|  |
| --- |
| Experiment No. 12 |
| Demonstrate the concept of Multi-threading |
| Date of Performance: 08/04/2024 |
| Date of Submission: 15/04/2024 |

**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.

**Program:**

import threading

def print\_cube(num):

"""

Print the cube of a given number.

"""

cube = num \*\* 3

print(f"Cube of {num}: {cube}")

def print\_square(num):

"""

Print the square of a given number.

"""

square = num \*\* 2

print(f"Square of {num}: {square}")

if \_\_name\_\_ == "\_\_main\_\_":

# Create thread objects

t1 = threading.Thread(target=print\_square, args=(12,))

t2 = threading.Thread(target=print\_cube, args=(13,))

# Start the threads

t1.start()

t2.start()

# Wait for both threads to finish

t1.join()

t2.join()

# All threads executed

print("Done!")

**Ouput:**

Square of 12: 144

Cube of 13: 2197

Done!

**Conclusion:**

This experiment effectively illustrated the concept of multithreading in Python, demonstrating the simultaneous execution of multiple tasks within a single process. Leveraging the threading module, the program executed distinct functions concurrently, thereby improving computational efficiency. Through this experiment, the advantages of multithreading in enhancing program performance by harnessing parallelism were underscored. The successful integration of multithreading emphasizes its crucial role in optimizing resource utilization and augmenting program responsiveness.