

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІКТА кафедра БІТ



МУЛЬТИПОТОКОВІСТЬ ТА МУЛЬТИПРОЦЕСОРНІСТЬ В PYTHON

Звіт до лабораторної роботи № 1
з навчальної дисципліни: “Програмування скриптовими мовами.”
Варіант №6

виконав:
студент групи КБ-212
Зінько А.В.
перевірив:
Гасілін Д.Л.

Львів-2025

Мета роботи. ознайомитись з модулями стандартної бібліотеки Python для мультипоточкового і мультипроцесорного програмування на прикладі brute-force атаки на пароль.

Хід роботи.

1. Вивести інформацію про кількість процесорних ядер n `cpu` для вашого ПК.

```
import multiprocessing

print(multiprocessing.cpu_count())

>>> 16
```

2. Здійснити підбір пароля заданої довжини, що використовує заданий набір символів та хеш-функцію (див. табл. 1), за його хешем (табл. 2).

Варіант	Довжина, символів	Набір символів	Хеш-функція
6.	6	lowChar + numChar + spcChar	blake2b

Варіант	Хеш паролю
6.	6f0874bd7ba418106ac15555ea927aef34bc05c8ba92cd2e4c3065e7751ccc125fd88f19eec18f007e9f033c8dd2749edf573ac8aeec47d073f5832553fcea9c

2.1. Не використовуючи мультипроцесорність. Виміряти і вивести час підбору паролю та знайдений пароль (підбір деяких паролів може зайняти декілька годин у випадку використання одного ядра).

Текст програми:

```
import os
import hashlib
from time import time
import multiprocessing

targetHash6 =
'6f0874bd7ba418106ac15555ea927aef34bc05c8ba92cd2e4c3065e7751ccc125fd88f19eec18f007e9f033c8dd2749edf573ac8aeec47d073f5832553fcea9c' #blake2b
uppChar = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
lowChar = "abcdefghijklmnopqrstuvwxyz"
numChar = "0123456789"
spcChar = "!@#$%^&*_- "
char6 = lowChar + numChar + spcChar
found = False

def findPassSixth():
    found = False
    start = time()
    tries = 0
    for i in range(len(char6)):
        for j in range(len(char6)):
            for k in range(len(char6)):
                for l in range(len(char6)):
                    for m in range(len(char6)):
                        for n in range(len(char6)):
                            temp_pass = char[i] + char[j] + char[k] + char[l] +
char[m] + char[n]
                            temp_hash_obj = hashlib.blake2b(temp_pass.encode('utf-8'))
                            digest = temp_hash_obj.hexdigest()
```

```

        tries += 1
        if digest == targetHash6:
            print(f"password {temp_pass} found in {tries} tries!")
            found = True
            break
        if found == True:
            break
        if found == True:
            break
        if found == True:
            break
        if found == True:
            break
        if found == True:
            break
    finish = time()
    elapsed = finish - start
    print(f"Password{temp_pass} found in {tries} tries!")
    print(f"Time elapsed: {elapsed:.4f} seconds")

```

Результат виконання програми:

```

Password *e5-qp found in 8,877,431,180 tries!
Time elapsed: 5877.1466 seconds

For testing - 4 character search space: 4,477,456 combinations

System info:
CPU cores: 16
Estimated speedup: 16x (theoretical maximum)
PS C:\Users\Dream\Desktop\script> 

```

2.2. З використанням Process або Pool (використовуючи число процесів рівне числу ядер вашого комп'ютера). Виміряти і вивести час підбору паролю та знайдений пароль. Якщо пароль знайдено одним з процесів, то він повинен передаватися в основну програму, а виконання процесів повинно завершуватися.

Текст програми:

```

import os
import hashlib
import multiprocessing
from multiprocessing import Pool
from time import time
targetHash6 =
'ca451287a87b2163cf8bcd63a4a46bd3c9ff7638f0e3a1f81e4ef9ca9c0ed8fef7150ffa06c0436e89a6c
668d2c5a7183015627601bb8baefd6a49ba05ab5daf' #blake2b accccc
uppChar = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
lowChar = "abcdefghijklmnopqrstuvwxyz"
numChar = "0123456789"
spcChar = "!@#$%^&* _ -"
char6 = lowChar + numChar + spcChar

