记一次院赛CTF的Crypto和Re题（入门）

原创

CTF小白 于 2019-04-14 14:34:21 发布

分类专栏： CTF 文章标签： CTF 小白入门

本文链接：https://blog.csdn.net/qq_41429081/article/details/89296213

目录

Crypto

easy crypto

easy crypto

50

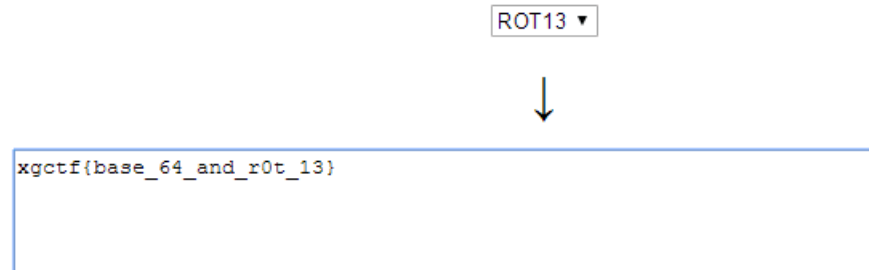
阿强盯着这串奇怪的字符，目光锁定在最后两个等号上....

a3RwZ3N7b25mcl82NF9uYXFfZTBnXzEzfQ==

首先，这个可以很容易的看出这是一个base64加密一串密文，然后用base64解密后是

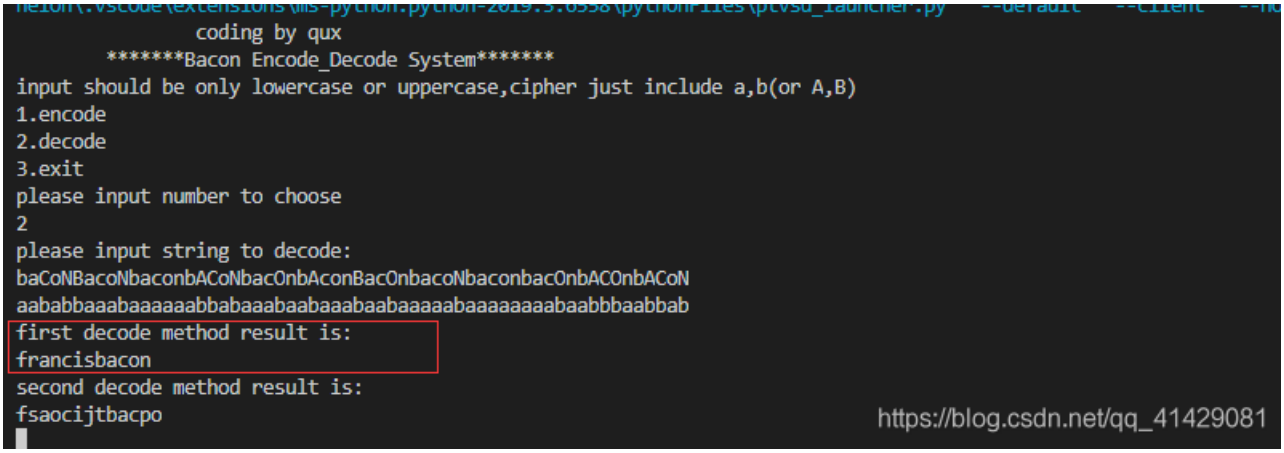
```
ktpgs{onfr_64_naq_e0g_13}
```

可以看出这就是flag的格式，所以一般就是凯撒加密了，但是因为有花括号下划线之类的，所以是凯撒的一个变形rot13



bAcOn

这题是给了一个字符串baCoNBacoNbaconbACoNbacOnbAconBacOnbacoNbaconbacOnbACOnbACoN
可以看出给了我们的提示就是bacon，培根。
我这边的话是改了一个网上找的python代码，然后直接跑出来了。



