Presentación

Nombre: Omar Sánchez Díaz

ID: 1094544

Asignatura: Laboratorio de Análisis Númerico

Profesor: Javier García Maimo

Tema: Caso práctico

Detector y mapeado de nivel de seguridad de contraseñas

Indice:

Introducción :

Las contraseñas están quedando obsoletas, ya que todas pueden ser hackeadas con algoritmos de fuerza bruta usados por ciberdelincuentes, que consisten en insertar una contraseña tras otra, millones y trillones de veces hasta que inserte la adecuada.

Sin embargo, las alternativas para refuerzo como la doble autentificación no han sido implementadas por muchas páginas webs por el momento. Por lo cual, busco diseñar un programa para que las páginas puedan detectar la fortaleza de la contraseña insertada por el usuario.

Pero, ¿A qué se refiere la fortaleza de una contraseña? Bueno, en base a lo comentado sobre los ataques de fuerza bruta, hay ciertos criterios con los que tiene que cumplir una contraseña para ser más "fuerte" contra los ataques de fuerza bruta. O en otras palabras, mientras más de estos criterios se cumplan, más tiempo va a tardar el ataque en dar frutos. Para esto Hive Systems diseño una tabla que muestra dicho criterio:

4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years

Criterios

Para la contraseña ser valida debe cumplir con los siguientes criterios:

- **Tener 8 o más caracteres**
- **Contener 1 o más números**
- **Contener 1 o más símbolos**
- **Contener 1 o más letras en mayúscula**
- **Contener 1 o más letras en minúscula**

Planteamiento

- **Etapa 1: Diseñaremos un generador de contraseñas aleatorias, para hacer un caso practico de nuestro programa.**
- **Etapa 2: Creación de lista de contraseñas aleatorias para mapeado**
- **Etapa 3: Diseñaremos el programa para verificar la fortaleza de las diferentes contraseñas generadas aleatoriamente.**
- **Etapa 4: Verificaremos la fortaleza de la lista generada en la etapa 2**
- **Etapa 5: Mapearemos los resultados obtenidos de las diferentes contraseñas, para visualizar la cantidad de contraseñas erradas por parte del generador de contraseñas que actua como los usuarios.**

Etapa 1: Generador de contraseñas

In [69]:

```
import string
import random

## Crea una lista de caracteres que usaremos
characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")

def generador_aleatorio():
 ## Longitud = Generara aleatoriamente la longitud de la contraseña, yendo de 0 a 15
 longitud = (random.randint(1,15))

 ## Mezcla aleatoriamente los caracteres de la lista
 random.shuffle(characters)

 ## Escoge caracteres de la lista aleatoriamente
 password = []
```

```
 for i in range(longitud):
  password.append(random.choice(characters))

 ## Mezcla aleatoriamente la contraseña generada
 random.shuffle(password)

 ## convertimos la contraseña a string
 psw = ("".join(password))
 return psw
```

