Thread Communication

Two or more threads communicate with each other.

- Event
- Condition
- Queue

Event

This is one of the simplest mechanisms for communication between threads: one thread signals an event and other threads wait for it.

An event object manages an internal flag that can be set to true with the set() method and reset to false with the clear() method. The wait() method blocks until the flag is true.

The flag is initially false.

Create Event Object

```
from  threading import Event
e = Event()
```

Event Methods

set()- It sets the internal flag to true. All threads waiting for it to become true are awakened. Threads that call wait() once the flag is true will not block at all.

clear()- It resets the internal flag to false. Subsequently, threads calling wait() will block until set() is called to set the internal flag to true again.

is_set() – It returns true if and only if the internal flag is true.

Event Methods

wait(timeout=None) – It blocks until the internal flag is true. If the internal flag is true on entry, return immediately. Otherwise, block until another thread calls set() to set the flag to true, or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof).

This method returns true if and only if the internal flag has been set to true, either before the wait call or after the wait starts, so it will always return True except if a timeout is given and the operation times out.

Condition

Condition class is used to improve speed of communication between Threads. The condition class object is called condition variable.

A condition variable is always associated with some kind of lock; this can be passed in or one will be created by default. Passing one in is useful when several condition variables must share the same lock. The lock is part of the condition object: you don't have to track it separately.

A condition is a more advanced version of the event object.

Create Condition Object

```
from  threading import Condition
cv = Condition()
```

Condition Method

- notify(n=1) – This method is used to immediately wake up one thread waiting on the condition. Where n is number of thread need to wake up.

- notify_all() – This method is used to wake up all threads waiting on the condition.

- wait(timeout=None) – This method wait until notified or until a timeout occurs. If the calling thread has not acquired the lock when this method is called, a RuntimeError is raised. Wait terminates when invokes notify() method or notify_all() method. The return value is True unless a given timeout expired, in which case it is False.

Queue

The Queue class of queue module is useful to create a queue that holds the data produced by the producer.

The data can be taken from the queue and utilized by the consumer.

We need not use locks since queues are thread safe.

Create Queue Object:

```
from queue import Queue
q = Queue()
```

Queue Methods

put ( ) – This method is used by Producer to insert items into the queue.
Syntax:- queue_object.put(item)
Ex:- q.put(i)

get ( ) – This method is used by Consumer to retrieve items from the queue.
Syntax:- producer_object.queue_object.get(item)
Ex:- p.q.get(i)

empty() – This method returns True if queue is Empty else returns False.
Ex:- q.empty()

full() – This method returns True if queue is Full else returns False.
Ex:- q.full()