Module 06 – Threads

What is Thread

- A Thread or a Thread of Execution is defined in computer science as the smallest unit that can be scheduled in an operating system

- Threads are contained in processes. More than one thread can exist within the same process.

- These threads share the memory and the state of the process.

threading Module

- Threading module is a simple way to create threads. Threading APIs very similar to multiprocessing API
- Using threads allows a program to run multiple operations concurrently in the same process space.
- To create a new thread in out program we should use the Tread. class of Threading module

Thread class

- Thread(group=None, target=None, name=None, args=())
  - target is the callable object to be invoked by the Process
  - name is the process name
  - **args** is the argument tuple for the target invocation.
  - group — should be always be None

- Thread has start and join functions, exactly like Process does

Threading Module example

Demo

Threading Module example

```
import time
import threading

global_num = 10
def func():
   global global_num
   Global_num = 11

thread1 = threading.Thread(target=func)
thread1.start()
thread1.join()
print(global_num)
```

Threading module Synchronization

- threading module has 3 classes for threads synchronization, like multiprocessing module
  - Lock - non-recursive lock object
  - Rlock - recursive lock object
  - Semaphore — created with internal counter and can be acquired counter times before released

```
lock = threading.Lock()
with lock:
        # critical section code
```

The Global Interpreter Lock

- In CPython, the Global Interpreter Lock (GIL), is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once.
- The GIL is controversial because it prevents multithreaded CPython programs from taking full advantage of multiprocessor systems
- There are some GIL free operations, such as I/O and image processing. They happen outside the GIL.
- The multithreaded programs that spend a lot of time inside the’ i GIL, interpreting CPython bytecode, that the GIL becomes a bottleneck

Console Methods

Demo

Lab 01

Lab

Questions