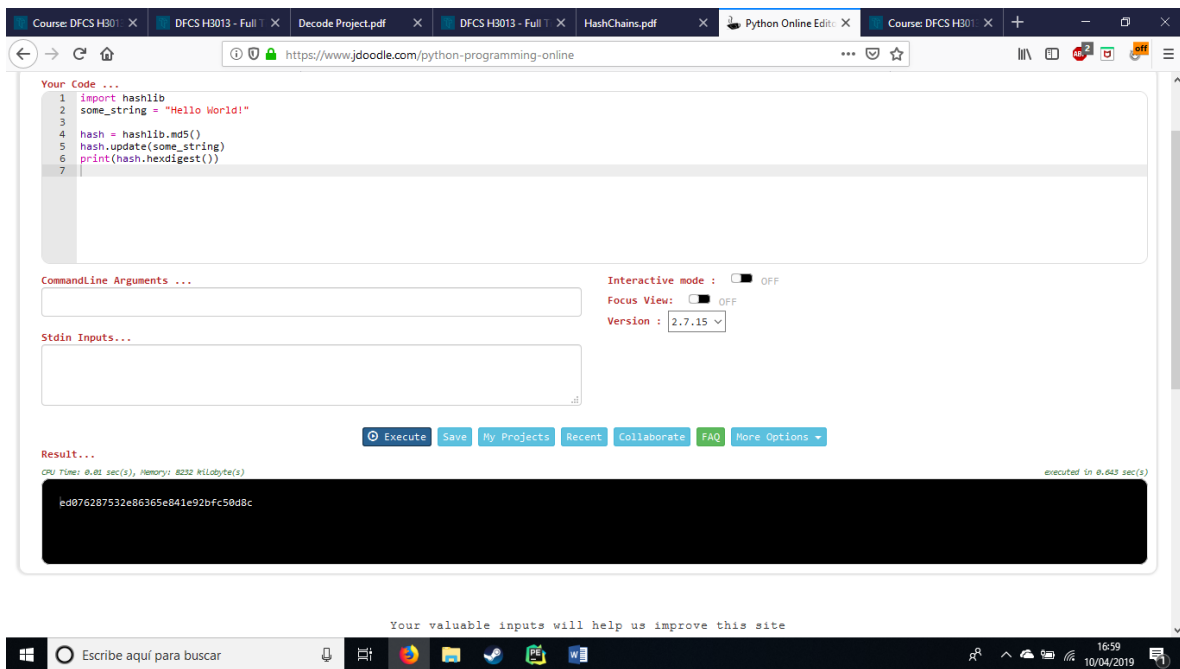


# LAB 8 - BLOCKCHAIN

Cristian Villarroya Sánchez

The first test is just for testing how to show a hash in python2.