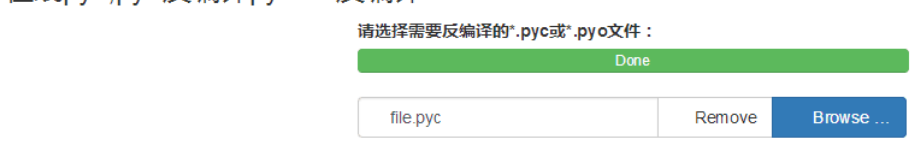
因为培根密码是有两张密码表，同时因为大小写的话也有种情况，解出来应该是有四种（我这边另外一种以小写字母为b以大写字母为a的显示结果没有放出来）

```
import re
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y',
'z']
first_cipher = ["aaaaa","aaaab","aaaba","aaabb","aabaa","aabab","aabba","aabbb","abaaa","abaab","ababa","ababb",
```

```
"abbaa","abbab","abbba","abbbb","baaaa","baaab","baaba","baabb","babaa","babab","babba","babbb","bbaaa","bbaab"]
second_cipher = ["aaaaa","aaaab","aaaba","aaabb","aabaa","aabab","aabba","aabbb","abaaa","abaaa","abaab","ababa"
,"ababb","abbaa","abbab","abbba","abbbb","baaaa","baaab","baaba","baabb","baabb","babaa","babab","babba","babbb"
]
def decode():
    upper_flag = False  # 用于判断输入是否为大写
    e_string = input("please input string to decode:\n")
    e_string=change(e_string)
    print(e_string)
    if e_string.isupper():
        upper_flag = True
        e_string = e_string.lower()
    e_array = re.findall(".{5}",e_string)
    d_string1 = ""
    d_string2 = ""
    for index in e_array:
        for i in range(0,26):
            if index == first_cipher[i]:
                d_string1 += alphabet[i]
            if index == second_cipher[i]:
                d_string2 += alphabet[i]
    if upper_flag:
        d_string1 = d_string1.upper()
        d_string2 = d_string2.upper()
    print ("first decode method result is:\n"+d_string1)
    print ("second decode method result is:\n"+d_string2)
    return
def change(s):
    str=""
    for i in s:
        if(i>='a' and i<='z'):
            str+="a"
        if(i>='A' and i<='Z'):
            str+="b"
    return str
def change2(s):
    str=""
    for i in s:
        if(i>='a' and i<='z'):
            str+="b"
        if(i>='A' and i<='Z'):
            str+="a"
    return str
if __name__ == '__main__':
    print ("\t\tcoding by qux")
    while True:
        print ("\t*******Bacon Encode_Decode System*******")
        print ("input should be only lowercase or uppercase,cipher just include a,b(or A,B)")
        print ("1.encode\n2.decode\n3.exit")
        s_number = input("please input number to choose\n")
        if s_number == "1":
            encode()
            input()
        elif s_number == "2":
            decode()
            input()
        elif s_number == "3":
            break
        else:
            continue
```

敌军密报

题目展示就是这样的，一个密文，以及一个文件可以下载。这边这个文件用记事本打开稍微看了一下，有很多类似.pyt、__main__之类的东西，所以应该就是一个python编译后的文件了，这边推荐一个在线反编译python的网站
http://tools.bugscaner.com/decompyle/




```python
#! /usr/bin/env python 2.7 (62211)
#coding=utf-8
# Compiled at: 2019-03-22 10:43:03
#Powered by BugScaner
#http://tools.bugscaner.com/
#如果觉得不错,请分享给你朋友使用吧!
import base64

def convert(c, key, start='a', n=26):
    a = ord(start)
    offset = (ord(c) - a + key) % n
    return chr(a + offset)


def encode_1(s, key):
    o = ''
    for c in s:
        if c.islower():
            o += convert(c, key, 'a')
        elif c.isupper():
            o += convert(c, key, 'A')
        else:
            o += c

    return o
```


可以发现的是有四层加密
第一层凯撒加密
第二层base32
第三层base16
第四层栅栏加密取2

这边分享下我是如何分析这些代码是如何加密的。
在得到这个python代码后，每一种加密其实都可以单独取出来，然后自己定义一些需要加密的字符，然后可以看到加密后的内容，然后可以判段，如果判断不出来也没关系。
只需要将他的程序逆着写，然后将我们测试的加密后的内容，可以还原出加密前的字符，那么我们逆着的程序也就没有写错了。
这样我们就能一步一步逆向出明文了。