Etapa 2: Creación de lista de contraseñas aleatorias para mapeado

In [74]:

```
##Iteraciones para el bucle
i = 0

##Lista de contraseñas
lista = []

##Bucle para añadir contraseñas aleatorias a la lista
##En este caso, usaremos 400 contraseñas para la lista
while (i < 400):
    lista.append(generador_aleatorio())
    i +=1

##Mostraremos la lista
print("Tamaño de la lista: ",len(lista))
print("\nLista aleatoria")
print(lista)
```

```
Tamaño de la lista:  400

Lista aleatoria
['vYxSxl1ewpX', 'LH0T3)e', 'u!%QrJc8np$^', 'AbmN*42HM2', '@s0', 'ePF', '$BP', 'G5Zh$lTI9'
, 'N)Lfg@Obuu', 'RvB@!t)#VWm1MR', 'Pw0qlm@%oQABzka', '32OQ', '#', 'G93', 'j7H%Uv@8b@5^i',
'O0CDF0&6acGwTD8', 'jKN&KTEH3^', 'Pc@&^U%H', 'kLw', '!AP^#2&a', 'zBq', 'oEvDj2lMh8V(U', '
16ILEQ5kAD@L9', '$Rb', 'AN', 'e3FIA', 'Y@5f', 'Zg', '@mK2C', 'cTw#D1No', 'PDYjsUqxa', 'Ap
4QRskQ^hvng', '612Xbo', 'dDEZl4cCgggrn', 'X2GddDD@', 'WYj9VuXD%U5469E', 's99YvE', '@(ZpKF
mH9TZg&(', 'L!', 'lxQ64R(%$h7', 'QxhtoInk@', 'GvfRW(gt6d!8Z7', 'mbynTJE0jaBY', 'P57', 'w1
MUNe', 'FlT', 'fT)$GLap#Gx', 'lrlt6BW', 'kDMngk$bsB#', 'eKVUVSqOObG1', '4uQE*Z%Pd%v', '7m
B!SnR7r8^c1^', '6L0x#', '6q6^(tUHtwN', ')I%gz#!', '&dblk', 'usITYEq', 'T#', 'Ae)j', '98T7
n*u', 'RI8qjL(bTS', 'Ia^YHv', 'F5k$', 'JUoEUXNJ', 'h', 'jtrWcdbRpCePb1', 'SbmmBP', '^J&z4
!^', '$Wsud%XrQ9^A0', 'v^pn$AQ@', 'MDb*S8', 'pRbOC9)4c', 'B3!', 'PQg#uA', 'DDR', 'heD$Cgm
^JQ', 'gPLVjm', 'ZDw2o82QBie*Dj1', 'd4', '3hgV', 'SOg6Y!XRK#S*', 'vdCFd3EJS!ZcJ9)', 'Pb(H
5*3Bto!B', ')(LZ%', 'pIGsdG', '%bN8vX7zpx', '6TN^q9OBIK', '#7@8P', 'yTaOQRFl%Iad', 'cma',
'(fqbeDos5V', '(6ZH$v&oEB', 'n6sH3fl^Tvg', 'nJ', 'qp', 'b!pSXy', '4GZoqIE)5', '8StM%z8M',
'&tvUh', '(GZit', 'Bxx0$aCVcXhCO', 'V', '3l9f*xRyuLsk', 's$jT3', 'FLa7uWFZmz3D', 'iFm19',
'yGTV$kOaf1t)', 'o4jRCmi^F0*DX#', 'sXn14z', 'V79SRfOk', 'Ye0pkn#39O', 'p9fZFBq!', '33', '
^kI707sIOobL4(', 'cnKn9@pTFnXe', '6!6dASTlp0CwgmI', 'z*GECH1hqPtBiR', '1Lj@kOk*ewQWavh',
')Sj@RtdsVVbw', 'M8X', 'F7EGtn', 'B', 'iRb', '%K$QQ6q(QQ', 'uY0', 'cp3^k5Pb', 'TXAm8pneMK
', '@MP*LbL)', ')DH', '!KAT', '4R', 'Bv^A(LX&PgU', 'CPTV', '%hz6', 'wINSmEcDhn6', '@RusAO
sLO)Hjq', 'iOeRJ08*)Sek1c6', 'E', '^rm', 'Y', 'S%gZOH&EI', '9jmXZE2Qvx%3f', 'Fv^dh', '5E5
qmoPU', 'JNvow)PoFyKFU8T', 'fpdys', 'al88Ve', 'X5VMSX', 'R#!h4', '@', 'cg', 'J2dG', 'I0cC
tMnm^qAl2', '5hN', 'vMTJy#i&Zm^&a', 'wCHiXeBcx0v', 'F*@A', 'lEPkP6', 'B%lbwsZxEvx#', 'b!k
B0T', 'Ubt2%MP58J%', 'K^L', 'Li', 'B4yY', ')j**y#0H8Tes4yF', 'g@fg', 'fc*LqvjM8p', 'xvVVB
)5DU1vB2', '#', '1%QXtzcBkXUFh', 'e5', '$nRQ7f%l5vwt', 'eLz!n2gpkq@E', 'W!Fy6q14HJ6Jl(',
'mMzJN9F', 'ljjkd43iKkK&Rkc', '6d', 'Gid', 'fD7)9Yyv', 'AjZC6MHn^Yr%16h', '5phxK0!(', '5Y
