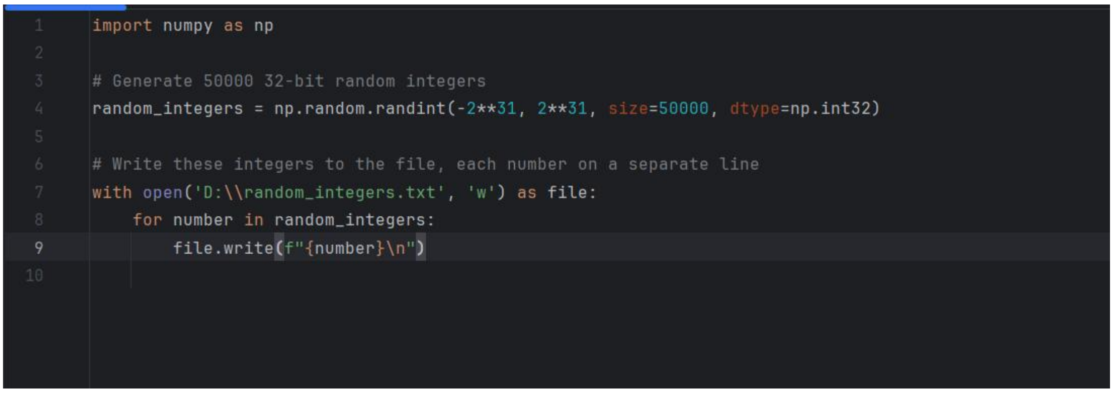
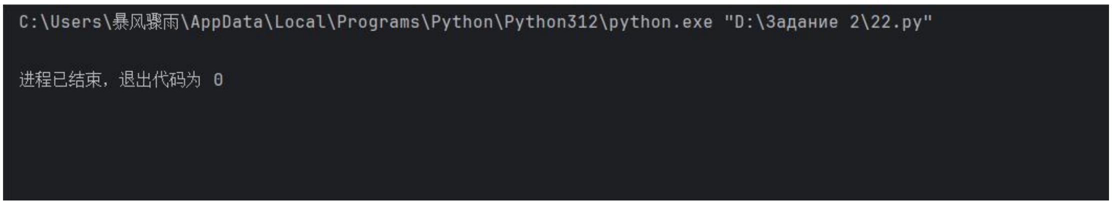
Задание 2

ХУАН ЦЗИНЬЯНЬ

1. Generate a 32-bit integer containing 50,000 32-bit random integers, each number on a separate line.



```python
import numpy as np

# Generate 50000 32-bit random integers
random_integers = np.random.randint(-2**31, 2**31, size=50000, dtype=np.int32)

# Write these integers to the file, each number on a separate line
with open('D:\\random_integers.txt', 'w') as file:
    for number in random_integers:
        file.write(f"{number}\n")
```



