Practical – 11: Fundamentals of threading

Instructions:

1. Use IDE of your choice (notepad/IDLE/vscode/pycharm) for editing following programs.
   - I have used vscode editor.
   - Use file extension '.py' for saving the following programs.
2. Use 'Anaconda Prompt' for testing the following commands.
3. E.g. for running a python file program-1.py, issue following command in the **'Anaconda Prompt'**:
   ```
   python program-1.py
   ```

----

* How to Find currently running thread's name?

```
import threading

# # Find the name of present thread
# print("Current thread name:", threading.current_thread().getName())

print("Current thread name:", threading.current_thread())


# # Check if it is a main thread or not
if threading.current_thread() == threading.main_thread():
    print("The current thread is the main thread.")
else:
    print("The current thread is not main thread.")
```

* How to create a thread without using a class?

```
from threading import *

def display():
    print("display() is running.:",current_thread().getName())
##


t = Thread(target=display)
t.start()

print('Bye.')
```

* How to create multiple threads in a loop without using a class?

```
from threading import *

def display():
    print("display() is running.:",current_thread().getName())
##
```

```python
for i in range(5):
    t = Thread(target=display)
    t.start()

print('Bye.')
```

* How to create thread and pass a single parameter to it?

```python
from threading import *

def display(msg):
    print(msg)
##

for i in range(5):
    t = Thread(target=display, args=("Hello",))
    t.start()
```

Notice the args parameter. It accepts a tuple.

* How to create thread and pass multiple parameters to it?

```python
from threading import *

def display(msg,i):
    print(msg,i)
##

for i in range(5):
    t = Thread(target=display,args=("Hello",i))
    t.start()
```

* How to create a thread by creating a subclass of Thread class?

```python
from threading import Thread

# Create a subclass of the Thread class
class MyThread(Thread):
    #Override the run() method of Thread class
    def run(self):
        for i in range(1,10):
            print(self.getName(), i)
    ##
##
```

```python
# Create an instance of the MyThread class
t1 = MyThread()

# Submit the thread thread t1 for execution
t1.start()

print("Bye.")
```

Run the above program several times and observe the variations in output obtained during each execution.

* How to make the main thread to wait for the other threads?

```python
from threading import Thread
import threading

# Create a subclass of the Thread class
class MyThread(Thread):
    #Override the run() method of Thread class
    def run(self):
        for i in range(1,10):
            print(self.getName(), i)
    ##
##

# Create an instance of the MyThread class
t1 = MyThread()

# Start running the thread t1
t1.start()

# wait till the thread t1 completes execution
t1.join()

print("Hi from mainthread.")
print("Bye.")
```

One more example:

```python
from threading import Thread
import threading

# Create a subclass of the Thread class
class MyThread(Thread):
    #Override the run() method of Thread class
    def run(self):
        for i in range(1,10):
            print(self.getName(), i)
    ##
```

```
##

# Create an instance of the MyThread class
t1 = MyThread()
t2 = MyThread()

# Start running the thread t1 & t2
t1.start()
t2.start()

# wait till the thread t1 & t2 complete execution
t1.join()
t2.join()

print("Hi from mainthread.")
print("Bye.")
```

* How to create a thread by creating a subclass of thread?

```
from threading import Thread

# Create a subclass of the Thread class
class MyThread(Thread):
    # Constructor that calls the Thread class constructor
    def __init__(self,str):
        Thread.__init__(self)
        self.str = str
    ##

    #Override the run() method of Thread class
    def run(self):
        for i in range(1,6):
            print(self.str)
    ##
##

# Create an instance of the MyThread class
t1 = MyThread("Ahimsak andolan")
t2 = MyThread("Jaliya  vala baug")

# Start running the thread t1
t1.start()
t2.start()

# wait till the thread completes execution
t1.join()
t2.join()
```

```
print("Independent India!")
```