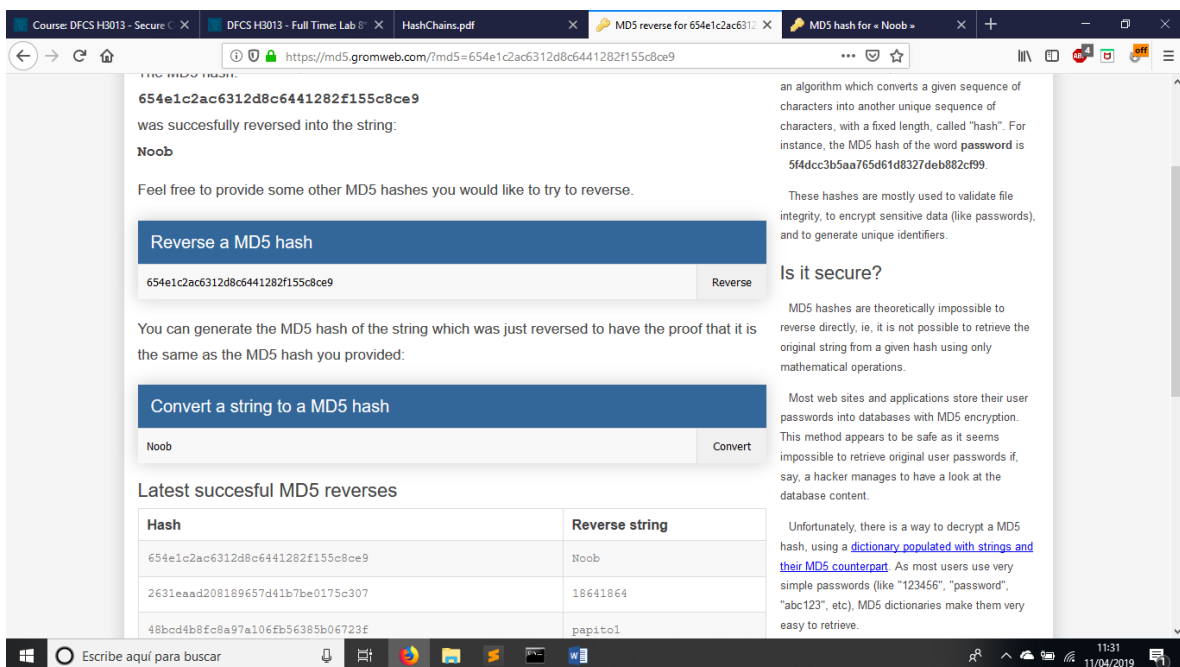


```
1 import hashlib
2 some_string = "Hello World!"
3
4 hash = hashlib.md5()
5 hash.update(some_string)
6 print(hash.hexdigest())
7
```

Result...

e4d076287532e86365e841e92bfc58d8c

Now its time to search for info, we have the root for login as the user nOOB. If we search for it on google, we have a match for that key. Is a md5 key and the match is the word Noob.



MD5 reverse for 654e1c2ac6312d8c6441282f155c8ce9

654e1c2ac6312d8c6441282f155c8ce9 was successfully reversed into the string: Noob

Feel free to provide some other MD5 hashes you would like to try to reverse.

Reverse a MD5 hash

654e1c2ac6312d8c6441282f155c8ce9 Reverse

You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:

Convert a string to a MD5 hash

Noob Convert

Latest succesful MD5 reverses

Hash	Reverse string
654e1c2ac6312d8c6441282f155c8ce9	Noob
2631eaaad208189657d41b7be0175c307	18641864
48bcd4b8fc8a97a106fb56385b06723f	pipito1

an algorithm which converts a given sequence of characters into another unique sequence of characters, with a fixed length, called "hash". For instance, the MD5 hash of the word password is 5f4dcc3b5aa765d61d8327deb882c99.

These hashes are mostly used to validate file integrity, to encrypt sensitive data (like passwords), and to generate unique identifiers.

Is it secure?

MD5 hashes are theoretically impossible to reverse directly, ie, it is not possible to retrieve the original string from a given hash using only mathematical operations.

Most web sites and applications store their user passwords into databases with MD5 encryption. This method appears to be safe as it seems impossible to retrieve original user passwords if, say, a hacker manages to have a look at the database content.

Unfortunately, there is a way to decrypt a MD5 hash, using a [dictionary populated with strings and their MD5 counterpart](#). As most users use very simple passwords (like "123456", "password", "abc123", etc), MD5 dictionaries make them very easy to retrieve.

If we look the new word and the original word, we can see that the upper and lower cases are swapped. This gives as a hint about how to calculate the root.

Knowing this, I have create a method in python that, given a seed, just swap the lower case for upper case and viceversa.

Now its time to calculate the hash, but first lets do a little bit of research about hash chains.

A hash chain is simply a hash of a hash of a hash...

They can be used in login processes. For example, if we hash 1000 times, the server will store the hash 100 of the password which is provided by the user. Then, when the user authenticates, it sends the hash that hashes that hash in the server (hash99(password)).