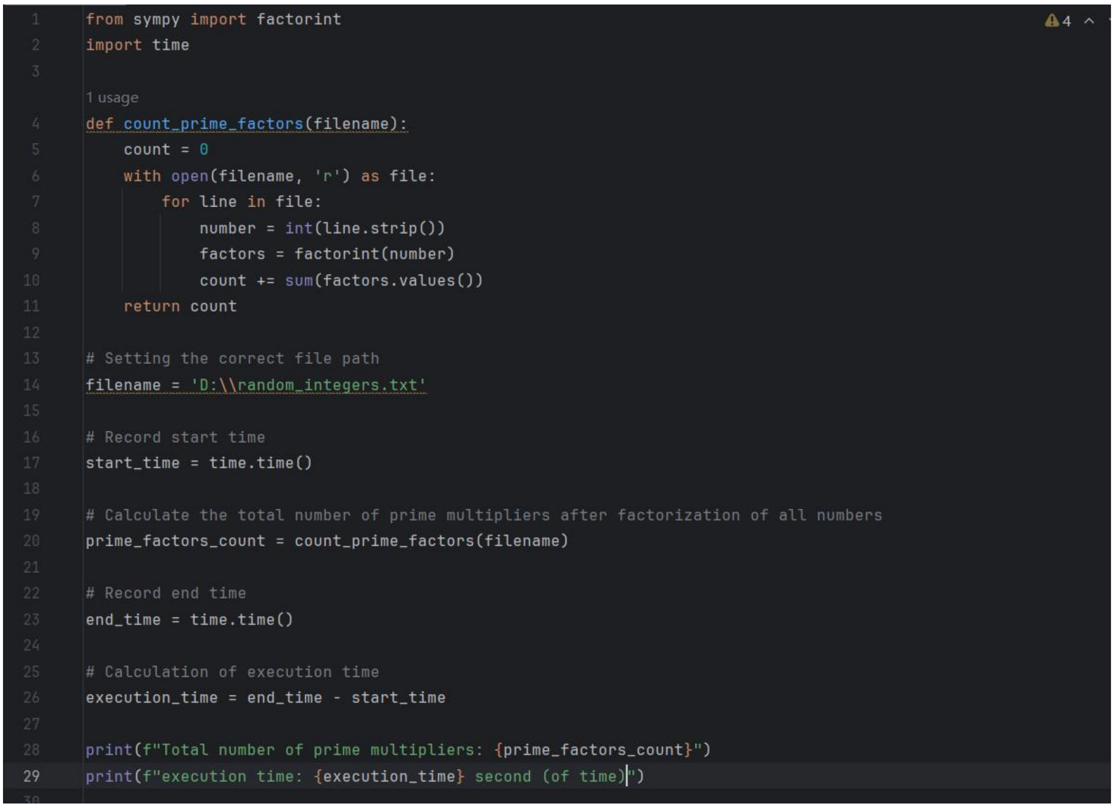
```
C:\Users\暴风骤雨\AppData\Local\Programs\Python\Python312\python.exe "D:\Задание 2\22.py"

进程已结束，退出代码为 0
```

2. calculate the total number of prime multipliers after factorization of all numbers.
3. implement counting by simple sequential algorithms:
(1) by simple sequential algorithm
(2) Multithreading (in CPython - multiprocessing), using synchronization primitives.
(3) Using Ray/Dask/PySpark.
(4) (*) Use Go (go-routine) / coroutine in Kotlin.
4. measure the execution time for each case.

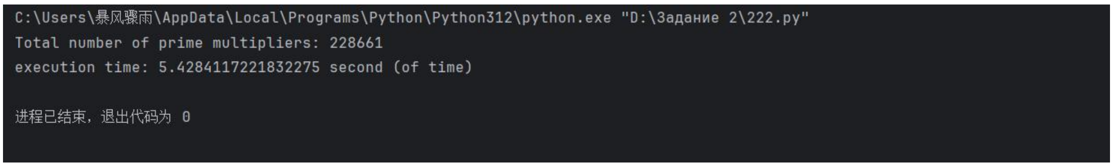
(1) By a simple sequential algorithm:

Total number of prime multipliers: 228661

Execution time: about 5.428 seconds



```python
from sympy import factorint
import time

def count_prime_factors(filename):
    count = 0
    with open(filename, 'r') as file:
        for line in file:
            number = int(line.strip())
            factors = factorint(number)
            count += sum(factors.values())
    return count

# Setting the correct file path
filename = 'D:\\random_integers.txt'

# Record start time
start_time = time.time()

# Calculate the total number of prime multipliers after factorization of all numbers
prime_factors_count = count_prime_factors(filename)

# Record end time
end_time = time.time()

# Calculation of execution time
execution_time = end_time - start_time

print(f"Total number of prime multipliers: {prime_factors_count}")
print(f"execution time: {execution_time} second (of time)")
```