cQ', 'F', '^', 'qEq3sBJulPs', 'O5nj6peuh)vQM', '51r', 'm*qf94afyJz6U', 'zHmRDN#', 'LM*a',
'Qy9y%I*)5', 'I7sITO0', 'TIR0t', '*DR1A)z6SwbKb', 'h&r7z!5sx1CSEg', 'tqhsnDE', 'HzHs%eQo*
', 'cWE39Zw^@', 'bJL%mbWHu', 'm*r)E&qCkR#(Xp#', '5BRRBviWV', ')', '6gT2(6^iCjp!lg', '7z(r
pOHPQY', 'o$mT7sDyW', '!byS', 'RL', 'UE@@T', '#grogrsSUAK', ')IXlW5tME5WoHL', 'hG&zB$%Ti'
, 'aG5)3Sp708f', 'SId8R14v8q9W2%p', 'qMFsW', 'ZUZR(9G', 'mcV#6*o58oVp', 'AoV', 'Asi(C', '
ZGm$SwLZys', 'DM', 'aIkS572(Swy)', 'z0*XPx$', '8W', 'YGd(OO4R6)wEhw', 'I*Pnr*s*ZM&U', '(r
BC', 'eZ%kyG)!p', '@5AYT4TaMZ', '!EygrAQ04yTwx', '!6Mud', ')H', '^c2zKO2iCAe', 'FSU', 'Mk
8', 'nsDJY4', 'R5DdqcV', 'jGxcz^gs7oiIKk', '@bv9JRqfY1J', '9*hccx^', 't6xHQ#DSE6CbW', 'M6
', 'i', 'BM&$ApG', 'SjHr9S9CTrJ', 'CUYEc26NGR9', 'WYB', '$D!UwBiN', 'V(aKQo5Ej', 'Ey5fSnk
#!', 'SngshdO21u', 'G(z)Gxb16Bcc07', 'cUxlskgi)u', 'u7capt#0awzz', 'z*qc8g', 'd%@KFF@^o6x
)', 'C', '@)n4ffzT', '8H)Tj@a0CU$(g', '1S', 'QZgC^$24Hl@y#IK', 'FccjXvxfx#S*l', 'Nqz', '$
lK@vGK6', 'KShUcD9&S#', 'oLXs%xZBYMn', 'zJ&', 'CSBh&Gq', 'k0hIJ^HSnh*l89', 'NIwpr9DB', 'j
o@SnTn0F', 'T', '0tyLBxZ', 'A^yw0I3XyxuzU8p', '1', 'n$1C!mfT!', 'h', '^H*baMtyOguN^', '8p
```

```
LC#Udhpw$S(', 'PL7', '0cAYi*1ArO', '1AG1T^MZdAYhTNu', '9^REjDBXaTL&*zH', 'ajB9', '8pFjE',
'ROByLhcnF', 'yv0E1U(D', '(j)', '$0iGn1m9', 'w#g', 'ldpq1)Jsc*c9Ut', '2SY^Qu8xf0', '3hO5q
ifJgJ', '2', '4', 's96s', '9al@lEOSIW', 'oaP2J', 's1ptlU*z', '6HJ', '2Ci', '8IGN', 'dB8H&
TdHv', 'KkoIY', 'MY#^', 'uwulkzBlf', 'plFYQcGj3', 'CJxgX7Lu', 'n^cBJU@', '*2^F^y*', 'Q3vy
a(!CrC2&G', 'obv0yh', '@E', '1R4odW8DJ', 'QtWtwIoXFgl8', '!mSid(3de', '3m', 'YcX7lUzkxv',
'Jpg$aLf', '9', '58&XgRjIKI', ')Ve3XZBRoz8AI', '!N*KVt', '8', '%D&rNBvHVQ', '4#x&q8kdIbow
cNJ', 'kSEnYXm', 'MNqfZZr3pzx', 'wpBTAG1Vz3IZY$', 'yX4K', 'd', '7))2KoYfvqWJ', '8Xc@OekN)
xy5&', 'UFNFt@ch', 'f!8U', 'yq(igC*b@m4oAO!', 'wMO', '6zQ', 'D@Ek84HEgF^4*DO', 'GEwGex!h3
', 'SLCK', 'bVmmn5bZEVwy', 'bSf#Fo0eklRl&j', '8WNB(j', 'ns!JS*&Wg', '2bKsnl$GcT6d', 'mBSE
#ekA', 'A2cLk&Z7U4RWTW', '(!0q)(uC3', 'YdlHAQT2vj', 'UYX@7oM)(PdW1wz', 'lmd', 'BDhlK^k9',
'r7HJZXJ', 'WNzAiA*xhEon', 'enF', 'z!V63njp1x', 'S(0', 'HD3oYUTg3YXB', 'YCcYB', '5T%', 't
&', 'agBvVFEFI', 'vn!6s', 'vR*(!A', 'DR$OG)rNpzq(', 'i%', 'I', 'dU%d', 'L', 'iA4p', 'd5%'
, 'UU&WUPM&g7zmvPW', 'hs(68$bgeEGlv!', '@C$Gs#m', 'u!*82ZFmNIK', '&BiUm4Y', 'Qt)#91%2i',
'YFxdS6&N57', 'tq3Er^', 'x3Hw(Q2', '8rszgvO*', 'mF%', '6', 'EqP6$6RUhN', 'br0t3', 'cXD90#
', 'MdL^sPeLno5Y', 'v', 'COmy5rGNsFb', 'Ki7kbXGa64lZ!', 'hPaE@Pvx(', 'JG', 'I3iKY!HtAOE#'
, ')m9cy8khVh3F2', 'Yx', 'RRqdX', 'F0', 'sl*OIAw0', '5a4QHa2@GakPOq', ')nwJO']
```

