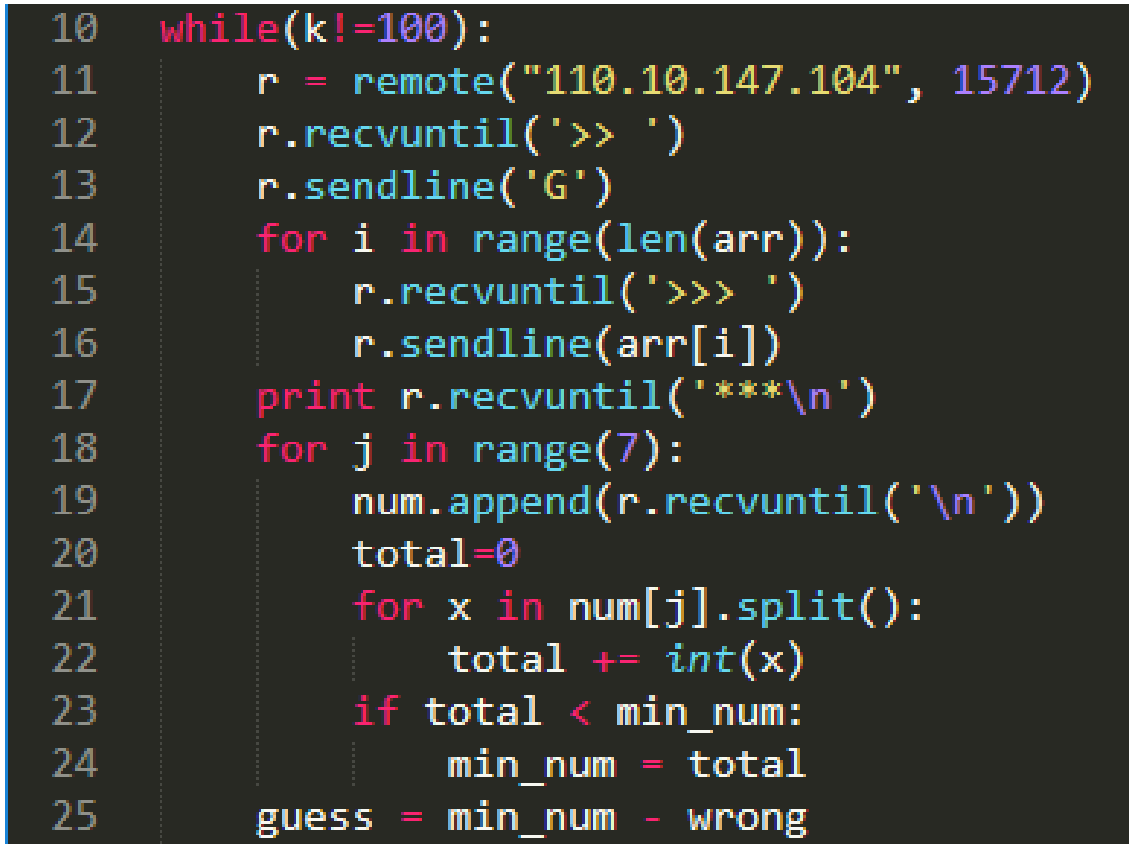
Point

- 최단 경로와 최소 비용 경로는 다름!
- 정확히는 비용이 같을 수도 있고 다를 수도 있음

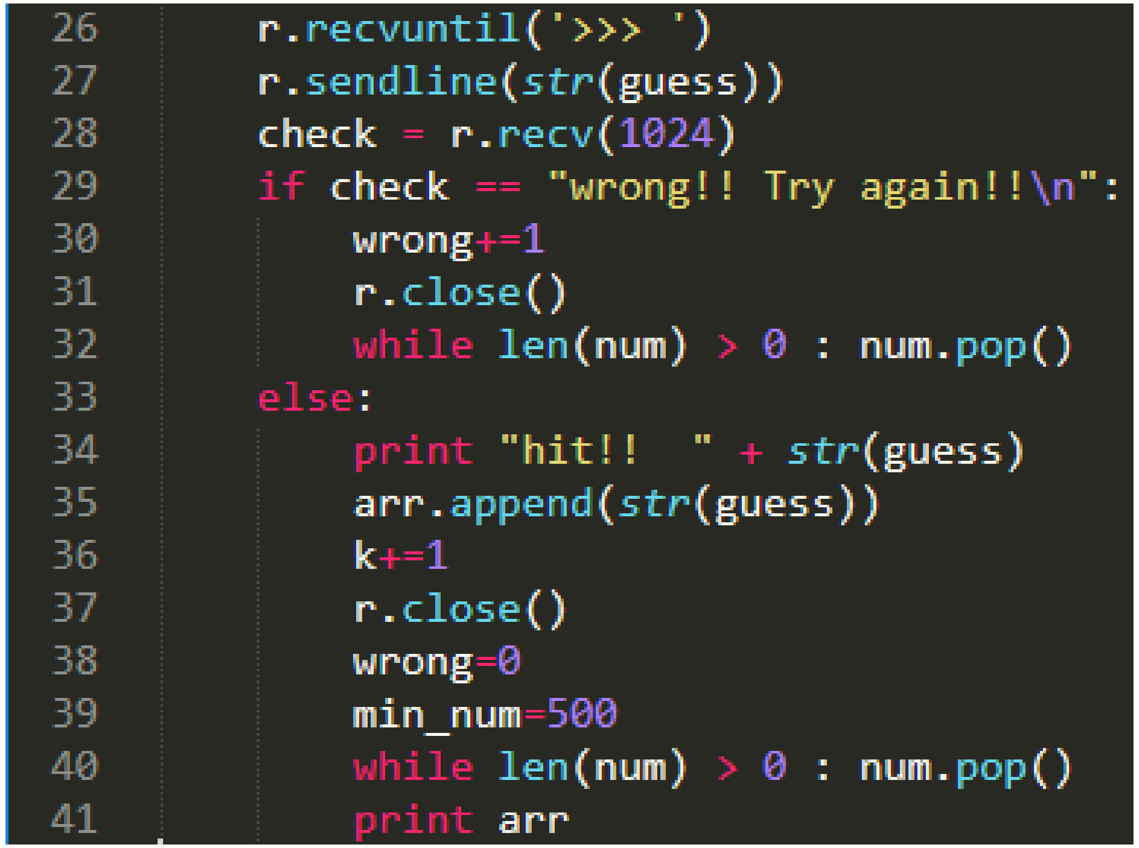
- 만약 최단 경로의 비용이 틀렸다면 정답인 경로는 더 비용이 ↓

- 즉, 최단 경로의 비용 ≥ 최소비용 경로의 비용

Modified Code

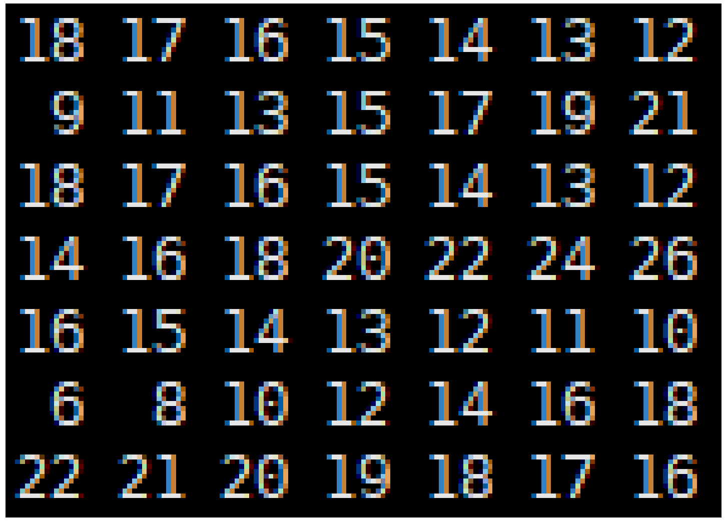


```
while(k!=100):
    r = remote("110.10.147.104", 15712)
    r.recvuntil('>> ')
    r.sendline('G')
    for i in range(len(arr)):
        r.recvuntil('>>> ')
        r.sendline(arr[i])
    print r.recvuntil('***\n')
    for j in range(7):
        num.append(r.recvuntil('\n'))
        total=0
        for x in num[j].split():
            total += int(x)
        if total < min_num:
            min_num = total
    guess = min_num - wrong
```

