

```
Password brute-force algorithm
Python 3
import string
from itertools import product
from time import time
from numpy import loadtxt
easy dB = {'admin': 'admin'}
```

- Using the 'numpy' library
 - Good w/ large data
 - provides a high-performance multidimensional array object, and tools for working with
- Database data structure is a dict.
 - Key = username
 - Value = password

.

show_all_usrs():

- Will show all valid users in database
- Users IDed by key

```
def show_all_usrs():
    print('\n')
    print(' USERS__')
    for key in easy_dB.keys():
        print(key)
    print('\n')
    menu()
```

- add usr():
 - Will add a user to the database
 - If username != already exists
 - If password == 5 chars
 - easy_db[usr]= pwd
 - Key -> user
 - Value user -> pwd

```
def add_usr():
    usr = input("\nEnter username: ")
    if (usr in easy_dB.keys()):
        print('\nUser already exists\n')
    else:
        pwd = input("Enter password (5 characters only): ")
        if (len(pwd)== 5):
            easy_dB[usr] = pwd
            print('\nUSER_CREATED\n')
            menu()
    else:
        print('\nInvalid_password\nUSER_NOT_CREATED\n')
        menu()
```

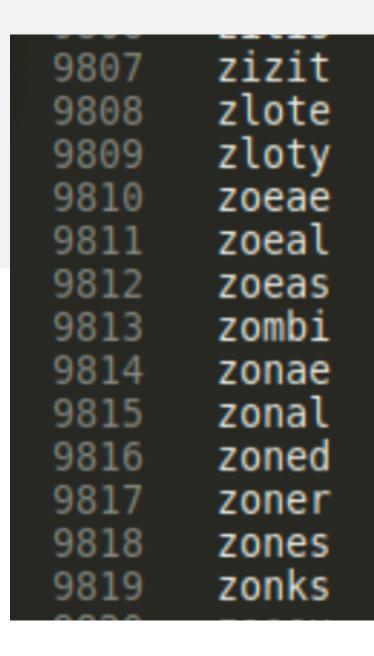
Permutations

- A Permutation is an ordered Combination
- For this program
 - Password = 5 chars \rightarrow ___ __ __
 - #of letters in alphabet == 26 *2 (to include uppercase)
 - #of digits in base 10 decimal number == 10 (0-9)
 - #of possibilities for 1st char = 26*2 + 10 == 62
 - #of possibilities for 5 char password → 62*62*62*62
 - 625 possible combinations (916,132,832 combinations)
 - IT TAKES A LOT OF TIME + MEMORY TO DO EVERY COMBINATION
 - A 'dictionary attack' helps to reduce time

The BruteForce

```
def bruteforce(username, max nchar=5): #change max nchar to accept larger pass
  password = easy dB[username]
  print('\n1) Comparing with most common passwords / first names')
  common pass = loadtxt('probable-v2-top12000.txt', dtype=str)
   common names = loadtxt('middle-names.txt', dtype=str)
   cp = [c for c in common pass if c == password]
  cn = [c for c in common names if c == password]
   cnl = [c.lower() for c in common names if c.lower() == password]
  if len(cp) == 1:
     '\tPASSWORD:', cp)
     return cp
  if len(cn) == 1:
     '\tPASSWORD:', cn)
     return cn
  if len(cnl) == 1:
     '\tPASSWORD:', cnl)
     return cnl
```

- Bruteforce(username, max_nchar=5):
 - Will take in a username and
 - Try every possible combination until passwords match
 - 1st part is a dictionary attack not bruteforce
 - Used to save time



1026	baloo		
1027	ronny		
1028	robyn		
1029	pasha		
1030	owner		
1031	owned		
1032	order		
1033	orbit		
1034	nikko		
1035	mogul		
1036	jjjjj		
1037	jessy		
1038	hawks		
1039	hands		
1040	gecko		
1041	ester		
1042	eeeee		
1043	dingo		
1044	daryl		
1045	ccccc		
1046	bonny		
1047	blast		
1048	avril		
1049	ashes		
1050	angle		
1051	alive		
1052	1052A		

2 Decent Sized Text Files

- probable-v2-top12000.txt
 - < 1000 entries
- middle-names.txt
 - < 9800 entries
- Total entries ~ 10k entries

The Hard Work

- 2) Digits Cartesian Product
 - Tries all possible number combinations
- 3) ASCII Lowercase + Digits
 - Tries all possible lowercase + digit combinations
- 4) ASCII Uppercase/Lowercase + Digits
 - Tries all possible uppercase +
 lowercase + digit combinations

```
print('\n2) Digits Cartesian Product')
for l in range(max nchar, max nchar + 1):
    generator = product(string.digits, repeat=int(l)
    print("\t...%d digit" % l)
    p = product loop(password, generator, username)
    if p is not False:
        return p
print('\n3) ASCII Lowercase + Digits')
for l in range(max nchar, max nchar + 1):
    print("\t...%d char" % l)
    generator = product(string.ascii lowercase + str
                        repeat=int(l))
     = product loop(password, generator, username)
    if p is not False:
        return p
print('\n4) ASCII Uppercase/Lowercase + Digits')
all char = string.ascii uppercase + string.digits
for l in range(max nchar, max nchar + 1):
    print("\t...%d char" % l)
    generator = product(all char, repeat=int(l))
       product loop(password, generator, username)
      p is not False:
```