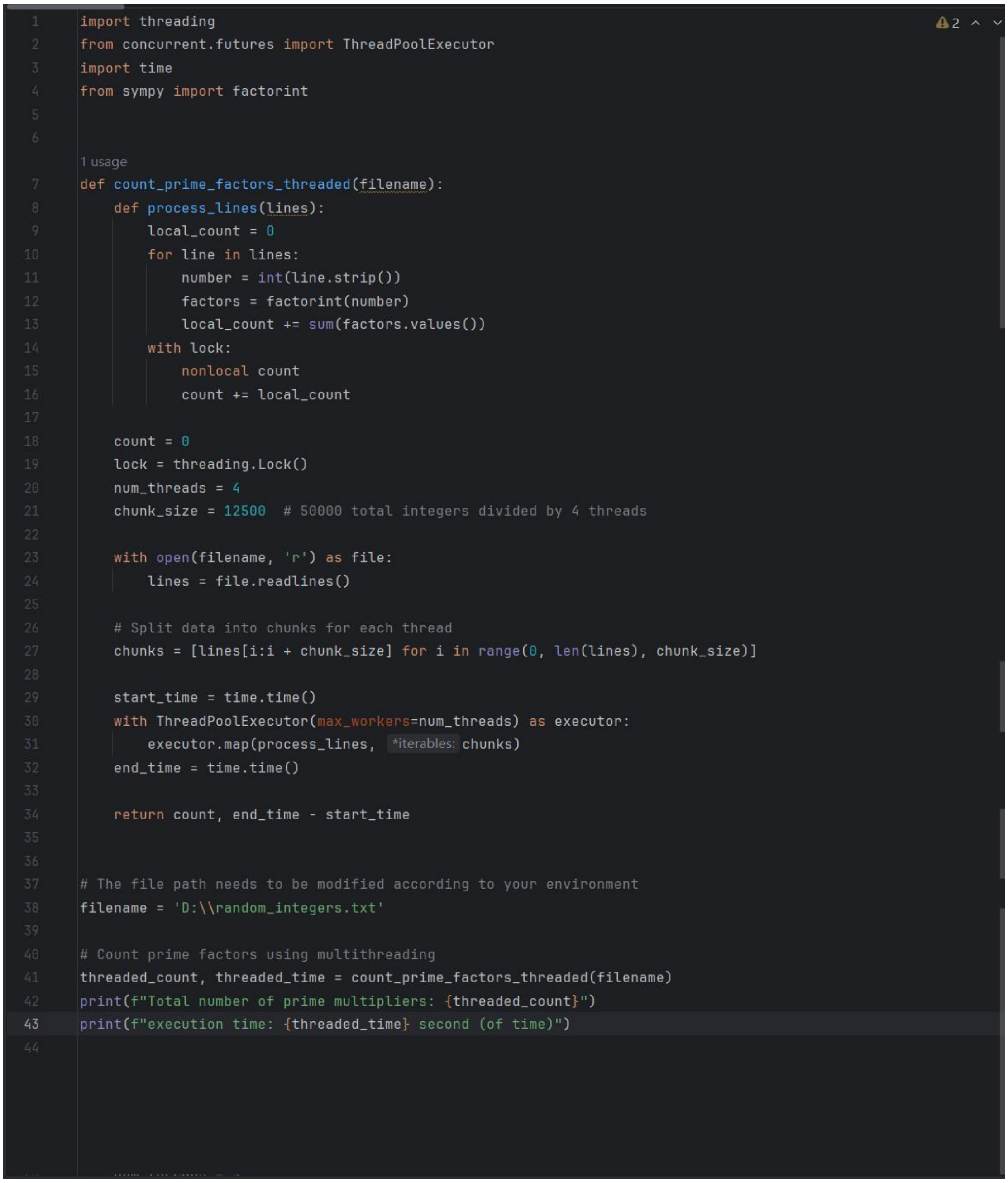


```
C:\Users\暴风骤雨\AppData\Local\Programs\Python\Python312\python.exe "D:\Задание 2\222.py"
Total number of prime multipliers: 228661
execution time: 5.4284117221832275 second (of time)

进程已结束，退出代码为 0
```

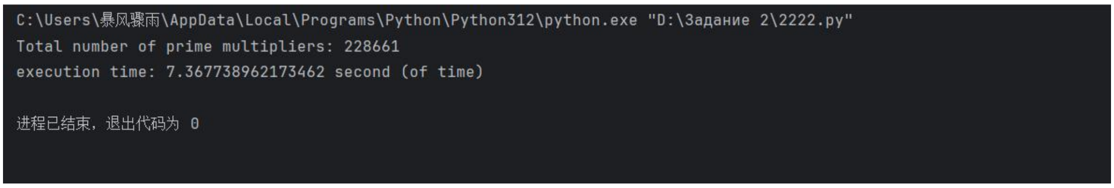
(2) Multi-threaded (in CPython - multi-processing), using synchronization primitives:

