



Python | Communicating Between Threads | Set-1

Last Updated : 12 Jun, 2019



Related Articles

Perhaps the safest way to send data from one thread to another is to use a Queue from the queue library. To do this, create a Queue instance that is shared by the threads. Threads then use `put()` or `get()` operations to add or remove items from the queue as shown in the code given below.

Code #1 :

```
from queue import Queue
from threading import Thread

# A thread that produces data
def producer(out_q):
    while True:
        # Produce some data
        ...
        out_q.put(data)

# A thread that consumes data
def consumer(in_q):
    while True:
        # Get some data
        data = in_q.get()
        # Process the data
        ...

# Create the shared queue and launch both threads
q = Queue()
t1 = Thread(target = consumer, args =(q, ))
t2 = Thread(target = producer, args =(q, ))
t1.start()
t2.start()
```



by as many threads as per requirement. When using queues, it can be somewhat tricky to coordinate the shutdown of the producer and consumer.

A common solution to this problem is to rely on a special sentinel value, which when placed in the queue, causes consumers to terminate as shown in the code below:

Code #2 :

```
from queue import Queue
from threading import Thread

# Object that signals shutdown
_sentinel = object()

# A thread that produces data
def producer(out_q):
    while running:
        # Produce some data
        ...
        out_q.put(data)

    # Put the sentinel on the queue to indicate completion
    out_q.put(_sentinel)

# A thread that consumes data
def consumer(in_q):
    while True:
```



```
if data is _sentinel:
    in_q.put(_sentinel)
    break
...
```

A subtle feature of the code above is that the consumer, upon receiving the special sentinel value, immediately places it back onto the queue. This propagates the sentinel to other consumers threads that might be listening on the same queue—thus shutting them all down one after the other.

Although queues are the most common thread communication mechanism, one can build own data structures as long as one adds the required locking and synchronization. The most common way to do this is to wrap your data structures with a condition variable.

Code #3 : Building a thread-safe priority queue

```
import heapq
import threading

class PriorityQueue:
    def __init__(self):
        self._queue = []
        self._count = 0
        self._cv = threading.Condition()

    def put(self, item, priority):
        with self._cv:
            heapq.heappush(self._queue, (-priority, self._count, item))
            self._count += 1
            self._cv.notify()

    def get(self):
        with self._cv:
            while len(self._queue) == 0:
                self._cv.wait()
            return heapq.heappop(self._queue)[-1]
```

Thread communication with a queue is a one-way and non-deterministic process. In general, there is no way to know when the receiving thread has actually received a



Code #4 :

```
from queue import Queue
from threading import Thread

# A thread that produces data
def producer(out_q):
    while running:
        # Produce some data
        ...
        out_q.put(data)

# A thread that consumes data
def consumer(in_q):
    while True:
        # Get some data
        data = in_q.get()
        # Process the data
        ...
        # Indicate completion
        in_q.task_done()

# Create the shared queue and launch both threads
q = Queue()
t1 = Thread(target = consumer, args =(q, ))
t2 = Thread(target = producer, args =(q, ))
t1.start()
t2.start()

# Wait for all produced items to be consumed
q.join()
```

Attention geek! Strengthen your foundations with the [Python Programming Foundation](#) Course and learn the basics.

To begin with, your interview preparations Enhance your Data Structures concepts with the [Python DS](#) Course. And to begin with your Machine Learning Journey, join the [Machine Learning – Basic Level Course](#)



Start building apps today with 25+ free services and a USD 200 credit.

[LEARN MORE](#)

[HIDE AD](#) • [AD VIA BUYSSELLADS](#)

Python | Communicating Between Threads | Set-2

RECOMMENDED ARTICLES

Page : [1](#) [2](#) [3](#)

01 Python | Communicating Between Threads | Set-2

11, Jun 19

05 How to run same function on multiple threads in Python?

25, Mar 21

02 Releasing GIL and mixing threads from C and Python

28, Mar 19

06 Important differences between Python 2.x and Python 3.x with examples

25, Feb 16

03 Joining Threads in Python

16, Jun 20

07 Communication between Parent and Child process using pipe in Python

20, Dec 17

04 Python Daemon Threads

21, Jan 21

08 Difference between Method and Function in Python

22, Mar 18

Article Contributed By :



manikachandna97

@manikachandna97

C





Start building apps today with 25+ free services and a USD 200 credit.

[LEARN MORE](#)

[HIDE AD](#) • [AD VIA BUYSPELLADS](#)

Easy

Normal

Medium

Hard

Expert

Article Tags : [python-utility](#), [Python](#)

Improve Article

Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments



5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

[About Us](#)

[Careers](#)

[Privacy Policy](#)

[Contact Us](#)

[Copyright Policy](#)

Learn

[Algorithms](#)

[Data Structures](#)

[Languages](#)

[CS Subjects](#)

[Video Tutorials](#)



Start building apps today with 25+ free services and a USD 200 credit.

[LEARN MORE](#)

[HIDE AD](#) • [AD VIA BUYSSELLADS](#)

Company-wise

Write Interview Experience

Topic-wise

Internships

How to begin?

Videos

@geeksforgeeks , Some rights reserved

