Análisis del problema

El mayor problema del ejercicio es poder “resolver” los hash md5, no existe una forma 100% correcta de hacerlo, encontramos varios servicios web y apis que lo hacen, normalmente de pago. Ejemplo https://www.dcode.fr/funcion-hash-md5
No es una opción viable para intentar resolver los 300 hash que hay en el fichero.
Busco la opción de hacerlo manual mediante una RainbowTable https://es.wikipedia.org/wiki/Tabla_arco%C3%ADris , veo que hay que generar las tablas y demás, intento buscar una opción más rápida sin descartar esta.
Pruebo el archiconocido JohnTheRipper, utilizando uno de los diccionarios que incluye la suite Kali, pero no me termina de cuadrar, a veces me aparece un resultado y otras, con el mismo hash, no, descarto la opción.

```
┌──(kali㉿kali)-[~/testJohn]
└─$ sudo john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash1.txt

[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4×3])
No password hashes left to crack (see FAQ)
```

Desarrollando una solución

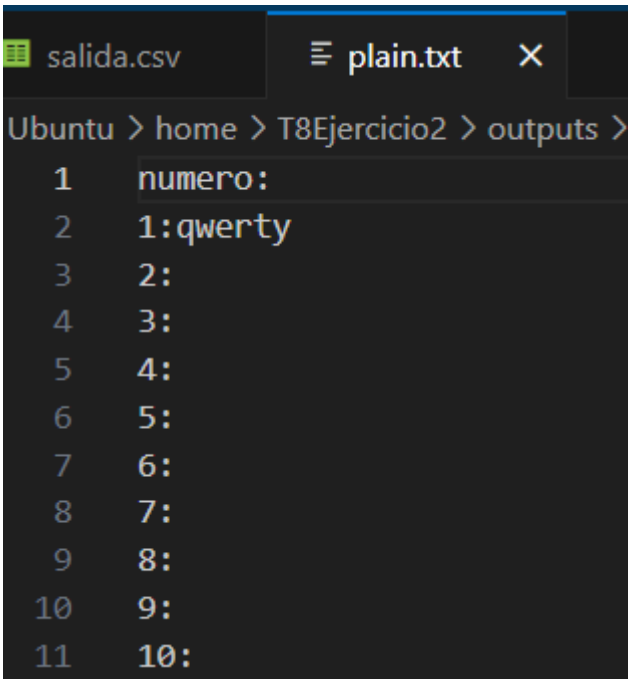
Buscando alternativas encuentro este proyecto que me cuadra: https://github.com/amirgi73/pyDecryptor/tree/master
Veo que utiliza su propio diccionario, el fichero de entrada, debe ser un csv tal que así *indice, hash*
Desarrollo un script en python que dado el fichero de hashes, nos cree un csv llamado **salida.csv**, el fichero se llama **setCsv.py**(está en raíz de git)
Captura parcial del fichero salida.csv, ubicada en el directorio /outputs

untu > home > T8Ejercicio2 > outputs > salida.csv

```
1   numero,string
2   1,d8578edf8458ce06fbc5bb76a58c5ca4
3   2,69f1975c3dde3f6894affab807aa00a0
4   3,429839149dcc0fb15647887600l5d612
5   4,81078a9e1947b5b78b0c3b0243b0c170
6   5,f7999e2ae9a91c18ae41a936a8c629ab
7   6,e64c093b3be40fa2d0765005a60d11c3
8   7,45e1d90a3e18ed0ee875139b1ff2076e
9   8,e1a7857c3893eed4250f5f3f6d87d350
0   9,ca09c50bc1b4214b997f6d38c7e659c5
```

Ahora ejecutamos el script pyDecryptor descargado de su repo.
Elijo la primera opción “ Unhash me using a password list file”, introduzco los nombres de los ficheros y luego elijo MD5
Se ha generado el fichero **plain.txt**, guardado en el git dentro del directorio Outputs, este ya nos muestra en una línea el MD5 resuelto si ha sido posible, si no, mostrará la línea vacía

En el ejemplo, el primer hash ha sido resuelto, los 9 siguientes, no.



Por último, creamos el script **encrypt.py**, este leerá el fichero plain.txt, las líneas en las que encuentre una contraseña ya descifrada, las encriptará usando sha-256, las que no, marcará líneas en blanco, generando como salida, el fichero **new_passwords.txt** en el fichero outputs