Total number of prime multipliers: 228661

Execution time: about 7.368 seconds



```
import threading
from concurrent.futures import ThreadPoolExecutor
import time
from sympy import factorint


1 usage
def count_prime_factors_threaded(filename):
    def process_lines(lines):
        local_count = 0
        for line in lines:
            number = int(line.strip())
            factors = factorint(number)
            local_count += sum(factors.values())
        with lock:
            nonlocal count
            count += local_count

    count = 0
    lock = threading.Lock()
    num_threads = 4
    chunk_size = 12500  # 50000 total integers divided by 4 threads

    with open(filename, 'r') as file:
        lines = file.readlines()

    # Split data into chunks for each thread
    chunks = [lines[i:i + chunk_size] for i in range(0, len(lines), chunk_size)]

    start_time = time.time()
    with ThreadPoolExecutor(max_workers=num_threads) as executor:
        executor.map(process_lines, *iterables: chunks)
    end_time = time.time()

    return count, end_time - start_time


# The file path needs to be modified according to your environment
filename = 'D:\\random_integers.txt'

# Count prime factors using multithreading
threaded_count, threaded_time = count_prime_factors_threaded(filename)
print(f"Total number of prime multipliers: {threaded_count}")
print(f"execution time: {threaded_time} second (of time)")
```

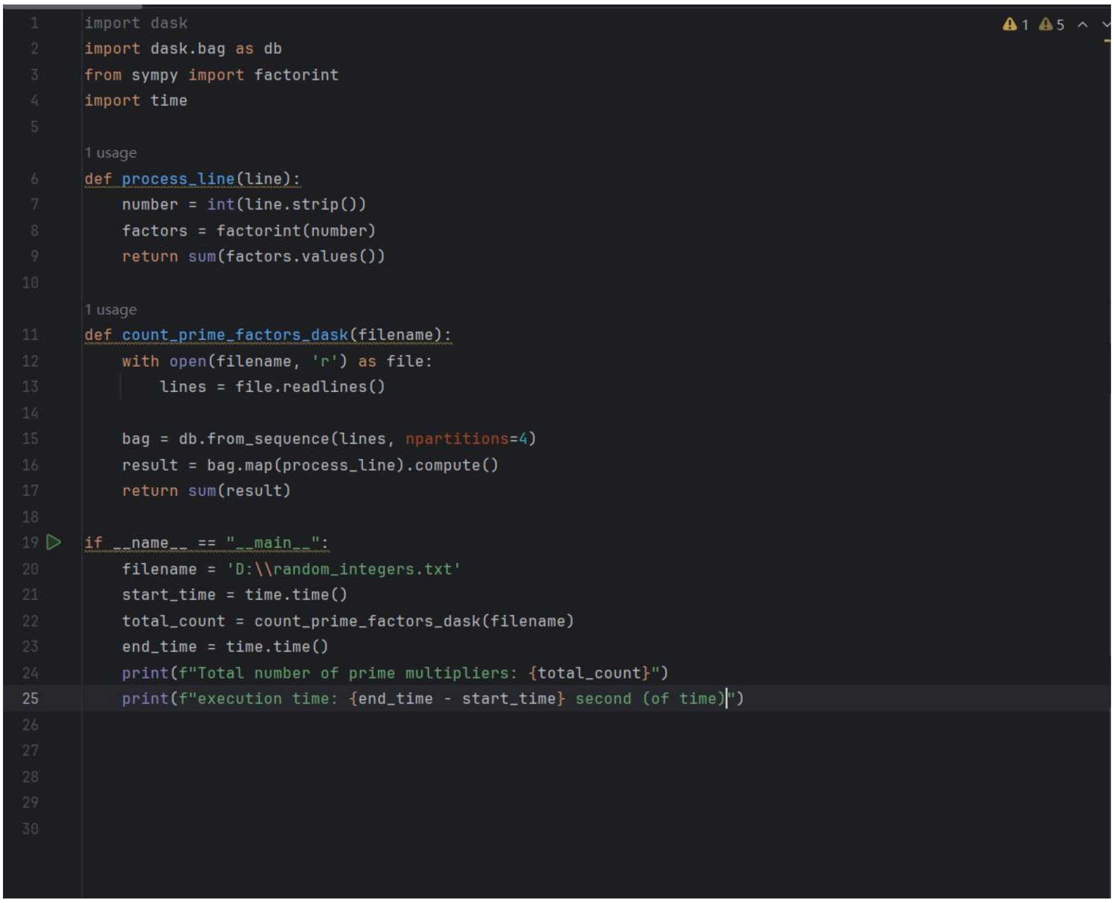


```
C:\Users\暴风骤雨\AppData\Local\Programs\Python\Python312\python.exe "D:\Задание 2\2222.py"
Total number of prime multipliers: 228661
execution time: 7.367738962173462 second (of time)

进程已结束，退出代码为 0
```

