¿Reversing o no reversing? Esa es la cuestión.

Hemos encontrado un programa que verifica si la flag que has introducido es correcta o no.

1. no hay extension, vamos a ver qué hace file

```
❯ file holaKeAseEntraOKeHase
holaKeAseEntraOKeHase: Python script, ASCII text executable, with very long lines
```

2. wow es un script directamente vamos a ver cómo es

```
#!/usr/bin/env python
print("Enter the flag and I will check it for you.")
gqvHmYuMFcJZIgGlcaCijvOCFVVRxaAHznpdHZSvPOosbBsBARzcQqzxlqGwkWHQgJisBtyoNoGakfPXrYrzQPmEeDZYmiKBhKYYlBZsvcR
etTpKhVLpdCNOVEpAnERv = raw_input()
tLwXLmpmvtaFnOXyUVawvhuPXUkOkPaOyATJbvCFnNLLJstBPXzkFLsoPtWjWmXCELoVqtwVMcUejFbrDGIfrmGtayUhcGVBUSxFidlIUVd
mjcgrCpGZaCdQAbMGFnNO = [[[], [], []], [[], [], []], [[], [], []]]
if
len(gqvHmYuMFcJZIgGlcaCijvOCFVVRxaAHznpdHZSvPOosbBsBARzcQqzxlqGwkWHQgJisBtyoNoGakfPXrYrzQPmEeDZYmiKBhKYYlBZ
svcRetTpKhVLpdCNOVEpAnERv) == 27:
    for
FanDUiNixUXFyymvDOppNjqGUtJzGVTFCmDECCFiAqqksvhBDmKhExTIXtJJYKWhGQaQkCNszRHdJrBPpkJRtyQRQTvTECrczrFMeiOTRIx
CSJjGYoEwtIgtPBivsKWv in range(3):
        for
AWThnRgGmxNIxvPijzPqVQHSOTRMOrodGHBMRskKTznwymAXOQblVxldzGYGvAmWLMjVQmfsWdPCsSkiSUodxCfNDErGyOJKlJcQApFPlhm
gOLMOsnJGZkHIwGYNFDTq in range(3):
            [**************************************************]
    if
RECGXHmnXfHWkqSibXniUTAxBtCVWxeUjyQtktOPZaeiSuBtDcsykbrLUfSIfibeUINCRliGptxkvobsRXOgyuiISEUtpdOcSgTxIzOymZp
QNvsjZhUzNSxFtRgvGCCq == "z}zzzeo_dbr{OnkacHiasmae_do":
        print("Your flag is correct!")
    else:
        print("Your flag is incorrect. :(")
else:
    print("Your flag is incorrect. :(")
```

3. Es un código obfuscado aparte de "z}zzzeo_dbr{OnkacHiasmae_do" que parece la flag removida.
4. Vamos a ver cómo funciona:

```
❯ ./holaKeAseEntraOKeHase
Enter the flag and I will check it for you.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Your flag is incorrect. :(
```

5. Para poder leer mejor vamos a desobfuscarlo.
6. Vamos a probar esto:
   https://github.com/sleeyax/PyDeobfuscator
   Tiene 2 modos: - intensio - pyminifier
   Probamos el primero:

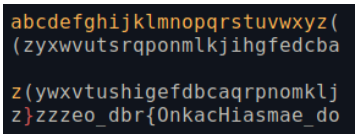
```
❯ python3 deobfuscator.py -i ../out/holaKeAseEntraOKeHase -o ../codigoDesobfuscado.py -d intensio
```

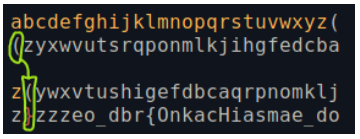
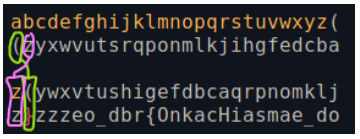
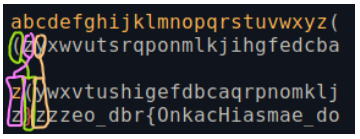
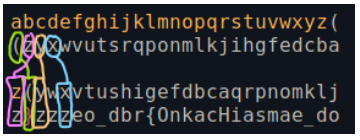
6. Vamos a ver si nos ha deobfuscado el código, si no, probamos

```
❯ cat codigoDesobfuscado.py
File: codigoDesobfuscado.py
#!/usr/bin/env python
print("Enter the flag and I will check it for you.")
var1 = raw_input()
var2 = [[[], [], []], [[], [], []], [[], [], []]]
if len(var1) == 27:
    for i1 in range(3):
        for i2 in range(3):
            for i3 in range(3):
                var2[i1][i2].append(var1[i1*9 + i2*3 + i3])
    var3 = ""
    for i1 in [var2[2], var2[0], var2[1]]:
        for i2 in i1[::-1]:
            for i3 in [i2[1], i2[2], i2[0]]:
                var3 = var3 + i3
    if var3 == "z}zzzeo_dbr{OnkacHiasmae_do":
        print("Your flag is correct!")
    else:
        print("Your flag is incorrect. :(")
else:
    print("Your flag is incorrect. :(")
```

7. Vale. Lo que hace es coger el input y darle muchas vueltas para que termine de una forma concreta.
8. Podemos tener 2 formas de resolverlo: a la cuenta de la vieja o con zmt solvers (z3 pro ejemplo). Yo he decidido resolverlo a manita y con lógica.
    1. Como sé la longitud voy a hacer que var1 sea "abcdefghijklmnopqrstuvwxyz("
    2. hago un print de var3 al final de toda la mezcla para ver cómo se ha transformado el string inicial:

```
❯ python codigoDesobfuscado.py
Enter the flag and I will check it for you.
z(ywxvtushigefdbcaqrpnomklj
Your flag is incorrect. :(
```

    3. ahora sabemos cómo varía el string. Es decir. Tenemos la posición de cada una de las letras de la flag.
    4. vamos a recolocar las letras (Tip: dale la vuelta al abecedario para ir de atrás adelante y no tener que mover el cursor)
    5. 

    6.  }
    7.  z}
    8.  zz}
    9.  zzz}
    10. Así hasta terminar el abecedario.

La flag es: `HackOn{bro_demasiado_ezzzz}`