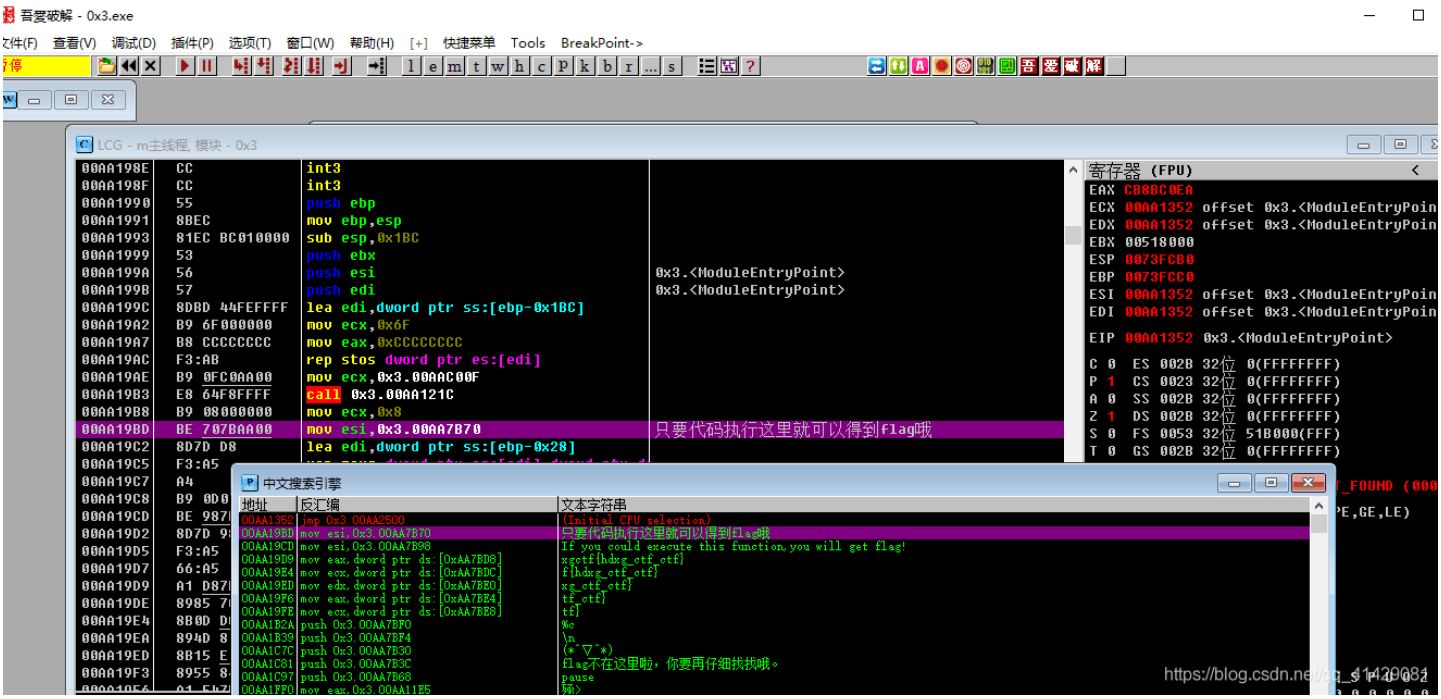
Re

easy re

这一题是真的太没有难度了

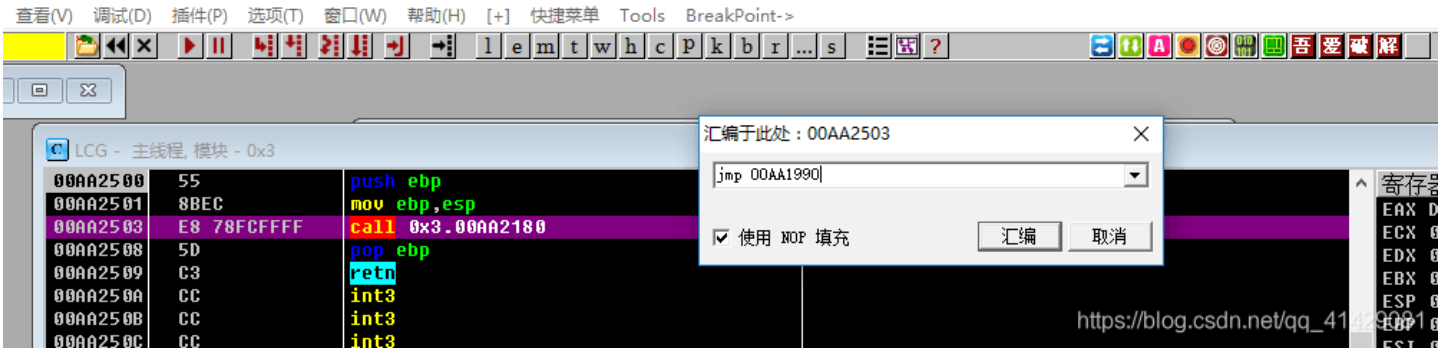
文本编辑器直接就搜出来了，当然更合适的做法肯定是用ida或od去搜

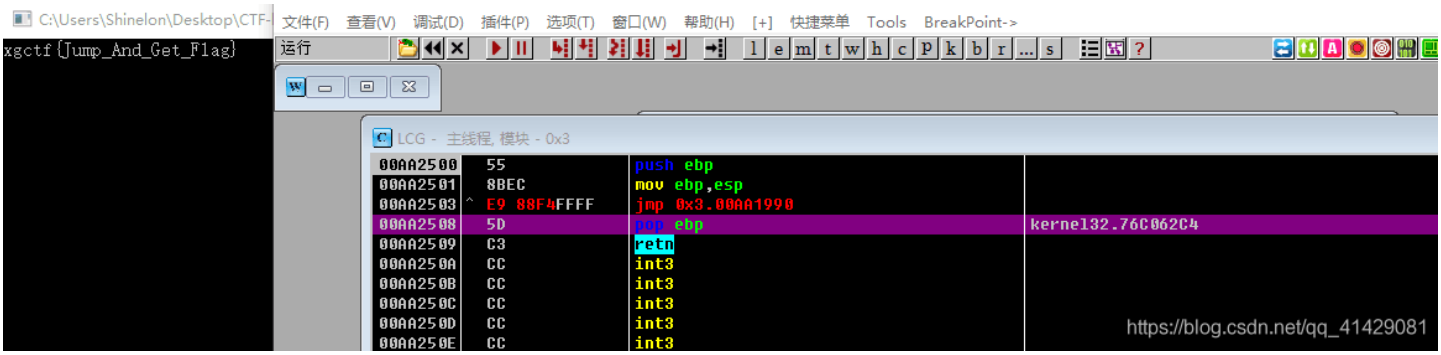
跳到对的地方



这一题的话，就是考察最基本的od动态调试的跳转，使用中文搜索引擎，然后找到相应的内容，双击进去，就可以看到这句话所在的汇编代码段了。