Etapa 3: Algoritmo para verificar la fortaleza de las contraseñas generadas

In [75]:

```
##Libreria para expresiones regulares
import re

##Para este programa, tendremos en cuenta el criterio mencionado anteriormente
def Verificar(password):

    ## Criterio longitud: 8 o mayor
    longitud_invalida = len(password) < 8

    ## Criterio números: 1 o más
    digitos_invalido = re.search(r"\d", password) is None

    ## Criterio mayuscula: 1 o más
    Mayuscula_invalida = re.search(r"[A-Z]", password) is None

    ## Criterio minuscula: 1 o más
    Minuscula_invalida = re.search(r"[a-z]", password) is None

    ## Criterio simbolos: 1 o más
    simbolos_invalido = re.search(r"\W", password) is None

    ## Validez
    password_valida = not ( longitud_invalida or digitos_invalido or Mayuscula_invalida
or Minuscula_invalida or simbolos_invalido )

    ##En caso de que uno de los 5 criterios no se cumpla, arrojara "true" o verdadero, ya
que el criterio es invalido.
    ##De igual forma, si se cumplen todos los 5 criterios (en fake), la contraseña es val
ida y arrojara "true" o verdadero.
    return {
        'Contraseña valida' : password_valida,
        'longitud invalida' : longitud_invalida,
        'digitos_invalido' : digitos_invalido,
        'Mayuscula_invalida' : Mayuscula_invalida,
        'Minuscula_invalida' : Minuscula_invalida,
        'simbolos_invalido' : simbolos_invalido,
    }
```

Algoritmo para realizar el mapeado

In [80]:

```
#Este metodo es el mismo que el anterior, pero su objetivo es catalogar las contraseñas e
n 1(validas), 0(invalidas)
#Para de este modo, poder realizar el mapeado de las mismas

def Verificar_valor(password):
```

```
    ## Criterio longitud: 8 o mayor
    longitud_invalida = len(password) < 8

    ## Criterio números: 1 o más
    digitos_invalido = re.search(r"\d", password) is None

    ## Criterio mayuscula: 1 o más
    Mayuscula_invalida = re.search(r"[A-Z]", password) is None

    ## Criterio minuscula: 1 o más
    Minuscula_invalida = re.search(r"[a-z]", password) is None

    ## Criterio simbolos: 1 o más
    simbolos_invalido = re.search(r"\W", password) is None

    ## Validez
    password_valida = not ( longitud_invalida or digitos_invalido or Mayuscula_invalida
or Minuscula_invalida or simbolos_invalido )

    if password_valida:
        return 1
    else:
        return 0
```

