USTHB
Faculté d'Electronique et Informatique **Année 2019/2020**
Département Informatique **Master SSI**

TP3 : gérer les threads

1. Écrivez le programme suivant:

```
# Python program to illustrate the concept
# of threading
import threading
import os

def task1():
    print("Task 1 assigned to thread: {}".format(threading.current_thread().name))
    print("ID of process running task 1: {}".format(os.getpid()))

def task2():
    print("Task 2 assigned to thread: {}".format(threading.current_thread().name))
    print("ID of process running task 2: {}".format(os.getpid()))

if __name__ == "__main__":

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

    # print name of main thread
    print("Main thread name: {}".format(threading.main_thread().name))

    # 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()
```

Enregistrez votre programme dans le fichier thread`.py` en allant dans le menu `File/save`. Vous prendrez soin d'enregistrer votre programme dans le dossier `tp3` que vous devez créer.

Exécuter le programme thread.py , noter le résultat, conclure.

2. Soit le programme suivant

```
import threading

# global variable x
x = 0
```

```
def increment():
    """
    function to increment global variable x
    """
    global x
    x += 1

def thread_task():
    """
    task for thread
    calls increment function 100000 times.
    """
    for _ in range(100000):
        increment()

def main_task():
    global x
    # setting global variable x as 0
    x = 0

    # creating threads
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

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

    # wait until threads finish their job
    t1.join()
    t2.join()

if __name__ == "__main__":
    for i in range(10):
        main_task()
            print("Iteration {0}: x = {1}".format(i,x))
```

Exécuter le programme noter le résultat et conclure

Soit le verrou lock défini dans la classe threading

```
    # creating a lock
    lock = threading.Lock()

    # creating threads
    t1 = threading.Thread(target=thread_task, args=(lock,))
    t2 = threading.Thread(target=thread_task, args=(lock,))
```

modifier le programme précèdent en introduisant les instructions ci dessus et d'autres modifications dans les thread pour synchroniser les threads

exécuter et vérifier que la synchronisation a été respectée

```
import threading

# global variable x
x = 0

def increment():
    """
    function to increment global variable x
    """
    global x
    x += 1

def thread_task(lock):
    """
    task for thread
    calls increment function 100000 times.
    """
    for _ in range(100000):
        lock.acquire()
        increment()
        lock.release()

def main_task():
    global x
    # setting global variable x as 0
    x = 0

    # creating a lock
    lock = threading.Lock()

    # creating threads
    t1 = threading.Thread(target=thread_task, args=(lock,))
    t2 = threading.Thread(target=thread_task, args=(lock,))

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

    # wait until threads finish their job
    t1.join()
    t2.join()

if __name__ == "__main__":
    for i in range(10):
        main_task()
        print("Iteration {0}: x = {1}".format(i,x))
```