然后一般是选择push ebp去跳转就能进去了。

启动程序，这边比较直接的就是直接跳进去，找一个能执行到的地方，改下汇编，让他jmp到我们之前找到的地址。

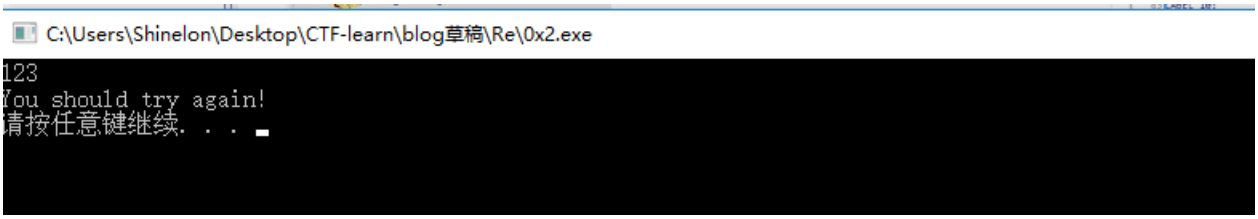




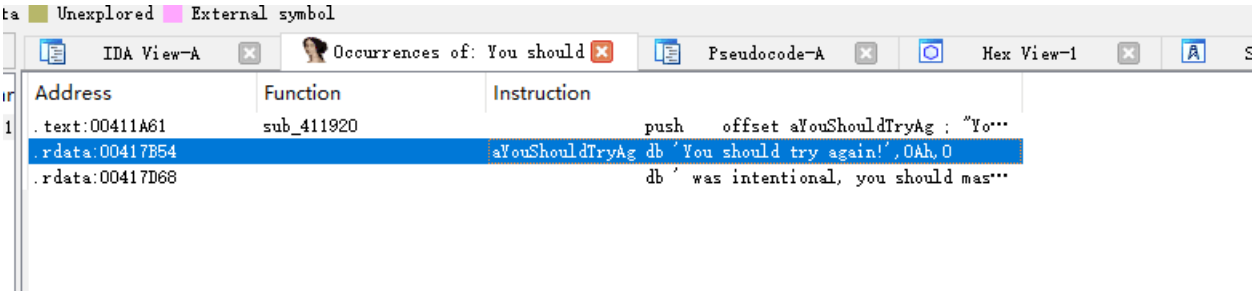
当然，这样可能会导致我们程序崩溃，但是我们只是要这个flag，只要让他显示出来就好了。