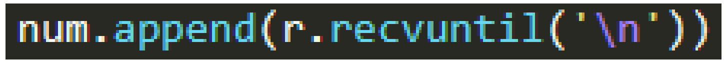


```
    r.recvuntil('>>> ')
    r.sendline(str(guess))
    check = r.recv(1024)
    if check == "wrong!! Try again!!\n":
        wrong+=1
        r.close()
        while len(num) > 0 : num.pop()
    else:
        print "hit!!  " + str(guess)
        arr.append(str(guess))
        k+=1
        r.close()
        wrong=0
        min_num=500
        while len(num) > 0 : num.pop()
        print arr
```

input

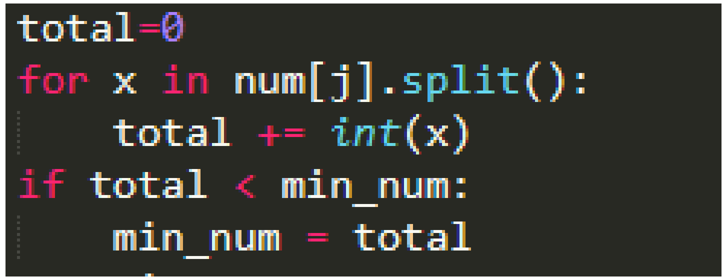


```
18 17 16 15 14 13 12
 9 11 13 15 17 19 21
18 17 16 15 14 13 12
14 16 18 20 22 24 26
16 15 14 13 12 11 10
 6  8 10 12 14 16 18
22 21 20 19 18 17 16
```



```python
num.append(r.recvuntil('\n'))
```

num=[“ 18 17 16 15 14 13 12 ”]



```python
total=0
for x in num[j].split():
    total += int(x)
if total < min_num:
    min_num = total
```

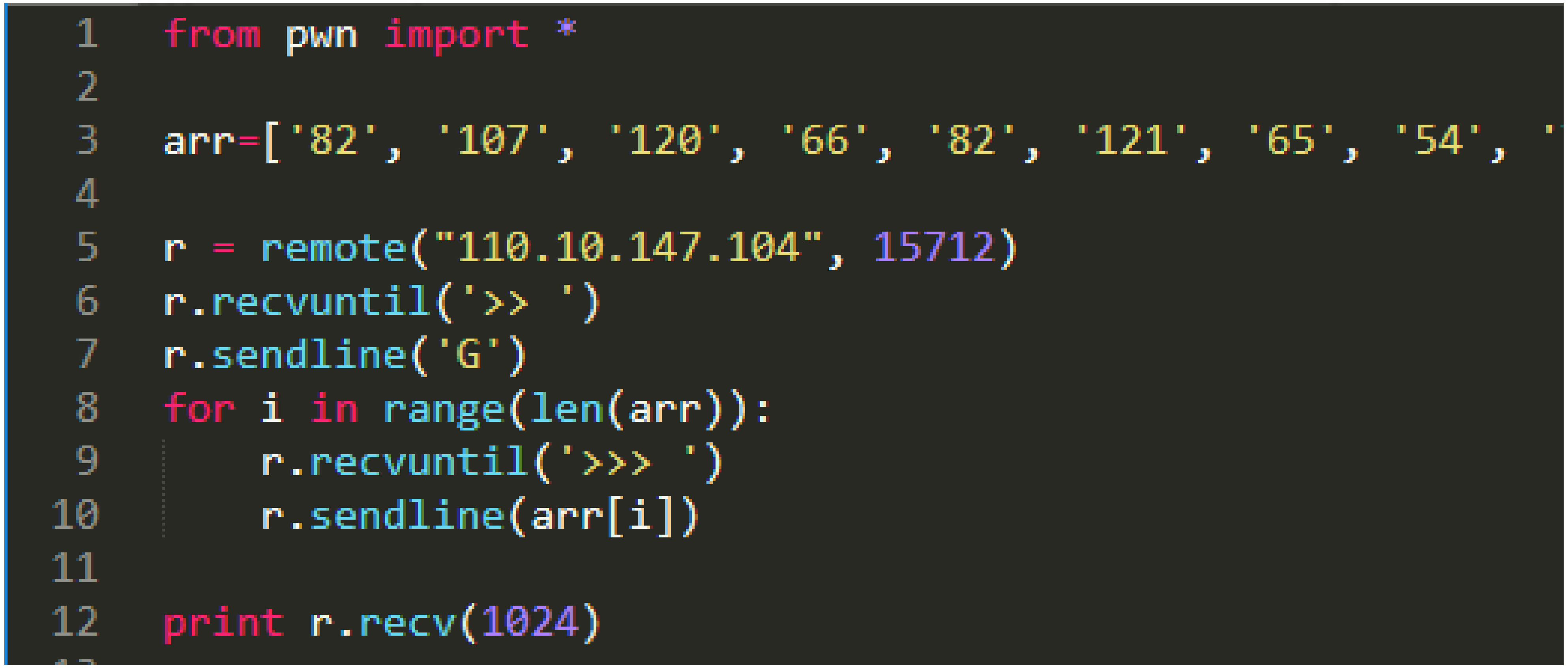
total = 18 + 17 + 16 + 15 + 14 + 13 + 12

