Experiment No. 12
Demonstrate the concept of Multi-threading
Date of Performance:
Date of Submission:

Experiment No. 12

Title: Demonstrate the concept of Multi-threading

Aim: To study and implement the concept of Multi-threading

Objective: To introduce the concept of Multi-threading in python

Theory:

Thread

In computing, a **process** is an instance of a computer program that is being executed. Any process has 3 basic components:

- An executable program.
- The associated data needed by the program (variables, work space, buffers, etc.)
- The execution context of the program (State of process)

A **thread** is an entity within a process that can be scheduled for execution. Also, it is the smallest unit of processing that can be performed in an OS (Operating System).

In simple words, a **thread** is a sequence of such instructions within a program that can be executed independently of other code. For simplicity, you can assume that a thread is simply a subset of a process!

A thread contains all this information in a **Thread Control Block (TCB)**:

- **Thread Identifier:** Unique id (TID) is assigned to every new thread
- **Stack pointer:** Points to thread's stack in the process. Stack contains the local variables under thread's scope.
- **Program counter:** a register which stores the address of the instruction currently being executed by thread.
- **Thread state:** can be running, ready, waiting, start or done.
- **Thread's register set:** registers assigned to thread for computations.
- **Parent process Pointer:** A pointer to the Process control block (PCB) of the process that the thread lives on.

Code:

```
    import threading

import time



# Define a function to be executed by each thread

def print_messages(delay, message, stop_event):
```

```
    while not stop_event.is_set():
        print(message)
        time.sleep(delay)

# Create a shared flag variable to control thread termination
stop_flag = threading.Event()

# Create two threads
thread1 = threading.Thread(target=print_messages, args=(1, "Thread 1: Hello!", stop_flag))
thread2 = threading.Thread(target=print_messages, args=(2, "Thread 2: Hi!", stop_flag))

# Start the threads
thread1.start()
thread2.start()

# Let the threads run for a while
time.sleep(5)

# Set the stop flag to signal threads to stop
stop_flag.set()

# Wait for both threads to finish
thread1.join()
```

```
thread2.join()


print("Main thread exiting.")
```

Output:

Conclusion: Multithreading has been successfully implemented in python.