简单的XOR

这题我当时是没有做出来的，只要当时没有找到真正的main函数，后来我发现一个找main函数的技巧。因为你直接运行那个需要你逆向的程序，是会出现一些信息的



输入123，他提示我try again。于是可以在ida中搜索这个字符串，ALT+T。



然后我就找到了他真正的main函数，很容易有木有

```
int v0; // edx
int v1; // ecx
int v2; // edx
int v3; // edx
int v4; // ecx
int v5; // ST08_4
signed int j; // [esp+D0h] [ebp-A4h]
signed int i; // [esp+DCh] [ebp-98h]
char v9[36]; // [esp+E8h] [ebp-8Ch]
char v10[36]; // [esp+10Ch] [ebp-68h]
char v11; // [esp+130h] [ebp-44h]
char v12; // [esp+131h] [ebp-43h]
char v13; // [esp+132h] [ebp-42h]
char v14; // [esp+133h] [ebp-41h]
char v15; // [esp+134h] [ebp-40h]
char v16; // [esp+135h] [ebp-3Fh]
char v17; // [esp+136h] [ebp-3Eh]
char v18; // [esp+137h] [ebp-3Dh]
char v19; // [esp+138h] [ebp-3Ch]
char v20; // [esp+139h] [ebp-3Bh]
char v21; // [esp+13Ah] [ebp-3Ah]
char v22; // [esp+13Bh] [ebp-39h]
char v23; // [esp+13Ch] [ebp-38h]
char v24; // [esp+13Dh] [ebp-37h]
char v25; // [esp+13Eh] [ebp-36h]
char v26; // [esp+13Fh] [ebp-35h]
char v27; // [esp+140h] [ebp-34h]
char v28; // [esp+141h] [ebp-33h]
char v29; // [esp+142h] [ebp-32h]
char v30; // [esp+143h] [ebp-31h]
char v31; // [esp+144h] [ebp-30h]
char v32; // [esp+145h] [ebp-2Fh]
char v33; // [esp+146h] [ebp-2Eh]
char v34; // [esp+147h] [ebp-2Dh]
char v35[32]; // [esp+150h] [ebp-24h]
int v36; // [esp+170h] [ebp-4h]
int savedregs; // [esp+174h] [ebp+0h]
```



```
strcpy(v35, "hang_dian_xin_gong_ctf!!");
v11 = 16;
v12 = 6;
v13 = 13;
v14 = 19;
v15 = 57;
v16 = 31;
v17 = 61;
v18 = 9;
v19 = 7;
v20 = 44;
v21 = 39;
v22 = 0;
v23 = 29;
v24 = 0;
v25 = 2;
v26 = 14;
v27 = 29;
v28 = 30;
v29 = 0;
v30 = 27;
v31 = 27;
v32 = 20;
v33 = 0;
v34 = 92;
sub_411154("%s", (unsigned int)v9, 25);
for ( i = 0; i < 24; ++i )
  v10[i] = *(&v11 + i) ^ v9[i];
for ( j = 0; j < 24; ++j )
{
  if ( v10[j] != v35[j] )
  {
    sub_41104B("You should try again!\n");
    system("pause");
    sub_41122B(v1, v0);
    goto LABEL_10;
  }
}
sub_41104B("The flag is your input!\n");
system("pause");
sub_41122B(v4, v3);
LABEL_10:
```

找到了main函数，就可以看的出来，他就是要将我们输入的东西，和v11到v34的值进行异或，最后的结果要等于v35就是”hang_dian_xin_gong_ctf!!”。 这边需要的知识点就是，抑或的时候A ^ B=C那么A ^ C=B 、B ^ C=A就都成立了