Etapa 4: Verificación de la fortaleza de las contraseñas generadas

In [85]:

```
##lista de contraseñas verificadas
verificados = []
u = 0
##Verificaremos cada contraseña de la lista generada aleatoriamente, y la añaderemos a la
lista de verificados
for x in lista:
    ##Mostraremos 25 contraseñas para confirmar
    if u <26:
        u +=1
        print("Contraseña: ",x)
        print(Verificar(x))

    ##Agregaremos a la lista de verificados, el valor de las contraseñas
    ##1 si son validas, 0 si son invalidas
    verificados.append(Verificar_valor(x))
```

```
Contraseña:  vYxSxl1ewpX
{'Contraseña valida': False, 'longitud invalida': False, 'digitos_invalido': False, 'Mayu
scula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': True}
Contraseña:  LH0T3)e
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  u!%QrJc8np$^
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  AbmN*42HM2
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  @s0
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': False, 'Mayus
cula_invalida': True, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  ePF
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': True}
Contraseña:  $BP
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': True, 'simbolos_invalido': False}
Contraseña:  G5Zh$lTI9
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  N)Lfg@Obuu
{'Contraseña valida': False, 'longitud invalida': False, 'digitos_invalido': True, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
```

```
Contraseña:  RvB@!t)#VWm1MR
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  Pw0qlm@%oQABzka
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  32OQ
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': True, 'simbolos_invalido': True}
Contraseña:  #
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': True, 'Minuscula_invalida': True, 'simbolos_invalido': False}
Contraseña:  G93
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': True, 'simbolos_invalido': True}
Contraseña:  j7H%Uv@8b@5^i
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  O0CDF0&6acGwTD8
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  jKN&KTEH3^
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  Pc@&^U%H
{'Contraseña valida': False, 'longitud invalida': False, 'digitos_invalido': True, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  kLw
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': True}
Contraseña:  !AP^#2&a
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  zBq
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': True}
Contraseña:  oEvDj2lMh8V(U
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  16ILEQ5kAD@L9
{'Contraseña valida': True, 'longitud invalida': False, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  $Rb
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': False}
Contraseña:  AN
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': True, 'Mayusc
ula_invalida': False, 'Minuscula_invalida': True, 'simbolos_invalido': True}
Contraseña:  e3FIA
{'Contraseña valida': False, 'longitud invalida': True, 'digitos_invalido': False, 'Mayus
cula_invalida': False, 'Minuscula_invalida': False, 'simbolos_invalido': True}
```

Etapa 5: Mapeo de las contraseñas procesadas anteriormente

In [86]:

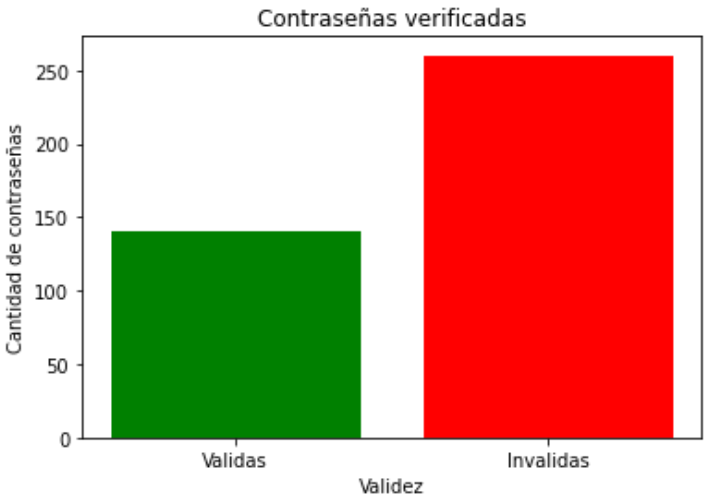
```
#Libreria para graficar
import matplotlib.pyplot as plt

#Listas para separar las contraseñas
validas = []
Invalidas = []

#Proceso de separación en base a su validez, 0 para invalidas, 1 para validas
for x in verificados:
    if x == 0:
        Invalidas.append(x)
    else:
        validas.append(x)
```

```
print(len(validas))
print(len(Invalidas))
#GRAFICA
x = ["Validas", "Invalidas"]
y = [len(validos), len(Invalidos)]
plt.bar(x, y, color = ['g','r'])
plt.title("Contraseñas verificadas")
plt.ylabel('Cantidad de contraseñas')
plt.xlabel('Validez')
plt.show()
```

```
140
260
```



Conclusión

Con este proyecto hemos podido simular con un algoritmo aleatorio las contraseñas ingresadas en una página web. Dichas contraseñas las hemos verificado con el criterio anti-fuerza bruta planteado. Como hemos visto al final del proyecto, 140 contraseñas resultaron ser validas, lo que apenas es un 35 % de 400, una cifra preocupante, mientras que 260 resultaron ser invalidas, lo que resulta ser un 65%. Lo que demuestra lo desprotegidas que están las personas que se registren en páginas web que no implementen algoritmos como este que acabo de desarrollar.

Con esto me despido, y espero que hayan disfrutado de este proyecto tanto como lo he hecho yo.