

**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.

**Literature Review:**

**Time Sharing:**

For specific applications, time-sharing can be enhancing by utilizing multithreading. In different applications, execution can be unaffected or even debased by utilizing multithreading. How execution is influence relies upon your application.

Different clients running numerous projects can be best take care of by multitasking. All projects run easily without a glitch in execution. All clients of OS had given a reasonable measure of time.

**Limitation of the Processor:**

Multithreading gives enhancement in the execution of the processor by synchronous execution of calculation and the I/O tasks.

In multitasking, the event that is in the processor is moderate in the PC, it can process programs moderate and deal with various projects takes longer time. Some substantial projects cannot run easily on the moderate processor since they require even more preparing forces.

**Simultaneous Access of Multiple**

**Applications:**

Multithreading gives access to numerous applications in the meantime in light of fast setting exchanging among threads.

In multitasking, multiple programs can run at the same time.

**Reliability:**

In multithreading, threads might run parallel on numerous processors. If there is just a single thread in a task, it is absurd to expect to separate the procedures into smaller tasks that distinctive processors can perform. Single thread process can run just on one processor despite of what number of processors is accessible. Multithreading on a numerous CPU machine builds parallelism.

OS runs easily in utilizing performing multitasking. All sort of PC clients wind up fulfilled. Either client can run a single program or different program; they do not feel any fault in utilizing a PC.

**Memory and Resource:**

In multithreading, system needs to allot memory to a process, various threads of that process has a similar memory and resources designated to the process

In performing multitasking, system needs to dispense separate memory and resources to each program that CPU is executing.

**Conclusion:**

Threading is comfortable to work with, however threads do not genuinely execute in parallel. Python is a fantastic scripting dialect that appears to be particularly reasonable for Linux applications. You may have heard that python has multithreading support, which is valid, and could significantly enhance your application's execution.

Python utilizes multitasking it does not share its memory, and while at least on systems with fork ( ) system all it can utilize duplicate on-compose enhancement, passing data between processes dependably summons a to a great degree ease back pickle-un pickle routine to pack and send the information.

**References:**

1. <http://www.itrelease.com/2018/07/advantages-and-disadvantages-of-the-multitasking-operating-system/>
2. <https://csharpdotnet.wordpress.com/2008/11/03/advantages-of-multithreading/>
3. <https://www.tutorialspoint.com/operating_system/os_multi_threading.htm>
4. <https://www.geeksforgeeks.org/operating-system-benefits-multithreading/>
5. <https://techdifferences.com/difference-between-multitasking-and-multithreading-in-os.html>