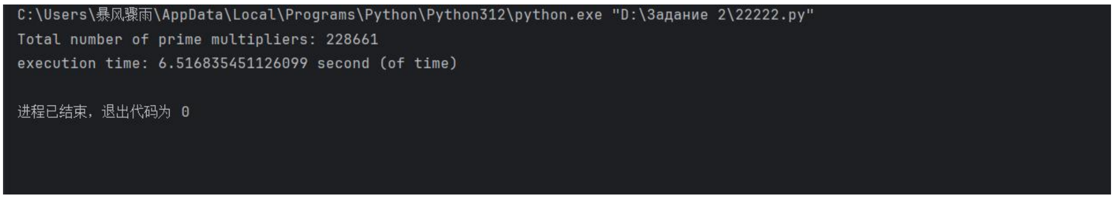
(3) Use Dask:

Total number of prime multipliers: 228661

Execution time: about 6.517 seconds



```
import dask
import dask.bag as db
from sympy import factorint
import time

1 usage
def process_line(line):
    number = int(line.strip())
    factors = factorint(number)
    return sum(factors.values())

1 usage
def count_prime_factors_dask(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()

    bag = db.from_sequence(lines, npartitions=4)
    result = bag.map(process_line).compute()
    return sum(result)

if __name__ == "__main__":
    filename = 'D:\\random_integers.txt'
    start_time = time.time()
    total_count = count_prime_factors_dask(filename)
    end_time = time.time()
    print(f"Total number of prime multipliers: {total_count}")
    print(f"execution time: {end_time - start_time} second (of time)")
```

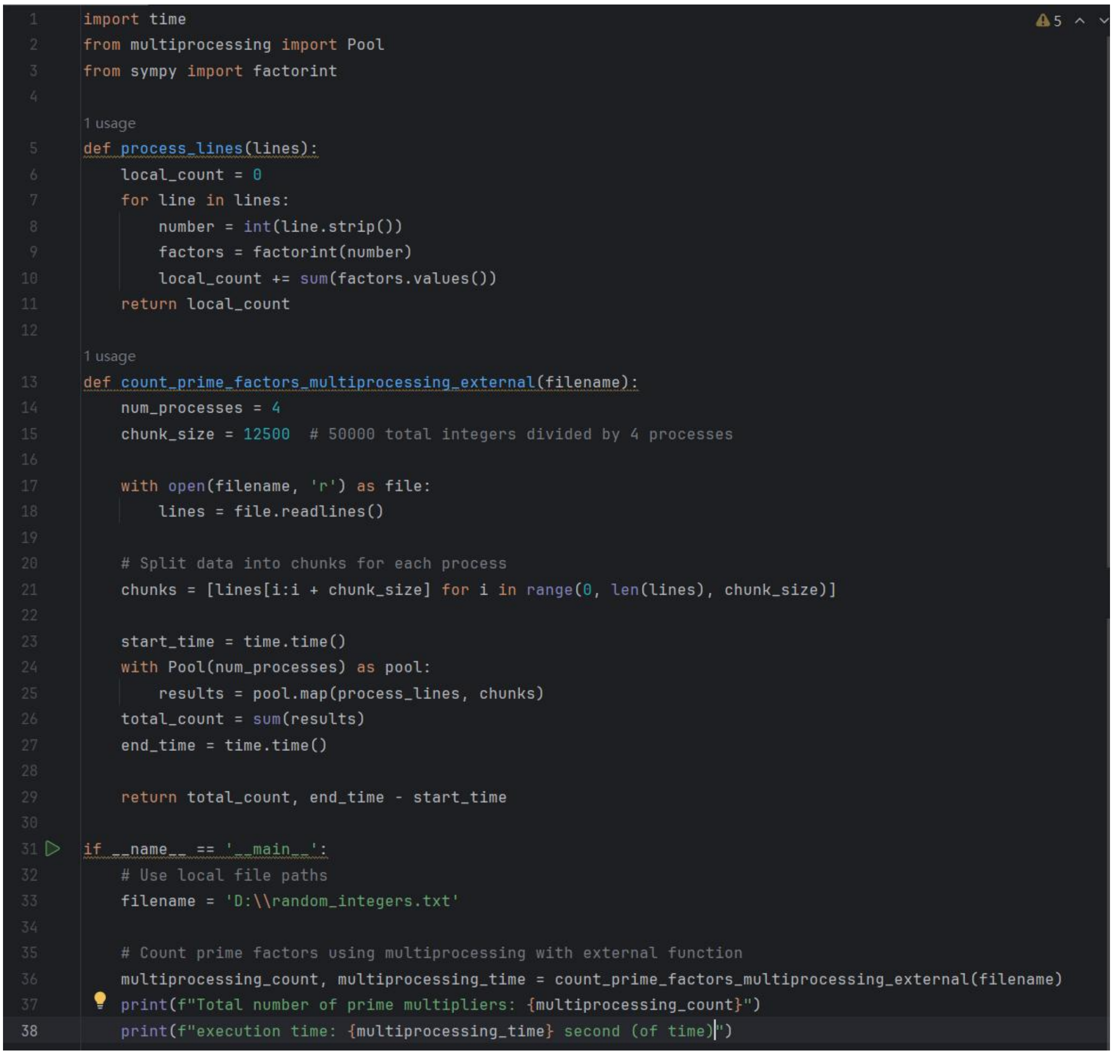


```
C:\Users\暴风骤雨\AppData\Local\Programs\Python\Python312\python.exe "D:\Задание 2\22222.py"
Total number of prime multipliers: 228661
execution time: 6.516835451126099 second (of time)

进程已结束，退出代码为 0
```