최솟값 판단 후 다음 줄도....

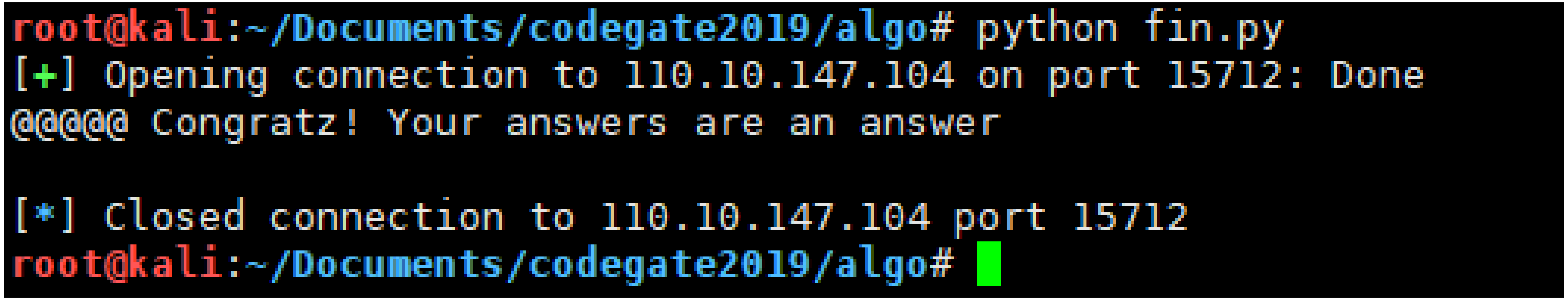
LIVE

Final code

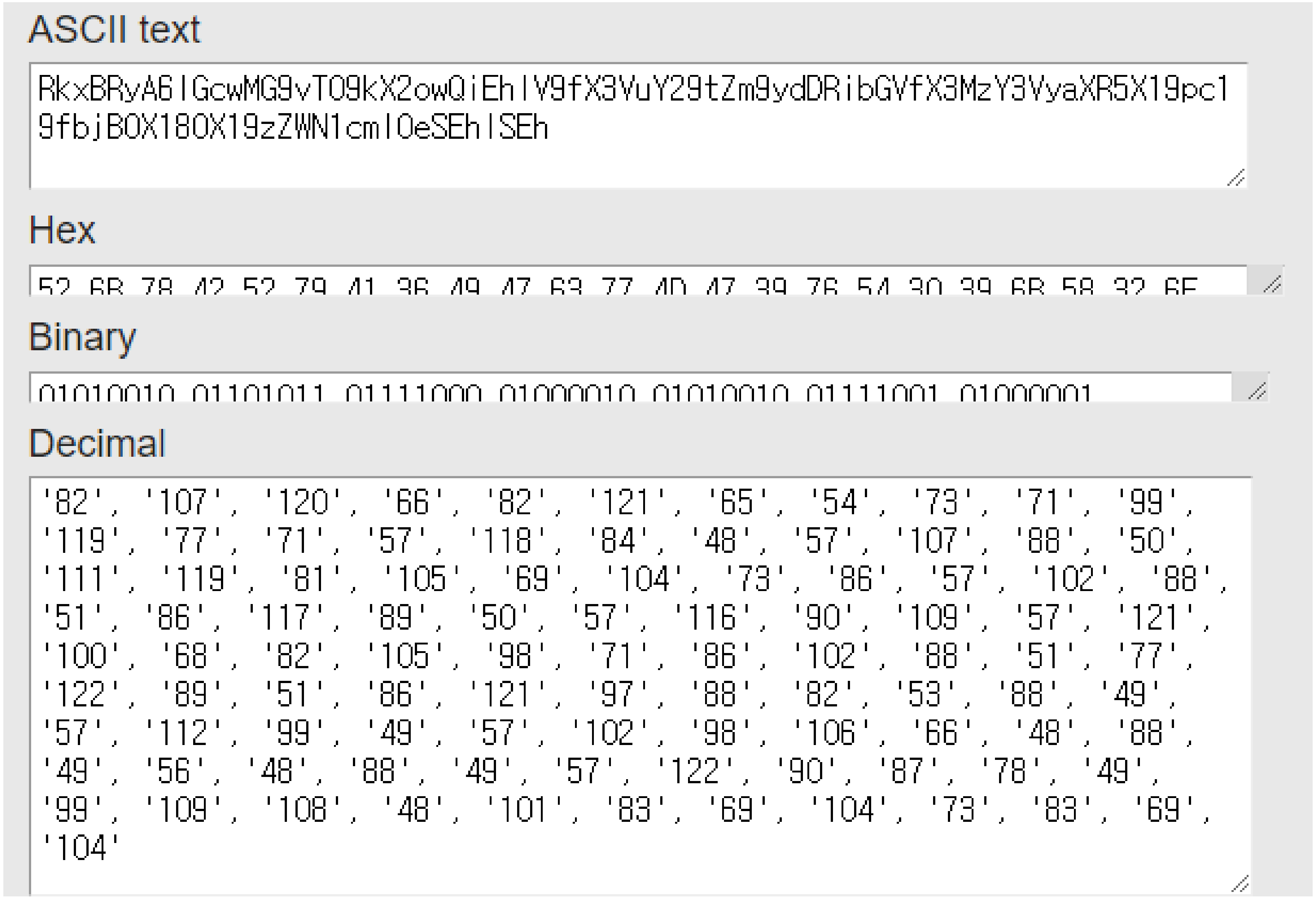
fin.py



```python
from pwn import *

arr=['82', '107', '120', '66', '82', '121', '65', '54', '

r = remote("110.10.147.104", 15712)
r.recvuntil('>> ')
r.sendline('G')
for i in range(len(arr)):
    r.recvuntil('>>> ')
    r.sendline(arr[i])

print r.recv(1024)
```



```
root@kali:~/Documents/codegate2019/algo# python fin.py
[+] Opening connection to 110.10.147.104 on port 15712: Done
@@@@@ Congratz! Your answers are an answer

[*] Closed connection to 110.10.147.104 port 15712
root@kali:~/Documents/codegate2019/algo# 
```



ASCII text

```
RkxBRyA6IGcwMG9vT09kX2owQiEhIV9fX3VuY29tZm9ydDRibGVfX3MzY3VyaXR5X19pc1
9fbjB0X180X19zZWN1cml0eSEhISEh
```

