#### ****MultiThread In Python****

**A program or process's smallest unit is called a thread, and it can run on its own or as part of a schedule set by the Operating System. Multitasking in a computer system is achieved by dividing a process into threads by an operating system. A string is a lightweight cycle that guarantees the execution of the interaction independently on the framework. When multiple processors are running on a program in Python 3**

**There are two main modules of multithreading used to handle threads in**[Python](https://www.javatpoint.com/python-tutorial)**.**

1. **The thread module**
2. **The threading module**

****Syntax(thread)****

1. thread.start\_new\_thread(function\_name,args[,kwargs])

### **2. Threading Modules**

The threading module is a high-level implementation of multithreading used to deploy an [application in Python](https://www.javatpoint.com/python-applications). To use multithreading, we need to import the threading module in [Python Program](https://www.javatpoint.com/python-programs).

****Thread Class Methods****

|  |  |
| --- | --- |
| Methods | Description |
| ****start()**** | A start() method is used to initiate the activity of a thread. And it calls only once for each thread so that the execution of the thread can begin. |
| ****run()**** | A run() method is used to define a thread's activity and can be overridden by a class that extends the threads class. |
| ****join()**** | A join() method is used to block the execution of another code until the thread terminates. |

****Example:****

**Example:**

import threading

import time

def print\_messages(thread\_name, delay):

count = 0

while count < 5:

time.sleep(delay)

count += 1

print(f"{thread\_name}: {time.ctime(time.time())}")

# Create threads

thread1 = threading.Thread(target=print\_messages, args=("Thread-1", 2))

thread2 = threading.Thread(target=print\_messages, args=("Thread-2", 4))

# Start threads

thread1.start()

thread2.start()

# Wait for both threads to finish

thread1.join()

thread2.join()

**print("Both threads have finished.")**

****Implementation of decorator and threading using time module****

**The threadingmodule provides a Timerclass, which is a subclass of the Threadclass. The Timerclass allows you to run a function after a specified interval.**

****Sample Example:****

**import threading**

**def print\_message():**

**print("Hello, world!")**

**# Create a timer that runs print\_message after 5 seconds**

**t = threading.Timer(5.0, print\_message)**

**# Start the timer**

**t.start()**

****Example:2****

**import threading**

**def print\_message():**

**print("Hello, world!")**

**# Create a timer that runs print\_message after 5 seconds**

**t = threading.Timer(5.0, print\_message)**

**# Start the timer**

**t.start()**

**# Simulate user input**

**user\_input = input("Press Enter to cancel the timer: ")**

**# Cancel the timer if user presses Enter**

**t.cancel()**

**print("Timer canceled.")**