***EXPERIMENT NO : 4B***

***Python programs to implement multithreaded application in python.***

***NAME : AKASH RAMKRIT YADAV ID.NO: VU4F2122016***

***BATCH : A BRANCH : IT DIV : A***

***Aim :- Python python programs to implement multithreaded application in***

***python.***

***THEORY:***

***OUTPUT:***

*Python 3.11.0a4 (main, Mar 1 2023, 10:57:32) [MSC v.1929 32 bit (Intel)] on win32*

*Type "help", "copyright", "credits" or "license()" for more information.*

*#AKASH YADAV ID.NO:VU4F2122016 EXP:4B DATE: 1/3/2023*

# ***Multithreading in Python***

*A thread is the smallest unit of a program or process executed independently or scheduled by the Operating System. In the computer system, an Operating System achieves multitasking by dividing the process into threads. A thread is a lightweight process that ensures the execution of the process separately on the system. In Python 3, when multiple processors are running on a program, each processor runs simultaneously to execute its tasks separately.*

*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*

### ***Thread modules***

*It is started with Python 3, designated as obsolete, and can only be accessed with****\_thread****that supports backward compatibility.*

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

*To implement the thread module in Python, we need to import a****thread****module and then define a function that performs some action by setting the target with a variable.*

***Thread.py***

*import thread # import the thread module*

*import time# import time module*

*import math*

*def cal\_sqre(num): # define the cal\_sqre function*

*print(" Calculate the square root of the given number")*

*for n in num:*

*time.sleep(0.3) # at each iteration it waits for 0.3 time*

*print(' Square Root is : ',math.sqrt(n))*

*def cal\_cube(num): # define the cal\_cube() function*

*print(" Calculate the cube root of the given number")*

*for n in num:*

*time.sleep(0.3) # at each iteration it waits for 0.3 time*

*print(" Cube Root is : ", n\*\*(1/3) )*

*arr = [16, 27, 64, 8, 2] # given array*

*t1 = time.time() # get total time to execute the functions*

*cal\_sqre(arr) # call cal\_sqre() function*

*cal\_cube(arr) # call cal\_cube() function*

*print(" Total time taken by threads is :", time.time() - t1) # print the total time*

***OUTPUT:***

*[Running] python -u "c:\Users\lenovo\Downloads\thread.PY"*

*Calculate the square root of the given number*

*Square Root is : 2.8284271247461903*

*Square Root is : 5.196152422706632*

*Square Root is : 8.0*

*Square Root is : 2.0*

*Square Root is : 1.4142135623730951*

*Calculate the cube root of the given number*

*Cube Root is : 2.0*

*Cube Root is : 3.0*

*Cube Root is : 3.9999999999999996*

*Cube Root is : 1.5874010519681994*

*Cube Root is : 1.2599210498948732*

*Total time taken by threads is : 3.013620138168335*

*Calculate the square root of the given number*

*Square Root is : 2.8284271247461903*

*Square Root is : 5.196152422706632*

*Square Root is : 8.0*

*Square Root is : 2.0*

*Square Root is : 1.4142135623730951*

*Calculate the cube root of the given number*

*Cube Root is : 2.0*

*Cube Root is : 3.0*

*Cube Root is : 3.9999999999999996*

*Cube Root is : 1.5874010519681994*

*Cube Root is : 1.2599210498948732*

*Total time taken by threads is : 3.0457797050476074*

*[Done] exited with code=0 in 6.816 seconds*

***THREADING:***

### ***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.* |

*Follow the given below steps to implement the threading module in Python Multithreading:*

***1. Import the threading module***

*Create a new thread by importing the****threading****module, as shown.*

***Syntax:***

***import****threading*

*A****threading****module is made up of a****Thread****class, which is instantiated to create a Python thread*

***2. Declaration of the thread parameters:****It contains the target function, argument, and****kwargs****as the parameter in the****Thread()****class.*