Hex

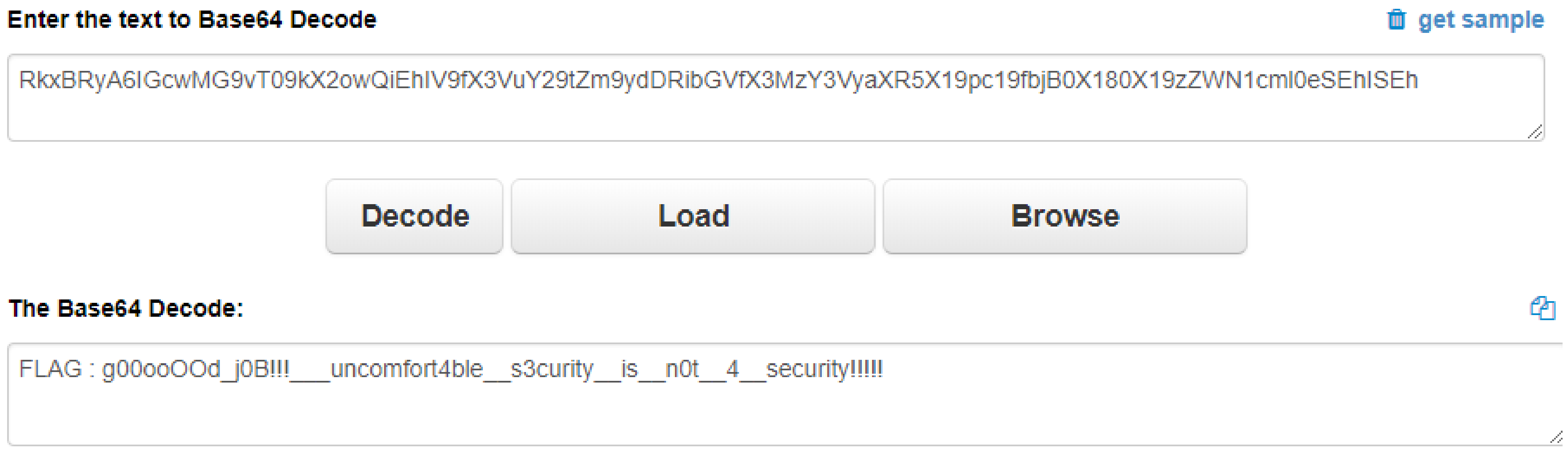
```
52 6B 78 42 52 79 41 36 49 47 63 77 4D 47 39 76 54 30 39 6B 58 32 6F
```

Binary

```
01010010 01101011 01111000 01000010 01010010 01111001 01000001
```

Decimal

```
'82', '107', '120', '66', '82', '121', '65', '54', '73', '71', '99',
'119', '77', '71', '57', '118', '84', '48', '57', '107', '88', '50',
'111', '119', '81', '105', '69', '104', '73', '86', '57', '102', '88',
'51', '86', '117', '89', '50', '57', '116', '90', '109', '57', '121',
'100', '68', '82', '105', '98', '71', '86', '102', '88', '51', '77',
'122', '89', '51', '86', '121', '97', '88', '82', '53', '88', '49',
'57', '112', '99', '49', '57', '102', '98', '106', '66', '48', '88',
'49', '56', '48', '88', '49', '57', '122', '90', '87', '78', '49',
'99', '109', '108', '48', '101', '83', '69', '104', '73', '83', '69',
'104'
```



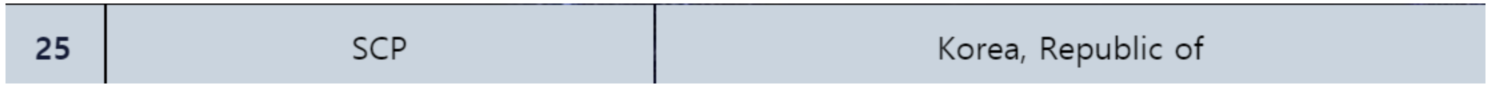
Enter the text to Base64 Decode

RkxBRyA6IGcwMG9vT09kX2owQiEhIV9fX3VuY29tZm9ydDRibGVfX3MzY3VyaXR5X19pc19fbjB0X180X19zZWN1cml0eSEhISEh

The Base64 Decode:

```
FLAG : g00ooOOd_j0B!!!___uncomfort4ble__s3curity__is__n0t__4__security!!!!!
```

결과



25	SCP	Korea, Republic of

졌지만 잘 싸웠다……

두두, 충제, py0zz1