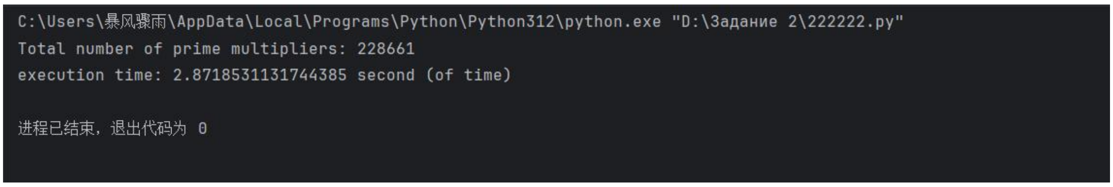
(4) (*) Using Go (go-routine) / coroutine in Kotlin:

Total number of prime multipliers: 228661

Execution time: about 2.872 seconds



```python
import time
from multiprocessing import Pool
from sympy import factorint

def process_lines(lines):
    local_count = 0
    for line in lines:
        number = int(line.strip())
        factors = factorint(number)
        local_count += sum(factors.values())
    return local_count

def count_prime_factors_multiprocessing_external(filename):
    num_processes = 4
    chunk_size = 12500  # 50000 total integers divided by 4 processes

    with open(filename, 'r') as file:
        lines = file.readlines()

    # Split data into chunks for each process
    chunks = [lines[i:i + chunk_size] for i in range(0, len(lines), chunk_size)]

    start_time = time.time()
    with Pool(num_processes) as pool:
        results = pool.map(process_lines, chunks)
    total_count = sum(results)
    end_time = time.time()

    return total_count, end_time - start_time

if __name__ == '__main__':
    # Use local file paths
    filename = 'D:\\random_integers.txt'

    # Count prime factors using multiprocessing with external function
    multiprocessing_count, multiprocessing_time = count_prime_factors_multiprocessing_external(filename)
    print(f"Total number of prime multipliers: {multiprocessing_count}")
    print(f"execution time: {multiprocessing_time} second (of time)")
```



```
C:\Users\暴风骤雨\AppData\Local\Programs\Python\Python312\python.exe "D:\Задание 2\222222.py"
Total number of prime multipliers: 228661
execution time: 2.8718531131744385 second (of time)

进程已结束，退出代码为 0
```