



Experiment No: 8

Aim: Write a program to implement Threading in python.

Theory:

Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching). Besides, it allows sharing of its data space with the main threads inside a process that share information and communication with other threads easier than individual processes. Multithreading aims to perform multiple tasks simultaneously, which increases performance, speed and improves the rendering of the application.

There are two main modules of multithreading used to handle threads in **Python**.

The thread module

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:

1. `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.

Threading Modules

The threading module is a high-level implementation of multithreading used to deploy an **application in Python**. To use multithreading, we need to import the threading module in **Python Program**.

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.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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:

1. **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.

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.

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.



PROGRAM

```
import threading
import time

exitFlag = 0

class myThread (threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        print ("Starting " + self.name)
        print_time(self.name, self.counter, 5)
        print ("Exiting " + self.name)

def print_time(threadName, delay, counter):
    while counter:
        if exitFlag:
            threadName.exit()
        time.sleep(delay)
        print ("%s: %s" % (threadName, time.ctime(time.time())))
        counter -= 1

# Create new threads
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 2)

# Start new Threads
thread1.start()
thread2.start()
thread1.join()
thread2.join()
print ("Exiting Main Thread")
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

OUTPUT:

```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.p
Starting Thread-1
Starting Thread-2
Thread-1: Sat Apr 20 12:17:20 2024
Thread-2: Sat Apr 20 12:17:21 2024
Thread-1: Sat Apr 20 12:17:21 2024
Thread-1: Sat Apr 20 12:17:22 2024
Thread-2: Sat Apr 20 12:17:23 2024
Thread-1: Sat Apr 20 12:17:23 2024
Thread-1: Sat Apr 20 12:17:24 2024
Exiting Thread-1
Thread-2: Sat Apr 20 12:17:25 2024
Thread-2: Sat Apr 20 12:17:27 2024
Thread-2: Sat Apr 20 12:17:29 2024
Exiting Thread-2
Exiting Main Thread
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> █
```

Conclusion:

The experiment successfully demonstrated the implementation of threading in Python, showcasing its ability to execute multiple tasks concurrently and improve program efficiency. Through this exercise, the benefits of threading in enhancing performance and resource utilization were clearly evident, emphasizing its importance in modern software development.