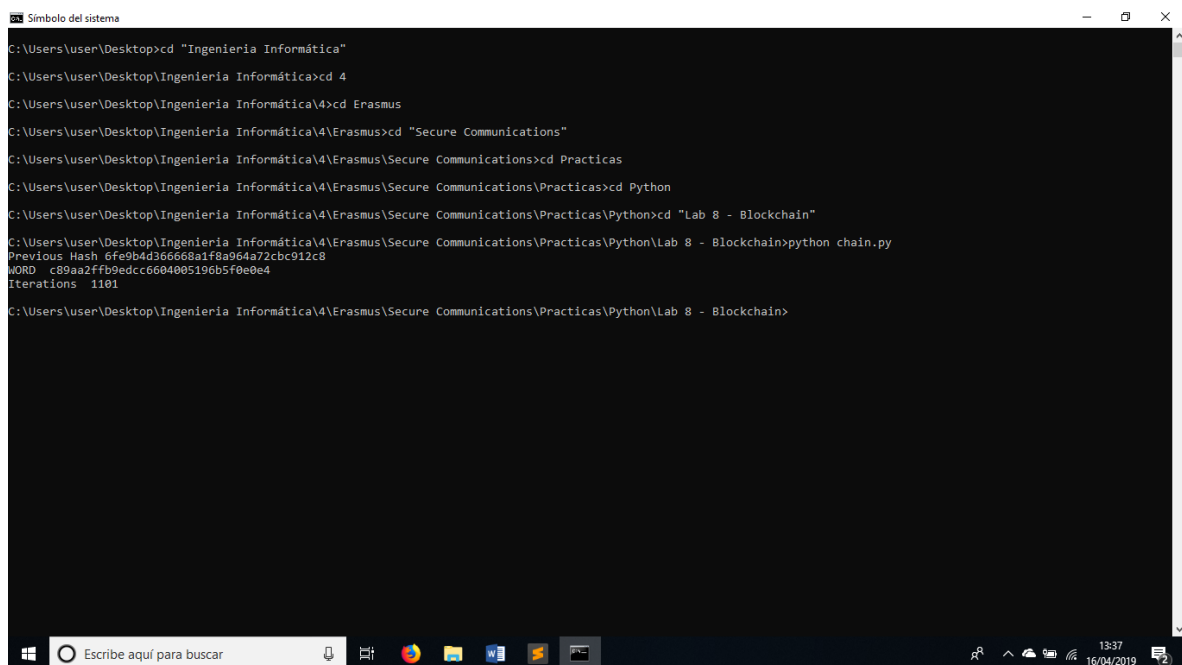
Knowing this, we are asked for the hashX-1 of the password that hashing that value will result in the given hash (c89aa2ffb9edcc6604005196b5f0e0e4)

For this, I have create a method in python. This method calculates a hash from a given seed.

Now that I have the methods for calculating the seed, and calculating the hash I can start looking for the hash asked in this exercise.

The method is simple, I star with the seed given (ECSC), then I calculate the seed (swap lower-upper cases) and I create a first hash of it. I have the expected hash stored in a value. I also create a tmp value for storing the previous hash(that will be the solution of the exercise) and a value to know how many iterations have been made. Finally, I create a while method in which I iterate until the word(hash) is equal to the expected hash. When the match, I print the previous hash(solution), the actual hash (expected) and the iterations made.

Here is the final output :



```
Símbolo del sistema
C:\Users\user\Desktop>cd "Ingeniería Informática"
C:\Users\user\Desktop\Ingeniería Informática>cd 4
C:\Users\user\Desktop\Ingeniería Informática\4>cd Erasmus
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus>cd "Secure Communications"
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus\Secure Communications>cd Practicas
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus\Secure Communications\Practicas>cd Python
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus\Secure Communications\Practicas\Python>cd "Lab 8 - Blockchain"
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus\Secure Communications\Practicas\Python\Lab 8 - Blockchain>python chain.py
Previous Hash 6fe9b4d366668a1f8a964a72cbc912c8
WORD c89aa2ffb9edcc6604005196b5f0e0e4
Iterations 1101
C:\Users\user\Desktop\Ingeniería Informática\4\Erasmus\Secure Communications\Practicas\Python\Lab 8 - Blockchain>
```

So the solution will be 6fe9b4d366668a1f8a964a72cbc912c8 which can be found in 1101 iterations.