```

```

test_password2 = 'accccc'
def worker_process(i_range):
    """Worker process that handles one range of the first character"""
    tries = 0
    start_i, end_i = i_range

    for i in range(start_i, end_i):
        for j in range(len(char6)):
            for k in range(len(char6)):
                for l in range(len(char6)):
                    for m in range(len(char6)):
                        for n in range(len(char6)):
                            temp_pass = char6[i] + char6[j] + char6[k] + char6[l] +
char6[m] + char6[n]
                            temp_hash_obj = hashlib.blake2b(temp_pass.encode('utf-8'))
                            digest = temp_hash_obj.hexdigest()
                            tries += 1

                            if tries % 1_000_000 == 0:
                                print(f"Process {start_i}-{end_i}: {tries} tries,
current: {temp_pass}")

                            if digest == targetHash6:
                                return temp_pass

    return None

def findPassSixthMultiprocess():
    start = time()

    num_processes = multiprocessing.cpu_count()
    chars_per_process = len(char6) // num_processes

    ranges = []
    for p in range(num_processes):
        start_i = p * chars_per_process
        if p == num_processes - 1:
            end_i = len(char6)
        else:
            end_i = (p + 1) * chars_per_process
        ranges.append((start_i, end_i))

    print(f"Using {num_processes} processes")
    print(f"Character ranges: {ranges}")

    with Pool(processes=num_processes) as pool:
        results = pool.map(worker_process, ranges)

    found_password = None
    for result in results:
        if result is not None:
            found_password = result
            break

    finish = time()

```

```

elapsed = finish - start

if found_password:
    print(f"Password {found_password} found!")
else:
    print("Password not found in search space")

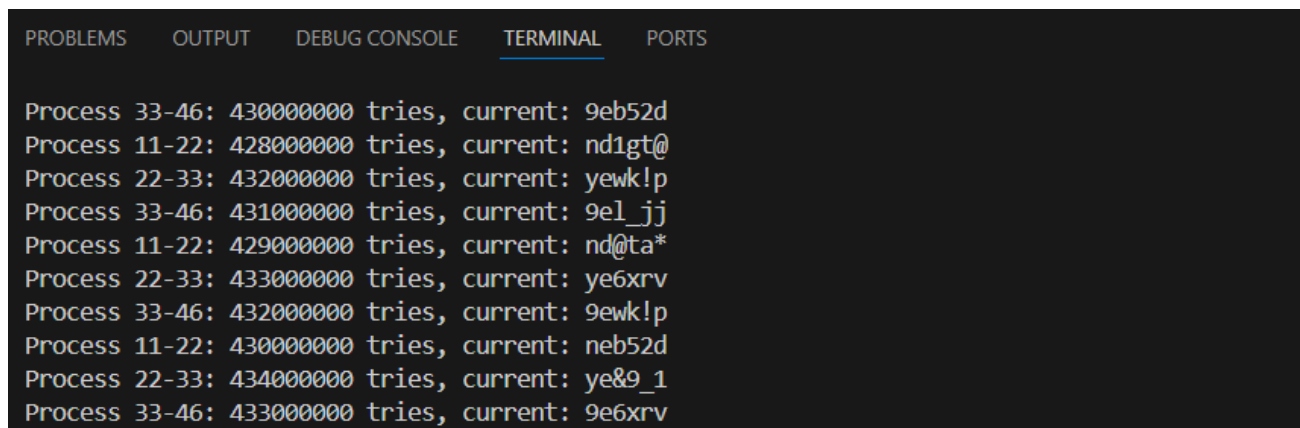
print(f"Time elapsed: {elapsed:.4f} s")
return found_password

if __name__ == "__main__":
    print(f"CPU cores available: {multiprocessing.cpu_count()}")
    print("Running with multiprocessing:")
    findPassSixthMultiprocess()

# print("\nRunning single-threaded:")
# findPassSixth()

```

Результат виконання програми



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Process 33-46: 430000000 tries, current: 9eb52d
Process 11-22: 428000000 tries, current: nd1gt@
Process 22-33: 432000000 tries, current: yewk!p
Process 33-46: 431000000 tries, current: 9el_jj
Process 11-22: 429000000 tries, current: nd@ta*
Process 22-33: 433000000 tries, current: ye6xrv
Process 33-46: 432000000 tries, current: 9ewk!p
Process 11-22: 430000000 tries, current: neb52d
Process 22-33: 434000000 tries, current: ye&9_1
Process 33-46: 433000000 tries, current: 9e6xrv

```

Висновок. При виконанні цієї лабораторної роботи я ознайомився з модулями стандартної бібліотеки Python для мультипоточкового і мультипроцесорного програмування на прикладі brute-force атаки на пароль. Також виконав підбір паролю шляхом brute-force, за допомогою написаного генератора, порівнюючи кожен хеш із хешом шуканого паролю (із мультипроцесорністю і без).