* How to create a thread without creating a subclass of Thread?

```
from threading import *

# Create our own thread class
class MyThread():
    # Constructor that calls the Thread class constructor
    def __init__(self,str):
        self.str = str
    ##

    #a method
    def display(self):
        for i in range(5):
            print(self.str,i)
    ##
##

# Create an instance of the MyThread class
obj1  = MyThread("Hello")
# Create a thread to run display method of obj
t1 = Thread(target=obj1.display)


# Run the thread
t1.start()

print("Bye.")
```

Another example:

```
from threading import *

# Create our own thread class
class MyThread():
    # Constructor that calls the Thread class constructor
    def __init__(self,str):
        self.str = str
    ##

    #a method
    def display(self):
        for i in range(5):
            print(self.str,i)
    ##
##
```

```python
# Create an instance of the MyThread class
obj1  = MyThread("T1")
# Create a thread to run display method of obj
t1 = Thread(target=obj1.display)
# Run the thread
t1.start()


obj2 = MyThread("T2")
t2 = Thread(target=obj2.display)
t2.start()

t2.join()
t1.join()
print("Bye.")
```

* Demonstrate single tasking:

```python
from threading import *
from time import *

# Create our own class
class MyThread():
    # A method that performs 3 tasks one by one
    def prepareTea(self):
        self.task1()
        self.task2()
        self.task3()
    ##

    def task1(self):
        print("Boil milk and tea powder for 5 minutes..",end='')
        sleep(5)
        print('Done.')
    ##
    def task2(self):
        print("Add sugar and boil for 3 minutes..",end='')
        sleep(3)
        print('Done.')
    ##
    def task3(self):
        print("Filter it and serve.",end='')
        print('Done.')
    ##
##

# Create an instance of our class
obj = MyThread()
```

```python
# Create a thread to run prepareTea() method of obj
t = Thread(target=obj.prepareTea)

# Run the thread
t.start()

print("Bye.")
```

* Demonstrate multithreading:

```python
from threading import *
from time import *

class Theatre:
    def __init__(self,str):
        self.str = str
    ##

    #  a method that repeats for 5 tickets
    def movieshow(self):
        for i in range(1,6):
            print(self.str,":",i)
            sleep(0.2)
    ##
##

# Create two instances of Theatre class
obj1 = Theatre("Cut ticket")
obj2 = Theatre("Show chair")

# Create two threads to run movieshow()
t1 = Thread(target=obj1.movieshow)
t2 = Thread(target=obj2.movieshow)

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

# Let the main program wait for completion of t1 & t2
t1.join()
t2.join()

# print("Bye.")
```

Exercises

1. Who does execute & manage threads in Python; operating system or PVM?
2. What is the role of `start()` and `join()` in the context of `threading.Thread`?
3. Create a Python program for the following requirements:
   a. Create and use a thread `thread_1to10` to print numbers from 1 to 10.
   b. Create and use a thread `thread_50to60` to print numbers from 50 to 60.
   c. Start all the threads, however
      i. The thread `thread_50to60` must finish first.
      ii. The thread `thread_1to10`must finish last.
4. Create a Python program for the following requirements:
   a. Create and use a thread to count vowel characters in a text file.
   b. Create and use a thread to count consonant characters in a text file.
   c. Start all threads, show output and observe your program's behavior.

   Note: Your text file should have sufficient data (You may generate such a file programmatically.)

5. Create a Python program for the following requirements:
   a. Create and use a thread to count words (having length <= 4) in a text file.
   b. Create and use a thread to count (having length 6) in a text file.
   c. Create and use a thread to count (having length >= 6) in a text file.
   d. Start all threads, show output and observe your program's behavior.

   Note: Note: Your text file should have sufficient data (You may generate such a file programmatically.)

6. Create a Python program for the following requirements:
   a. Create and use a thread to insert at least 20000 male student/employee records in a database table.
   b. Create and use a thread to insert at least 20000 female student/employee records in a database table.
   c. Start all threads, show output and observe your program's behavior.
7. Explain need of lock acquiring and releasing in multithreading. Demonstrate & explain using suitable example.