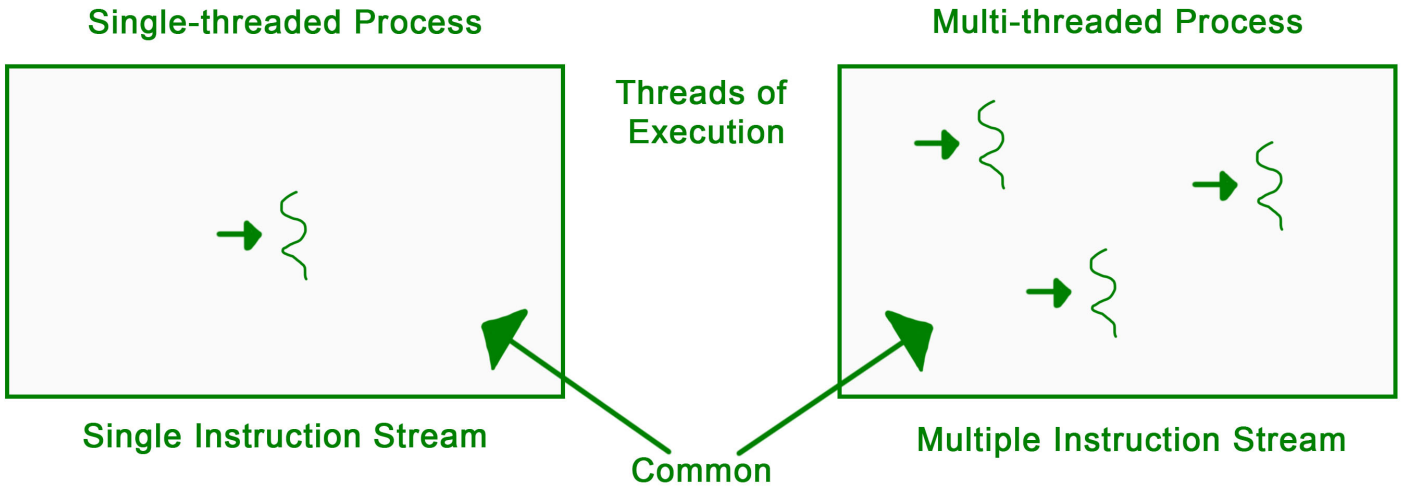
Multithreading

- Python is a multi-threaded programming language which means we can develop multi-threaded program using python.
- A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.
- By definition, multitasking is when multiple processes share common processing resources such as a CPU.
- Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads.
- Each of the threads can run in parallel.
- The OS divides processing time not only among different applications, but also among each thread within an application.
- Multi-threading enables us to write in a way where multiple activities can proceed concurrently in the same program.



Single Thread and Multi Thread Process

Thread :

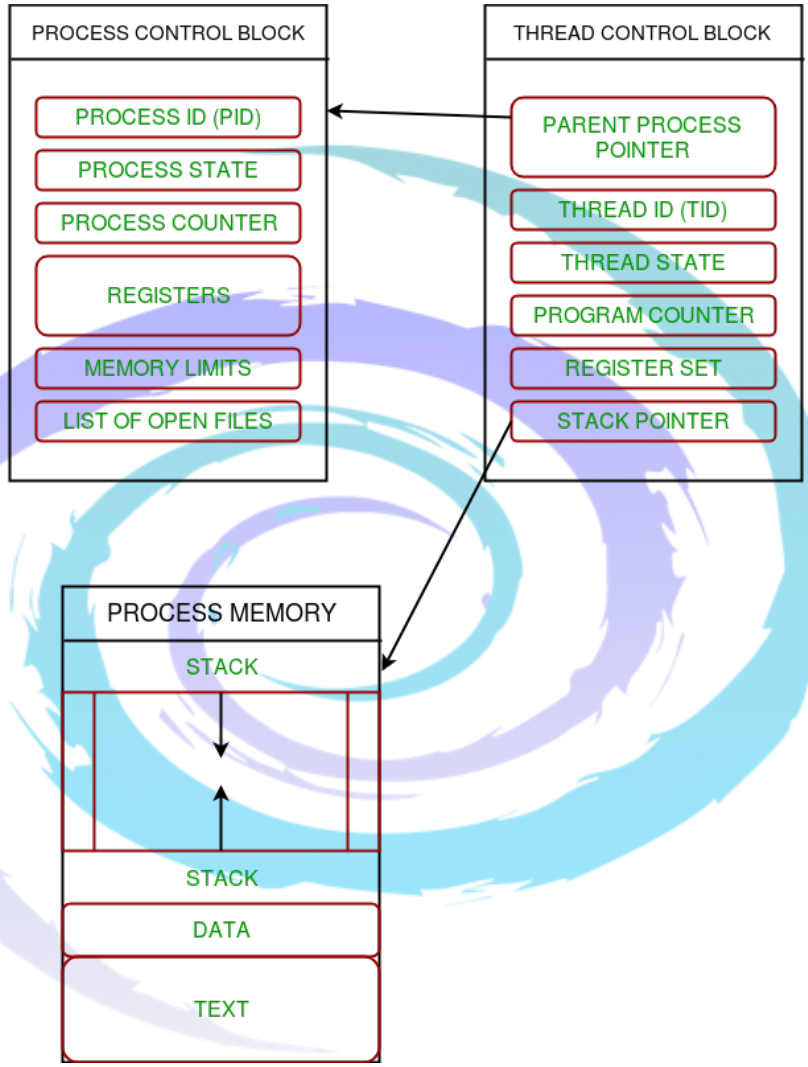
- A thread is an entity within a process that can be scheduled for execution.
- Also, it is the smallest unit of processing that can be performed in an OS (Operating System).
- In simple words, a thread is a sequence of such instructions within a program that can be executed independently of other code.
- For simplicity, we 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.

Consider the below diagram to understand the relation between process and its thread



Consider below application which demonstrates the concept of Multithreading

```
import threading

print("---- Marvellous Infosystems by Piyush Khairnar-----")

print("Demonstration of Multithreading")

def fun(number):
    for i in range(number):
        print(i)

def gun(number):
    for i in range(number):
        print(i)

if __name__ == "__main__":

    number = 5
    thread1 = threading.Thread(target=fun, args=(number,))

    thread2 = threading.Thread(target=gun, args=(number,))

    # Will execute both in parallel
    thread1.start()
    thread2.start()

    # Joins threads back to the parent process, which is this
    # program
    thread1.join()
    thread2.join()
```

Output of above application

```
MacBook-Pro-de-MARVELLOUS:Today marvellous$ python
multithreading.py
---- Marvellous Infosystems by Piyush Khairnar-----
Demonstration of Multithreading
0
 01

1
22

3
4
3
4
MacBook-Pro-de-MARVELLOUS:Today marvellous$
```