Threads

A process is an active program i.e. a program that is under execution. It is more than the program code as it includes the program counter, process stack, registers, program code etc. Compared to this, the program code is only the text section.

A thread is a lightweight process that can be managed independently by a scheduler. It improves the application performance using parallelism. A thread shares information like data segment, code segment, files etc. with its peer threads while it contains its own registers, stack, counter etc.

For simplicity, you can assume that a thread is simply a subset of a process!

```
def print_cube(num):

        print("cube of a number",num * num * num)

def print_square(num):



        print("cube of a number",num * num)

if __name__ == "__main__":

    # creating thread

    t1 = threading.Thread(target=print_square, args=(10,))

    t2 = threading.Thread(target=print_cube, args=(10,))

    # starting thread 1

    t1.start()

    # starting thread 2

    t2.start()

    # wait until thread 1 is completely executed

    t1.join()

    # wait until thread 2 is completely executed

    t2.join()

    # both threads completely executed
```

```
    print("Done!")
```

To create a new thread, we create an object of **Thread** class. It takes following arguments:

- **target**: the function to be executed by thread
- **args**: the arguments to be passed to the target function

To start a thread, we use **start** method of **Thread** class.

Once the threads start, the current program (you can think of it like a main thread) also keeps on executing. In order to stop execution of current program until a thread is complete, we use **join** method.

As a result, the current program will first wait for the completion of **t1** and then **t2**. Once, they are finished, the remaining statements of current program are executed.

```
import threading
import os

def task1():
    print(threading.current_thread().name)

def task2():
    print(threading.current_thread().name)


if __name__ == "__main__":

    # print ID of current process
    print("ID of process running main program:",os.getpid())

    # print name of main thread
    print("Main thread name:",threading.current_thread().name)
    # get the current thread object

    # creating threads
    t1 = threading.Thread(target=task1, name='t1')
    t2 = threading.Thread(target=task2, name='t2')

    # starting threads
    t1.start()
    t2.start()

    # wait until all threads finish
    t1.join()
    t2.join()
```

Concept of global

1)

```
x=20 # global
def f1():
        print(x)

f1() # 20
```

2)

```
x=20 # global
def f1():
        x=x+1 # error
        print(x)

f1()
```

3)

```
x=20 # global
def f1():
        global x
        x=x+1
        print(x)

f1() # 21
```

Example 3:

```
import threading
# global variable x
x = 0
def increment_global():
  global x
  x += 1
def taskofThread():
  for _ in range(1000000):
    increment_global()
def main():
  global x
  x = 0
  t1 = threading.Thread(target= taskofThread)
  t2 = threading.Thread(target= taskofThread)
  t1.start()
  t2.start()
```

```
  t1.join()
  t2.join()
if __name__ == "__main__":
  for i in range(10):
    main()
    print(i,x)
```

```
import time
import threading
from threading import Thread

def i_am_thread():
        time.sleep(10)
        print("this is thread")

th = Thread(target= i_am_thread,name='fast')
th.start()
print("Current Thread count:",threading.active_count())
print("state:",th.is_alive())
th.join()

print("state:",th.is_alive())

print("get name:",th.getName())
print("Main thread name:",threading.main_thread().name)
```