* ***Target****: It defines the function name that is executed by the thread.*
* ***Args****: It defines the arguments that are passed to the target function name.*

***For example:***

1. ***import****threading*
2. *def print\_hello(n):*
3. *print("Hello, how old are you ", n)*
4. *t1 = threading.Thread( target = print\_hello, args =(18, ))*

*In the above code, we invoked the****print\_hello()****function as the target parameter. The****print\_hello()****contains one parameter****n****, which passed to the****args****parameter.*

***3. Start a new thread:****To start a thread in Python multithreading, call the thread class's object. The start() method can be called once for each thread object; otherwise, it throws an exception error.*

***Syntax:***

1. *t1.start()*
2. *t2.start()*

***4. Join method:****It is a join() method used in the thread class to halt the main thread's execution and waits till the complete execution of the thread object. When the thread object is completed, it starts the execution of the main thread in Python.*

***Joinmethod.py***

1. ***import****threading*
2. *def print\_hello(n):*
3. *Print("Hello, how old are you,what is your name? ", n)*
4. *T1 = threading.Thread( target = print\_hello, args = (20, ))*
5. *T1.start()*
6. *T1.join()*
7. *Print("Thank you")*

***Output:***

*Hello, how old are you, what is your name? 20 ,,AKASH YADAV*

*Thank you*

*When the above program is executed, the join() method halts the execution of the main thread and waits until the thread t1 is completely executed. Once the t1 is successfully executed, the main thread starts its execution.*

#### **Note: If we do not use the join() method, the interpreter can execute any print statement inside the Python program. Generally, it executes the first print statement because the interpreter executes the lines of codes from the program's start.**

***5. Synchronizing Threads in Python***

*It is a thread synchronization mechanism that ensures no two threads can simultaneously execute a particular segment inside the program to access the shared resources. The situation may be termed as critical sections. We use a race condition to avoid the critical section condition, in which two threads do not access resources at the same time.*

*Let's write a program to use the threading module in Python Multithreading.*

***CODE:***

*import time # import time module*

*import threading*

*import math*

*from threading import \**

*def cal\_sqre(num): # define a square calculating function*

*print("Code by Akash yadav")*

*print(" Calculate the square root of the given number")*

*for n in num: # Use for loop*

*time.sleep(0.3) # at each iteration it waits for 0.3 time*

*print(' Square Root is : ', math.sqrt(n))*

*def cal\_cube(num): # define a cube calculating function*

*print(" Calculate the cube Root of the given number")*

*for n in num: # for loop*

*time.sleep(0.3) # at each iteration it waits for 0.3 time*

*print(" Cube Root is : ", n\*\*(1/3))*

*ar = [16, 8,27 , 64, 5] # given array*

*t = time.time() # get total time to execute the functions*

*#cal\_cube(ar)*

*#cal\_sqre(ar)*

*th1 = threading.Thread(target=cal\_sqre, args=(ar, ))*

*th2 = threading.Thread(target=cal\_cube, args=(ar, ))*

*th1.start()*

*th2.start()*

*th1.join()*

*th2.join()*

*print(" Total time taking by threads is :", time.time() - t) # print the total time*

*print(" Again executing the main thread")*

*print(" Thread 1 and Thread 2 have finished their execution.")*

***OUTPUT:***

*Code by Akash yadav*

*Calculate the square root of the given number*

*Calculate the cube Root of the given number*

*Square Root is : 4.0*

*Cube Root is : 2.5198420997897464*

*Square Root is : 2.8284271247461903*

*Cube Root is : 2.0*

*Square Root is : 5.196152422706632*

*Cube Root is : 3.0*

*Square Root is : 8.0*

*Cube Root is : 3.9999999999999996*

*Square Root is : 2.23606797749979*

*Cube Root is : 1.7099759466766968*

*Total time taking by threads is : 1.501918077468872*

*Again executing the main thread*

*Thread 1 and Thread 2 have finished their execution.*