于是写了个脚本

mb.py | Untitled-1 | R3.py | bacon.py

```python
s1="hang_dian_xin_gong_ctf!!"
arr1=[]
for i in s1:
    arr1.append(i)
print(len(arr1))
arr2=[16,6,13,19,57,31,61,9,7,44,39,0,29,0,2,14,29,30,0,27,27,20,0,92]
print(len(arr2))
str=""
for i in range(0,24):
    str+=chr(ord(arr1[i])^arr2[i])

print(str)
```


就获得了flag

```
n\.vscode\extensions\ms-python.python-2019.3.6558\pythonFiles\ptvsd_l
24
24
xgctf{This_is_easy_xor!}
```

多密码表替换

找到main函数如下

```
v25 = __readfsqword(0x28u);
v15 = 'swczdaeq';
v16 = 'tnvhfyrx';
v17 = 'iujklmbg';
v18 = 'po';
v19 = 0;
v20 = 'AZXCVBNM';
v21 = 'LKJHGFDS';
v22 = 'EWQYUIOP';
v23 = 'TR';
v24 = 0;
v10 = 'w!k9R!F9';
v11 = 'W88Mt!U3';
v12 = 'dO9';
v7 = '24837159';
v8 = '60';
v9 = 0;
makeEmpty(v14, 20LL);
makeEmpty(buf, 20LL);
puts(&s);
read(0, buf, 0x13uLL);
```



```
for ( i = 0; i <= 18; ++i )
{
  if ( !buf[i] )
  {
    puts(&byte_B00);
    return 0;
  }
}
for ( j = 0; j <= 18; ++j )
{
  if ( buf[j] <= '@' || buf[j] > 'Z' )
  {
    if ( buf[j] <= '`' || buf[j] > 'z' )
```

```
60      {
61        if ( buf[j] <= '/' || buf[j] > '9' )
62        {
63          if ( buf[j] == 95 )
64            v14[j] = '!';
65        }
66        else
67        {
68          v14[j] = *((_BYTE *)&v7 + buf[j] - 48);
69        }
70      }
71      else
72      {
73        v14[j] = *((_BYTE *)&v20 + buf[j] - 97);
74      }
75    }
76    else
77    {
78      v14[j] = *((_BYTE *)&v15 + buf[j] - 65);
79    }
80  }
81  for ( k = 0; k <= 18; ++k )
82  {
83    if ( *((_BYTE *)&v10 + k) != v14[k] )
84    {
85      puts(&byte_B28);
86      return 0;
87    }
88  }
89  puts(&byte_B60);
```


我把它改写成python代码大概是这个意思

```
1  buf=[]
2  v14=[]
3  v10="9F!R9k!w3U!tM88W9Od"
4  v7="9517384206"
5  v20="MNBVCXZASDFGHJKLPOIUYQWERT"
6  v15="qeadzcwsxryfhvntgbmlkjuiop"
7
```

```
30  '''
31  for i in range(0,19):
32      if(buf[i]>='A' and buf[i]<='Z'):
33          v14.append(v15[ord(buf[i])-65])
34      elif(buf[i]>='a' and buf[i]<='z'):
35          v14.append(v20[ord(buf[i])-97])
36      elif(buf[i]>='0' and buf[i]<='9'):
37          v14.append(v7[ord(buf[i])-48])
38      else:
39          v14.append('!')
40
41  '''
```


然后最后这个v14等于v10就是正确的

这样我们就知道v14，于是上面的代码中我们不知道的就只有需要我们输入的buf[]。所以写一个求buf的代码

```
v15="qeadzcwsxryfhvntgbmlkjuiop"

for i in range(0,19):
    v14.append(v10[i])

for i in range(0,19):
    if(v14[i]=='!'):
        buf.append(95)
    for j in range(0,len(v15)):
        if(v14[i]==v15[j]):
            buf.append(j+65)
            continue
    for j in range(0,len(v20)):
        if(v14[i]==v20[j]):
            buf.append(j+97)
            continue
    for j in range(0,len(v7)):
        if(v14[i]==v7[j]):
            buf.append(j+48)
            continue
str=""
for i in buf:
    str+=chr(i)
print(str)
'''
for i in range(0,19):
```


运行就能得到flag

```
ions\ms-python.python-2019.3.6558\pythonFiles\ptvsd_laur
0k_y0U_G4t_Pa55w0rD
```