

**PAF-KIET NORTH CAMPUS**

**OPERATING SYSTEM**

**CLASS ID: (100337)**

**PROJECT REPORT**

**EXPLORATION OF MULTITHREADING AND MULTITASKING IN PYTHON**

**GROUP MEMBER:**

UMNA ALEEM 61638

**Abstract:**

Python is famous among established researchers that esteem its effortlessness and power, particularly as it joins numeric libraries, for example, NumPy, SciPy, Dask, and Numba. As CPU center checks continue expanding, these modules can make utilization of numerous centers through multi-threading for productive multi-center parallelism. Be that as it may, strings can meddle with one another prompting overhead and wastefulness whenever utilized together in a solitary application on machines with countless. This execution misfortune can be anticipating if all multi-threading modules composed. Python acquaints more methodologies with coordination for both multithreading and multi-handling cases.

The Python threading module uses threads instead of processes. Threads uniquely run in the same unique memory heap. Whereas Processes run in separate memory heaps. This makes sharing information harder with processes and object instances. One problem arises because threads use the same memory heap, multiple threads can write to the same location in the memory heap, which is why the global interpreter lock (GIL) in CPython was, created as a mutex to prevent it from happening.

**Introduction:**

The multithreading library is lightweight, shares memory, in charge of responsive UI and is utilize well for I/O bound applications. Be that as it may, the module isn't killable and is liable to the GIL . Threading library in Python can be define as the various threads live in a similar procedure in a similar space, each thread will complete an explicit task, have its very own code, possess stack memory, guidance pointer, and offer load memory. In the event, that a thread has a memory spill it can harm alternate thread and parent process.

Multitasking is an intelligent augmentation of multi programming. The significant manner by which multitasking varies from multiprogramming is that multi programming works exclusively on the idea of context exchanging though multitasking depends on time sharing nearby the idea of context exchanging.

**Methodology:**

Time-sharing is the primary idea and advantage of MOS. All tasks give a reasonable measure of time and no waiting time happens for the CPU. Multiple clients running various programs can be handling by MOS. All programs run easily without a glitch in execution. All clients of OS give an appropriate measure of time. Multiple programs like MS Word, MS Excel, Photoshop, program, recreations, and adding machine can keep running at the equivalent time. OS runs easily in utilizing multitasking. All kind of PC clients end up fulfilled. Either client can run a single program or numerous projects they do not face any problem in utilizing a PC.

Multithreaded applications can take full advantage of various processors to increase better execution through concurrent execution of tasks. A very much executed multithreaded application effectively utilizes every one of the processors accessible for its own tasks where a single-threaded application must wait for each task to complete before proceeding with whatever remains with the application. At no time can a single-threaded application execute on in excess of one processor in the system.